(1) Abstract

The most common applications of deep neural networks are classification and regression. A standard hand-written digit dataset, taken from the Modified National Institute of Standards and Technology database and available through tensorflow, is included in the former and deep learning can be applied in a variety of methodologies and techniques to solve the problem. This paper will employ deep learning through tensorfow and keras and experiment with net structure, principal component analysis and feature reduction while observing the effects on prediction performance. The experimentation revealed the optimal number of nodes for use in a single hidden layer and that the features can be signicantly reduced (improving performance) with minor detriment to accuracy.

(2) Introduction

The Modified National Institute of Standards and Technology (MNIST) database (LeCun) is a foundational dataset used to demonstrate the capabilities of artificial neural nets and train engineers and data scientists in the available technologies and procedures. To gain understanding of all of the complex aspects of building and training neural nets, a simple and easily understandable problem is one of the best tools. The MNIST dataset has just 10 classes for prediction, one for each Arabic digit. The proposed problem is to train an artificial neural network (ANN) to process images of these digits, making a classification for each. The input layer of the network must be built according to the input data – which has been converted to an array the size of the image, twenty-eight by twenty-eight pixels, equaling a length of 784. This means that the input layer should have 784 nodes (though we'll reduce this in a later experiment). The output layer must have one node per class, hence ten. Only one hidden layer will be included using Rectified Linear Unit (ReLU) activation functions. We will then experiment on the hidden layer's number of nodes then the input layer's.

(3) Literature review

A comprehensive survey of MNIST studies was performed by Baldominos (2019). LeCun's early approach to handwritten character recognition using "graph transformer networks (GTN), allows such multimodule systems to be trained globally using gradient-based methods so as to minimize an overall performance measure" (1998). One of the best performing models is an ensemble of deep learning architectures, achieving an error rate of .18 percent (Kowarsi et al. 2018). The MNIST dataset is expounded upon by the Extended MNIST (EMNIST, Cohen et al.) increasing the complexity of the classification task by adding letters to the numbers.

(4) Methods

Review research design and modeling methods

The MNIST dataset is available to download through the TensorFlow python package. The provided data is split into 60,000 train and 10,000 test images paired with their labels (the actual digit from 0-9). Figure 6 shows how the "images" are actually encoded arrays of the 28x28 pixels with each pixel represented as a value between 0 (white) and 255 (black). To align with the 10 output nodes, the labels are converted to one-hot arrays. To align the 784 input pixels with the input nodes, they are flattened into a 1-D array then normalized to decimals between 0 and 1.

The model is then built using with structure previously discussed:

```
model = Sequential([
    Dense(input_shape=[784], units=1, activation =
      tf.nn.relu,kernel_regularizer=tf.keras.regularizers.L2(0.001)),
    Dense(name = "output_layer", units = 10, activation = tf.nn.softmax)
])
```

Beginning with a single node in the hidden layer: “units=1”. The model is compiled, trained on the 60,000 training images and tested for accuracy on the remaining 10,000. Experimentation on the number of nodes in the hidden layer is performed by altering the “unit” parameter. To simplify this experimentation, another python package Scikit-Learn includes a module called GridSearchCV we can provide a list of parameters we want to experiment on and it facilitates the experimentation and review of results.

With this being a fairly uneducated and brute-force approach, a more intelligent path should be investigated. One method available to us is Principal Component Analysis (PCA). PCA allows us to determine where most of the value in a large model is so we can seek to optimize. The ability to classify different subjects comes from finding their variance. We employ PCA to find which activate functions are finding the most variance in the model & eliminate the “less important” nodes in the hidden layer.

The remaining experimentation focuses on a different layer of the network, the input. With 784 input features, it’s highly likely that much of this is noise. A Random Forest model is not just highly flexible and easily generalizable, it can also provide feature importance, leading to dimensionality reduction. Choosing only the most important features can improve model efficiency.

(5) Results

Experiment 1 focused on using a single node in the hidden layer. The result is an activation function that is unable to differentiate between so many digits, as Figure 2 displays the overlap in classes’ activation values. It is able, however, to correctly classify (True Positives – see Figure 3) some digits such as 1 and 7 but still has many false positives on each and low accuracy across all other digits. Moving to Experiment 2, a comparison of Figure 3 to Figure 4 immediately visually displays the superior accuracy of the network structure using a hidden layer with two nodes. The result increased the accuracy on the test set from 36% to 68%. Figure 5 displays how increasing from one dimension (which

was shows in Figure 2) to two makes improved classification possible, where one node may be incorrect, the weight/bias going into the final layer would be adjusted for that specific class. The overlap in colors even across the two dimensions suggests that we need to continue to higher dimensions. Accuracy this high for a classification problem with 10 classes is fairly impressive, but with a simple activation function such as ReLU, it's likely that increasing the number of hidden nodes will continue to improve results.

