ABSTRACT

Deep Neural Network (DNN) can be used to classify images as a simple application in computer vision. This paper forays into the basic building blocks of a simple, single-hidden layer DNN built in Python. Alternative neural net structures within the single-hidden layer framework are explored to understand the effect of certain model parameters. The assignment objective is to understand how the neurons/nodes in a simple single-hidden layer network have learned to represent features within the input data, with an interest in the image pattern that generates maximum activation in the hidden to set the stage of further analysis in Convolution Neural Network in future assignments.

1. INTRODUCTION

In this exercise a Deep Neural Network (DNN) is built, fitted, and tested in the Python coding framework as an initial exploration of the architecture's application in image recognition (i.e. numeric data classification). The neural network does so with two important operations, compute/predict the output value from the input data through forward propagation and compute/adjust loss parameters, also known as "training", through backpropagation. The DNN model and its variations are examined in this paper for their performance in classifying MNIST data (source: http://yann.lecun.com/exdb/mnist/), which consists of a set of 70,000 small (28x28 pixel) grayscale images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents ranging from 0 to 9. 60,000 28x28 grayscale images have 10 balanced digit classes, along with a test set of 10,000 images. The neural net architecture is illustrated in figure 1.

Five sets of experiments are conducted: In Experiments 1 to 3, dense neural network models with one hidden layer but input dimensionality reduction is considered. The number of (hidden) nodes in the layer has been progressively increased from one until the "best" model in identified in Experiment 3. In addition, this exercise explores what the nodes in the hidden layer are "detecting" and what their outputs (i.e. activation values) contribute to the final classification of a digit image.

In Experiment 4, PCA decomposition is used to reduce the number of dimensions of our training set of 28x28 dimensional MNIST images from 784 to 154 (with 95% of training images variance lying along these components). As a result, the number of dimensions of the 'best' model from Experiment 3 is reduced to 154 inputs nodes and train it on the new lower dimensional data.

In Experiment 5, a Random Forest classifier is used to get the relative importance of the 784 features (pixels) of the 28x28 dimensional images in training set of MNIST images and select the top 70 features (pixels). The 'best' model from Experiment 3 using these 70 features is trained and compared to the models created in Experiments 3 and 4.

The overall objective of this assignment is to understand the basic mechanism of deep learning, especially in how hidden nodes affect model prediction performance.

2. LITERATURE REVIEW

In the November 2012 issue of IEEE Signal Processing Magazine "Best of the Web" article, Microsoft researcher Li Deng gave a clear, concise overview of the developments of machine learning research (2012). Within the brief article referenced 7 other publications by authored by

leading machine learning researchers about image classification architecture and convolutional neural networks built with the MNIST dataset.

3. METHODS

The DNN model was built in the Python environment using Keras packages imported from Tensorflow. The NN model consists of 784 input nodes, a hidden layer with varying nodes – starting from 1 - and 10 output nodes corresponding to the 10 digits. Other Python packages including NumPy, Pandas, Matplotlib, Seaborn are used to help with data manipulation and visualization, while Scikit learn and additional Keras packages are used to facilitate data pre-processing and model building.

The MNIST dataset was imported from Keras using the mnist.load_data function. The dataset itself is sourced from Yann LeCun and Corinna Cortes. Both the training set (60,000 images) and the test set (10,000) images along with their labels were first inspected through exploratory data analysis. The image dimension and label frequency were verified to be as expected. A sample of images (figure 2) were displayed with corresponding labels to show a snippet of each set of grayscales 28x28 pixels.

Before each experiment was run, the raw training and test set data were reshaped into an array of shape (60,000, 28*28) and its values normalized between 0 and 1. 5,000 out of the 60,000 training data were held back for validation during model fitting. The model is created by the Sequential class defined in Keras. Both layers are going to be “dense” layers, meaning which all the nodes of a layer are connected to all the nodes of the preceding layer as shown in figure 1. The Rectified Linear Unit (ReLU) is selected as the activation function of the hidden layer, while

Softmax regression is the go-to activation function for a multi-class classification problem such as this assignment, as Softmax assumes that each that each input is a member of exactly one class and assigns the output decimal probabilities to each class.

After setting up the model architecture, the algorithm used optimize the model weights and biases as per the given data was defined to be stochastic gradient descent. The loss function to be minimized by the optimizer was selected as "categorical cross entropy" commonly used in a multi-class classification problem. The model was then trained through 30 epochs, a full cycle of fitting each training data to its label and followed by validation. The "evaluate" method was used to evaluate the trained model performance – accuracy and loss - on the test set.

The model training and validation performance was also inspected by plotting the evolution of accuracy and loss (figure 3) as each epoch completes. As with classification problem analysis in general, a 10x10 confusion matrix was constructed in a heatmap format to visualize the occurrence and error rates of each predicted value against the true value of the label. A sample of misclassified digits were displayed (figure 4) as an eyeball inspection.

Another activity in Experiment 1 and 2 was to group the activation values of the hidden node(s) for the (original) set of training images by the 10 predicted classes. A new "activation model" was created, which takes the same input as our current model but instead outputs the activation value of the hidden layer. The "predict" function was then used on the activation model to obtain the activation values. In Experiment 1, these predicted sets of values in their predicted classes were visualized in matplotlib and seaborn boxplots (figure 5) for analysis. Whereas in Experiment 2, the activation values from both hidden nodes are plotted and color coded against their predicted classes in a scatterplot (figure 6) for analysis.

In Experiment 1, custom functions were defined to find and display the image pattern that maximally activates the hidden node.

In Experiment 3, 7 variations of the model architecture defined in Experiment 1 and 2 were tested to narrow down to a final "best" model as the basis for Experiment 4 and 5.

In Experiment 4, the training set data was transformed with the principal component analysis (PCA) technique to have its number of dimensions reduced from 784 to 154 - a result of retaining 95% of training images variance lying along these components. The transformed dataset was fed into the "best" model from Experiment 3 to 154 input nodes to see the effect of the new lower dimensional data had on model performance.

Similarly, in Experiment 5, the training set data was transformed with a random forest (RF) classifier with 100 "trees" to get the relative importance of the 784 features/pixels (figure 7). The top 70 features (figure 8) were retained and used to train the "best" model from Experiment 3. A comparison was drawn between models from Experiment 3, 4, and 5.

4. RESULTS

From the EDA, the training and test label sets contain balanced classes of digits across 0 to 9, an indicator of classification accuracy will reflect model success (Chollet, 2017).

The model characteristics, accuracy, and loss of each model across Experiment 1 to 5 are listed in Table 1. It also shows the number of epochs for each model beyond which the model is overfitting: the training and validation losses start to diverge considerably (Chollet, 2017), leading to a diminished model performance in the loss of generalization to unseen data. Whereas Table 2 lists the models in Experiment 4 and 5 with inputs of reduced dimensionality into the

"best" model architecture from Experiment 3. Comparing the summarized model attributes from both tables lends insight to understanding how the model hyperparameters, namely the number of hidden nodes and type of optimization algorithm, affect the model output performance.

From Table 1, it is observed that increasing the number of hidden nodes in the varying models of Experiment 3 yielded a diminishing performance improvement. The initial accuracy improvement was drastic (from 39.7% to 95.7%) when the number of hidden nodes went from 1 to 20, but from 20 to 100 nodes the model accuracy merely had a 2% rate increase from 95.7% to 97.7% at the cost of more training parameters and more epochs to reach the optimal model fit – both translate to a longer model training time. For example, after 30 epochs, the initial model in Experiment 1 still failed to reach the convergence between training and validation loss, while those in Experiment 3 only needed on average 5 epochs to get an optimal fit. Moreover, the gradient descent optimization algorithm is changed from RMSProp to SGD (Stochastic Gradient Descent) or Adam (Adaptive Moment Estimation). Empirically Adam led to the "best" model performance out of these methods as it retained the same accuracy as the RMSProp model as well as further minimized loss and number of epochs needed.

From Table 2, the input data are transformed by reducing its dimensionality with two different machine learning techniques: PCA decomposition in Experiment 4 and RF classification in Experiment 5. PCA successfully reduced the dimension from 784 to 154 features with 95% variance retained. This in turn streamlined the model to 154 input nodes while maintain the same level of accuracy with no observed negative impact to the "best" model in Experiment 3. On the other hand, the RF classification resulted in a performance loss in about 4% accuracy with more training epochs despite it successfully reduced the input to 70 most important features. It was also noticed that the RF classification is much less time efficient in

classifying the image features relative importance than PCA in generating the principal components. Although the experiment results suggested that PCA is a more desirable technique in pre-processing the input data, one should caution that a fair comparison should have the RF decision tree select the top 154 features, same number as the PCA model.