In Experiment 3, the gridsearch of the hidden layer node count revealed how crucial it is to use a reasonable number of nodes relative to the interfacing layers. Using only one nodes resulted in less than fifty-percent accuracy and increasing to two doubled it but not enough for real world use. Increasing the number of nodes too high didn't result in diminishing returns but actually degraded model performance. It revealed that the optimal number of nodes in the hidden layer is 128 (Figure 6). The confusion matrix (Figure 7) for the "best" model shows nearly all images are classified correctly, improving of the scattered results of the previous two experiments and test accuracy achieving almost 97%.

Experiments 4 and 5 focus on dimensionality reduction in the second and first layers, respectively. PCA was employed through scikit-learn on the hidden layer activation functions. The "n_components" parameter was set to .95, which according to the documentation selects "the number of components such that the amount of variance that needs to be explained is greater than the percentage specified by n_components". This allows the hidden layer to reduce to 44 nodes while only losing .07% accuracy. A Random Forest was then "grown" to determine the relative importance of features (or pixels) in the input layer. The top 70 were chosen & used as inputs to a new network structure with corresponding 70 input nodes. The resulting test accuracy dropped to 92%, significantly more than Experiment 4. This makes sense though, as experiment 5 reduced the first layer nodes by 91% while Experiment 4 reduced the hidden layer node count by only 66%. Comparison of the Confusion Matrices reveals that the biggest drop in accuracy was due to an inability to properly classify images of the digit "9".

(6) Conclusions. So, what does it all mean?

Conversion of images to an array of grayscale values for use in training a neural net results in reliable and useful performance in recognizing and classifying digits. Experimentation in net structure and input dimensions provides tradeoffs in performance vs complexity and accuracy. Other research has shown that a simple single sequential network is limited in its ability to achieve greater accuracy and more complex net structures or ensembles would be required to improve further.

(7) References

1. "THE MNIST DATABASE of handwritten digits". Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond.

2. Baldominos, Alejandro, Yago Saez, and Pedro Isasi. 2019. "A Survey of Handwritten Character Recognition with MNIST and EMNIST" Applied Sciences 9, no. 15: 3169. https://doi.org/10.3390/app9153169

3. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

4. Kamran Kowsari, Mojtaba Heidarysafa, Donald E. Brown, Kiana Jafari Meimandi, and Laura E. Barnes. 2018. RMDL: Random Multimodel Deep Learning for Classification. In Proceedings of the 2nd International Conference on Information System and Data Mining (ICISDM '18). Association for Computing Machinery, New York, NY, USA, 19–28. DOI:https://doi.org/10.1145/3206098.3206111

5. G. Cohen, S. Afshar, J. Tapson and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2921-2926, doi: 10.1109/IJCNN.2017.7966217.

(8) Figures

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 167 208  19   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  13 235 254  99   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  74 254 234   4   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 154 254 145   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 224 254  92   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  51 245 211  13   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   2 169 254 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  27 254 254  88   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  72 255 241  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  88 254 153   0   0  33  53 155 156 102  15   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 130 254  31   0 128 235 254 254 254 254 186  10   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 190 254  51 178 254 246 213 111 109 186 254 145   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 192 254 229 254 216  90   0   0   0  57 254 234   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 235 254 254 247  85   0   0   0   0  32 254 234   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 235 254 254 118   0   0   0   0   0 107 254 201   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 235 255 254 102  12   0   0   0   8 188 248 119   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 207 254 254 238 107   0   0  39 175 254 148   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  84 254 248  74  11  32 115 238 254 176  11   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  21 214 254 254 254 254 254 254 132   6   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  14  96 176 254 254 214  48  12   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

Figure 1: Pixel Encoding Sample, “Six”