Lastly, useful insights were gained from a further analysis of the class error rates presented in confusion matrices, and from examination of the activation values visualization in boxplots and scatterplots constructed for Experiment 1 and 2. The original model set up in Experiment 1 had a disappointing 40% accuracy, a big part of which can be attributed to the model failing to detect 0's and 5's as observed in its confusion matrix (figure 9). As the number of connected nodes increases in the hidden layer, the model prediction becomes more capable as a higher level of complexities in the input-output relationship can be captured by the model in general. It is reflected in the confusion matrices of the model variants in Experiment 3, where the label error rate is miniscule. And to our expectation, the boxplot (figure 5) and scatterplot (figure 6) of the activation values in relation to the class label has confirmed minimal overlapping. The visual pattern in the scatterplot has in particularly pointed to how the node outputs contribute and explain to some extent the final classification of an image. An effort was made to highlight the visual pattern using a custom gradient descent function that maximizes the activation value of the hidden node to determine what the hidden node is "detecting", but the insights generated are of less significance in the context of a DNN.

5. CONCLUSION

By building a fully connected DNN structure with a single-hidden layer, applying the model to train and test on the well-studied MNIST image dataset, and going through the heuristic

process of varying certain model hyperoperators to study the model behavior, we are given the opportunity to explore neural nets on a practical programming level particularly in the application of computer vision. Owing to the simplicity of a single hidden layer, we can take further steps with relative ease to analyze hidden nodes and activation values, allowing us to draw a linkage between their output and the effect to the final class prediction. This exercise sets the stage for building the foundational understanding of advanced neural networks that are more complex in later part of the course.

Appendix

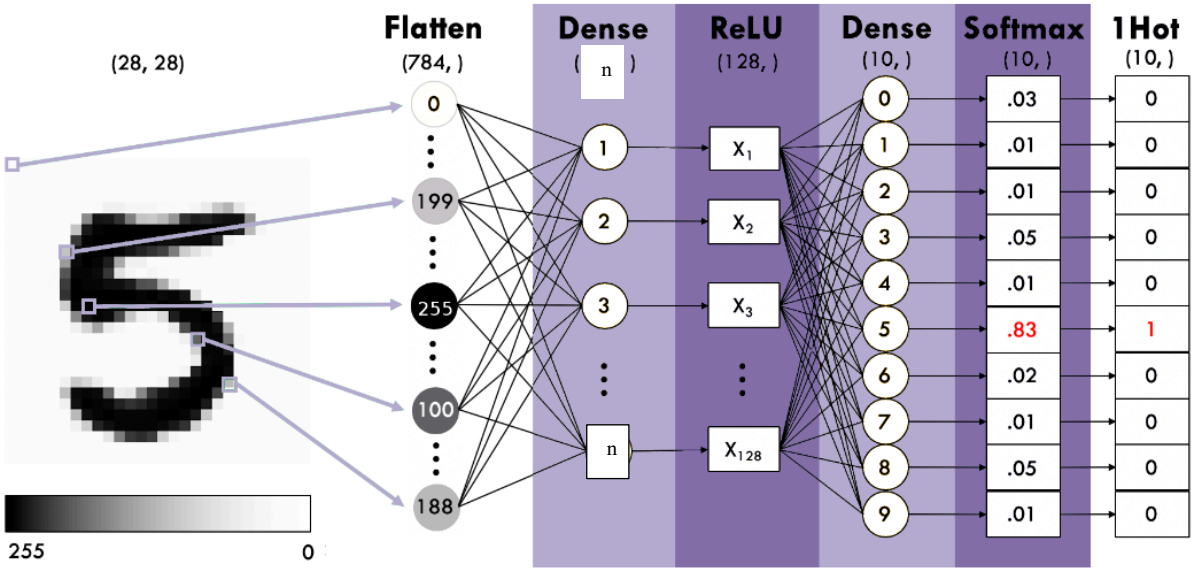

Figure 1. DNN Architecture (adapted from https://github.com/djp840/MSDS_458_Public/blob/master/MSDS458_Assignment_01/MSDS458_Assignment_01_v26_20220906.ipynb).

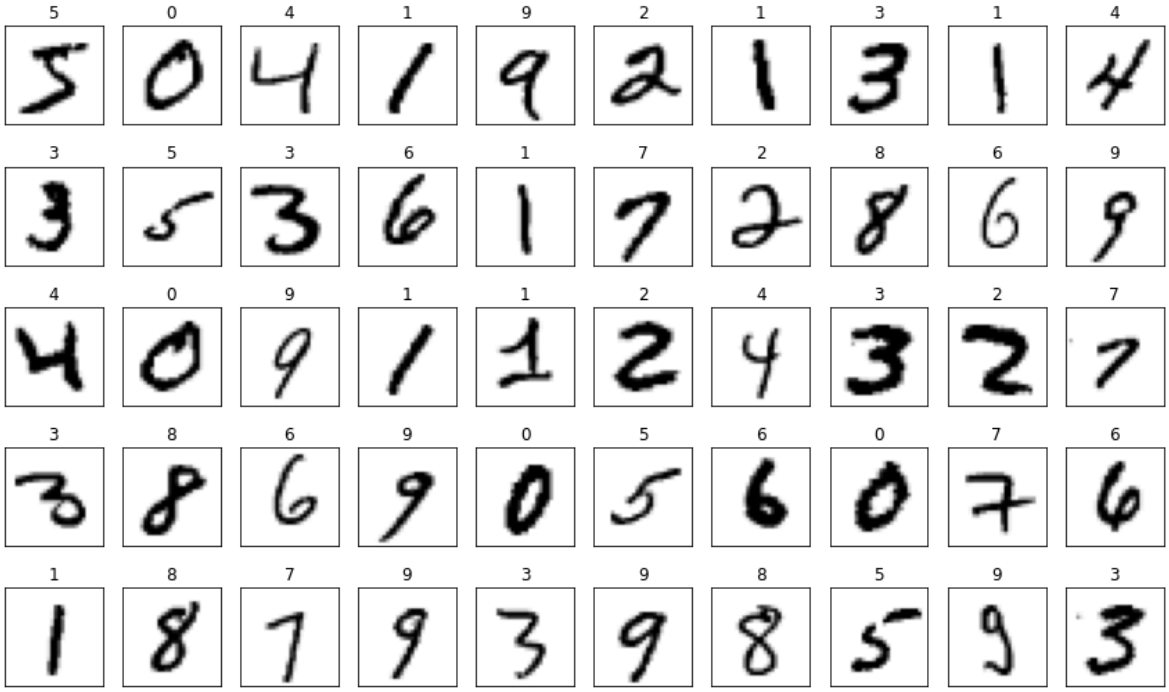

Figure 2. EDA: sample MNIST images with labels.

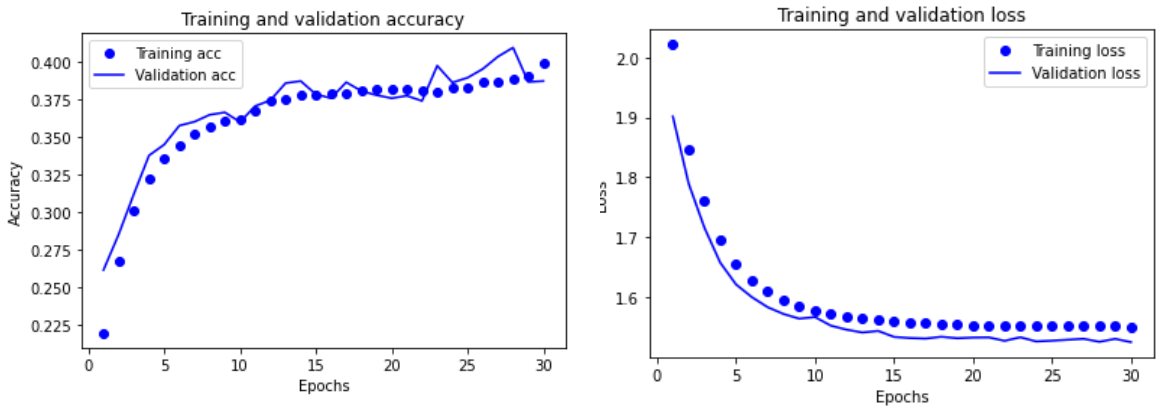

Figure 3. (left) Example model training and validation accuracy. (right) Example model training and validation loss.