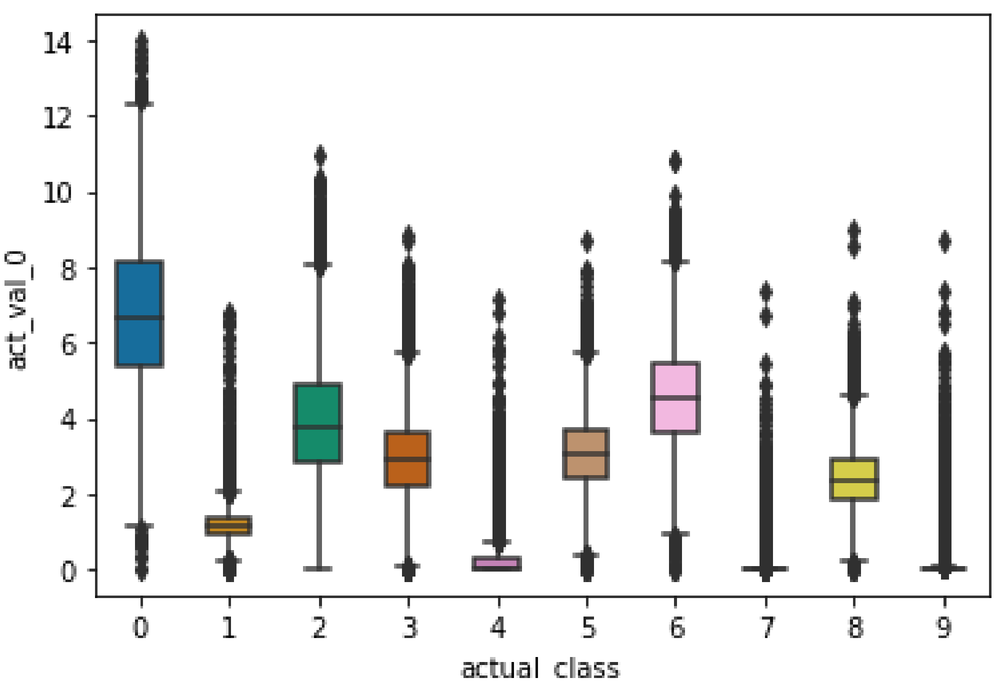

Figure 2: Experiment 1 - Single Hidden Node Activation Values

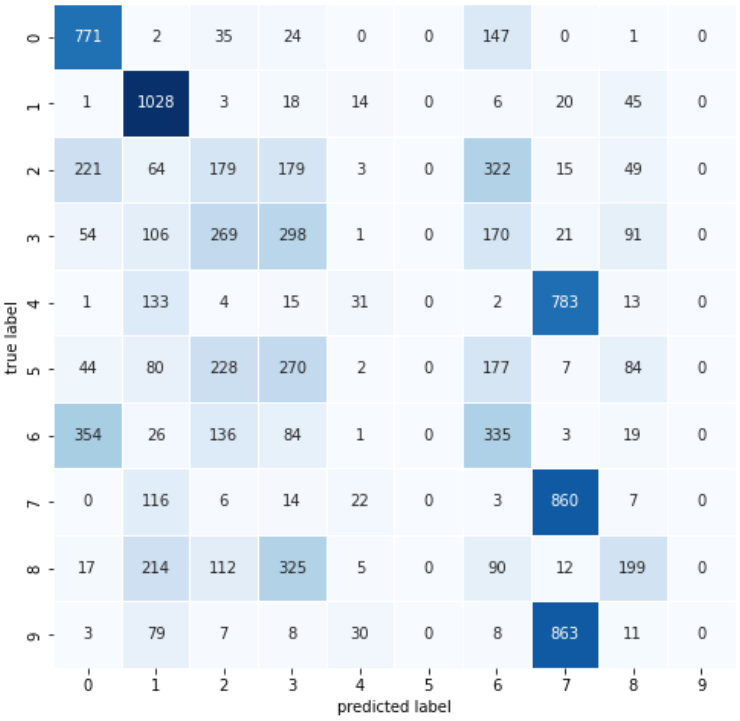

Figure 3: Experiment 1 - Single