Figure 4. (left) Sample of classified and misclassified 2's and 6's in Experiment 1. (right) Sample of classified and misclassified 7's and 9's in Experiment 2.

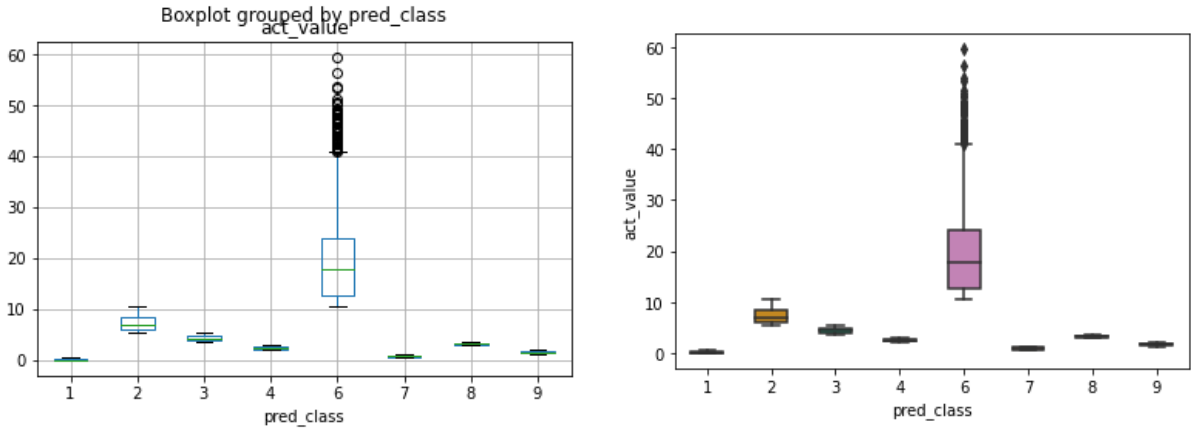

Figure 5. (left) Matplotlib boxplot of activation values in Experiment 1. (right) Seaborn version of said boxplot.

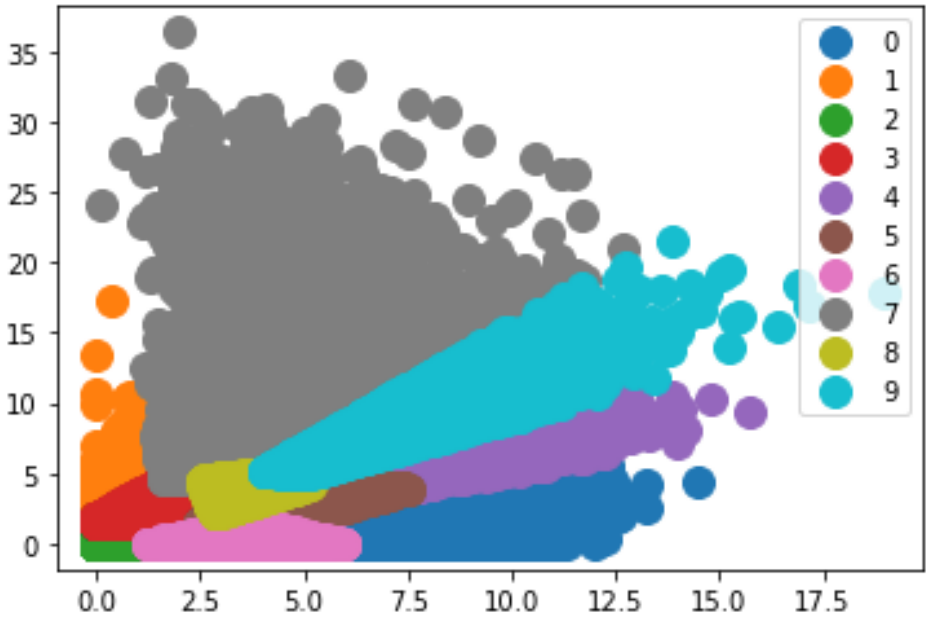

Figure 6. Scatterplot of activation values and class label boundaries in Experiment 2.

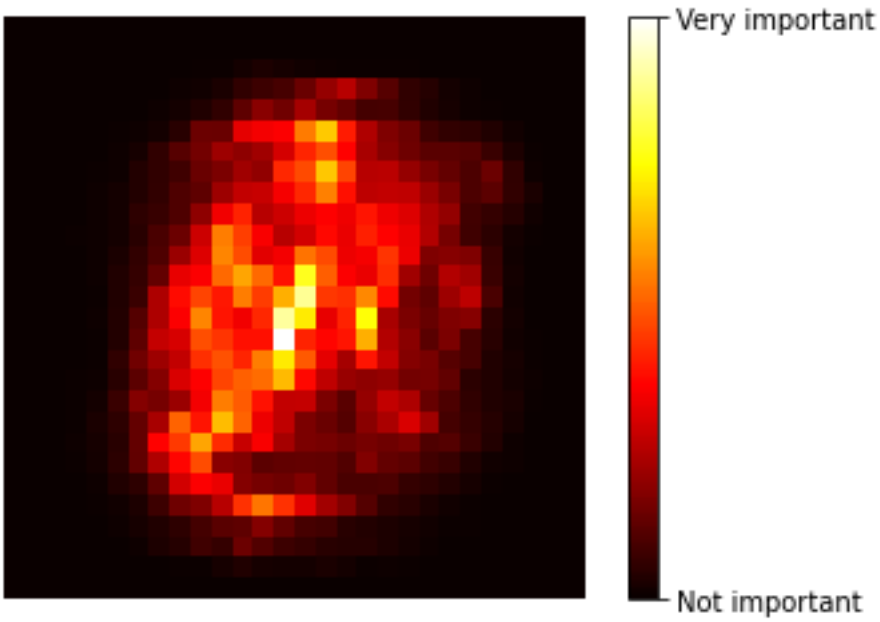

Figure 7. Heatmap of relative importance from RF classification of an image pixels in Experiment 5.

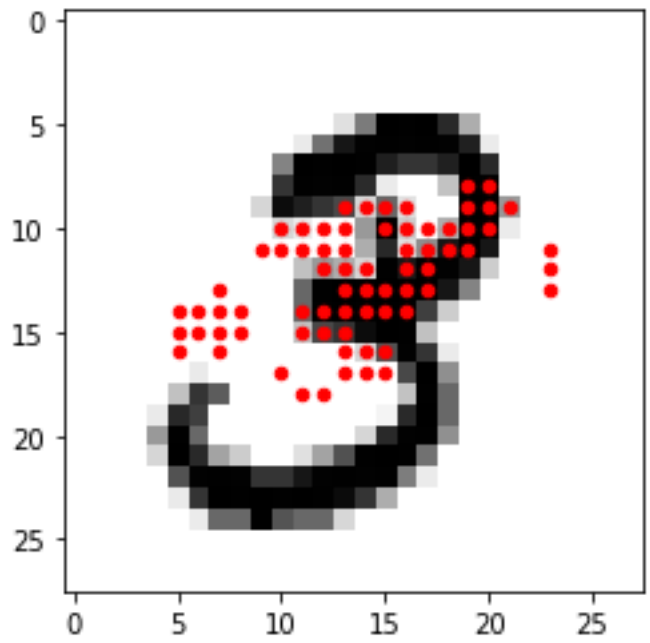

Figure 8. Highlight of the 70 most important pixels of a sample image in Experiment 5.

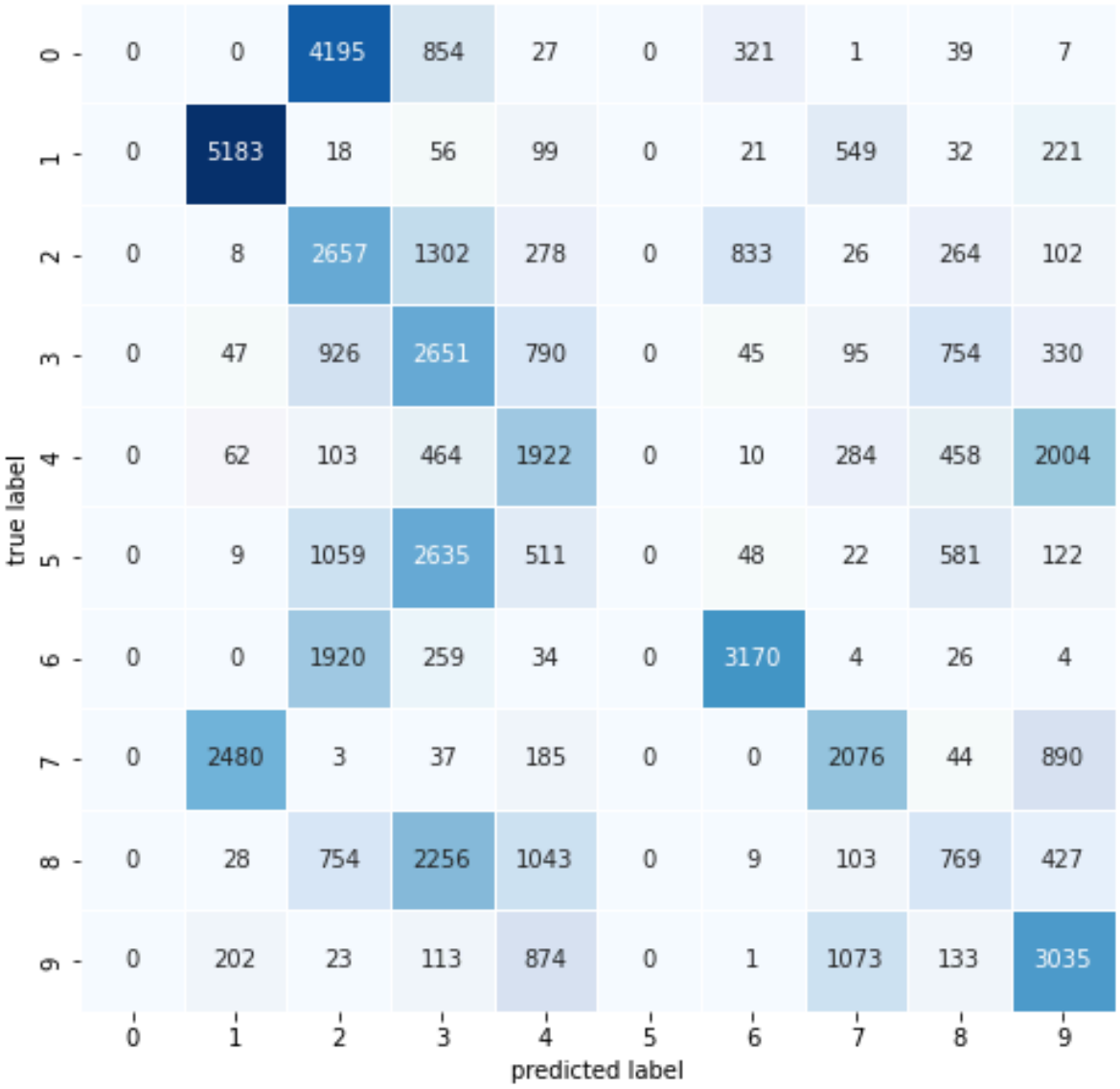

Figure 9. Confusion matrix of predicted v.s. true label of the original model in Experiment 1.

	Exp 1	**Exp 2**	**Exp 3 Var 1**	**Exp 3 Var 2**	**Exp 3 Var 3**	**Exp 3 Var 4**	**Exp 3 Var 5**	**Exp 3 Var 6**	**Exp 3 Var 7**
Hidden Nodes	1	2	20	40	60	80	100	100	100
Optimizer	RMSProp							SGD	Adam
Parameters	805	1600	15910	31810	47710	63610	79510		
Accuracy	39.68%	69.19%	95.67%	96.64%	97.00%	97.43%	97.73%	96.99%	97.71%
Loss	1.543	0.9671	0.1913	0.1795	0.1766	0.1690	0.1639	0.1009	0.1167
No. of Epochs to divergence	N/A	10	6	4	4	4	4	15	3

Table 1. Comparison of model attributes in Experiment 1-3.

	Exp 4	**Exp 5**
Input pre-processing	PCA decomposition	RF classification
Dimensionality	154	70
Hidden Nodes	100	100
Optimizer	Adam	Adam
Parameters	79510	79510
Accuracy	97.67%	93.96%

Loss	0.1439	0.2055
No. of Epochs to divergence	3	5

Table 2. Comparison of model attributes in Experiment 4-5.

References

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.

Chollet, F. (2021, December 21). *Deep Learning with Python, Second Edition* (2nd ed.). Manning.