Hidden Node Confusion Matrix

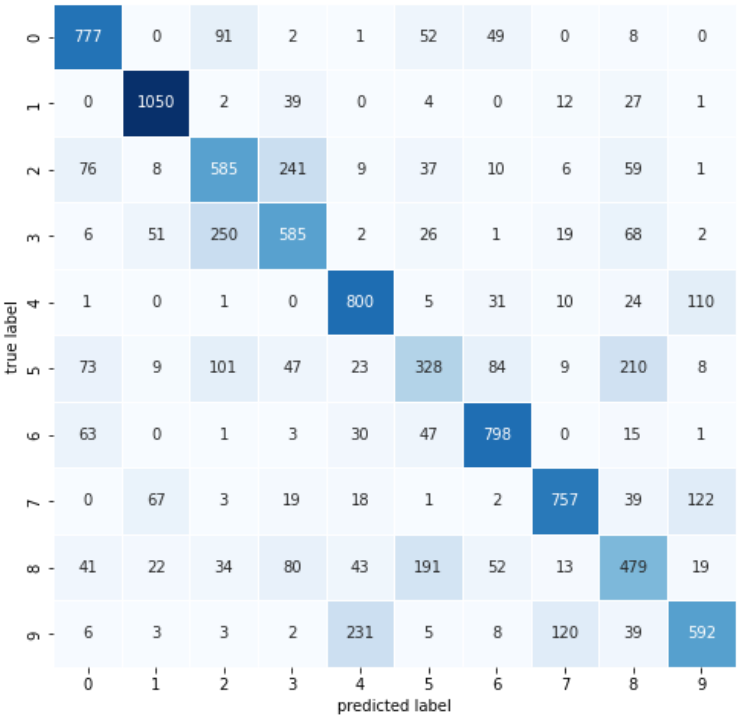

Figure 4: Experiment 2 - Two Hidden Node Confusion Matrix

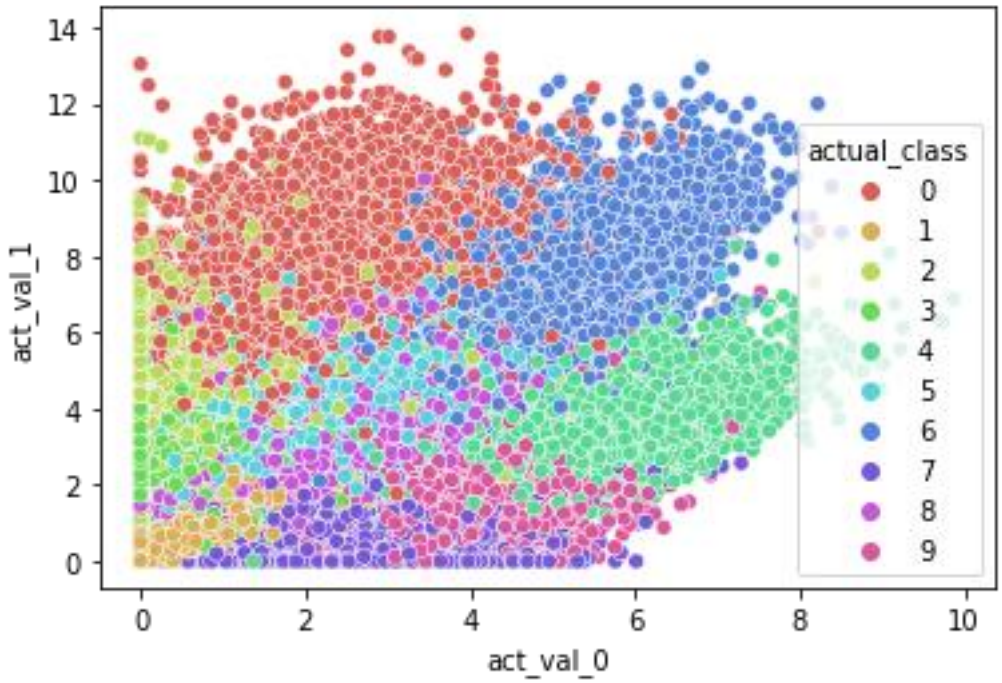

Figure 3:

Figure 5: Experiment 2 – Two Node Activation Functions with Color-Coded Classes

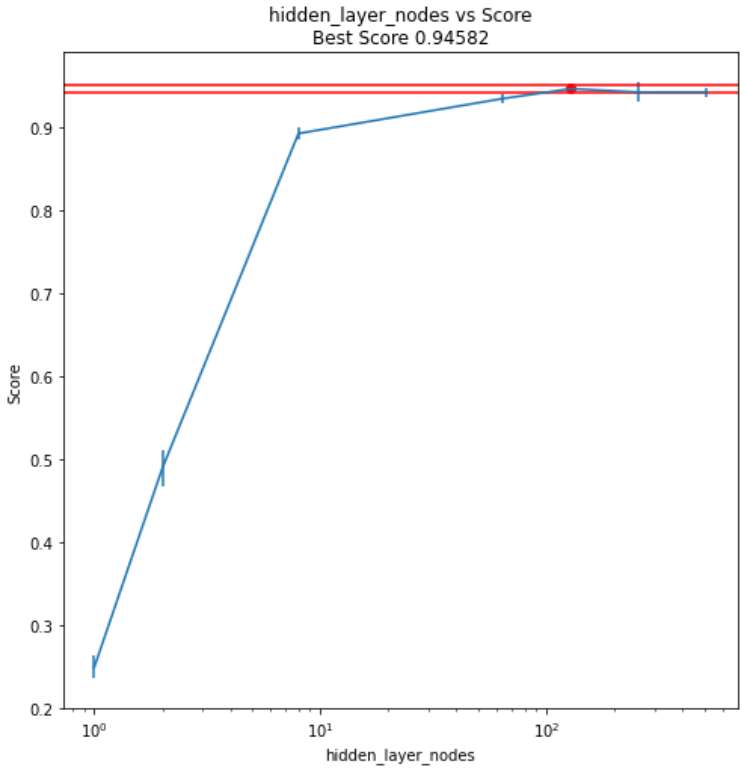

Figure 6: Hidden Layer Node Count Gridsearch Results by Accuracy

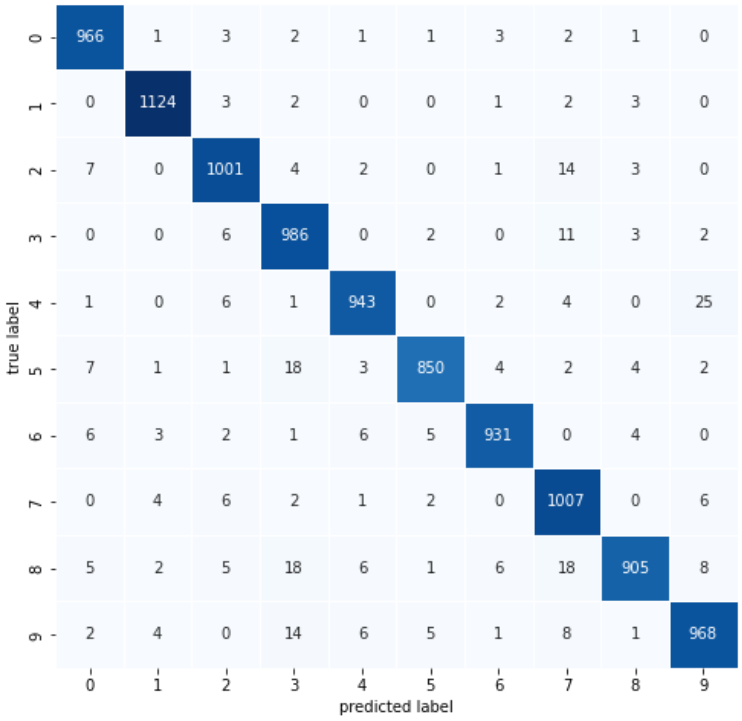

Figure 7: Experiment 3 – "Best" Model with 128 nodes in Hidden Layer

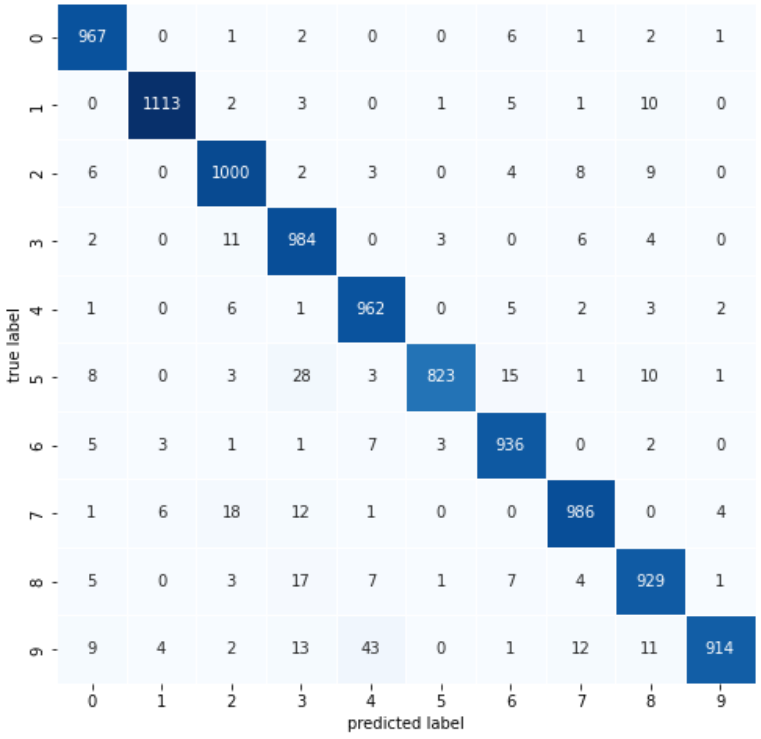

Figure 8: Experiment 4 – Principal Component Analysis Dimensionality Reduction Result

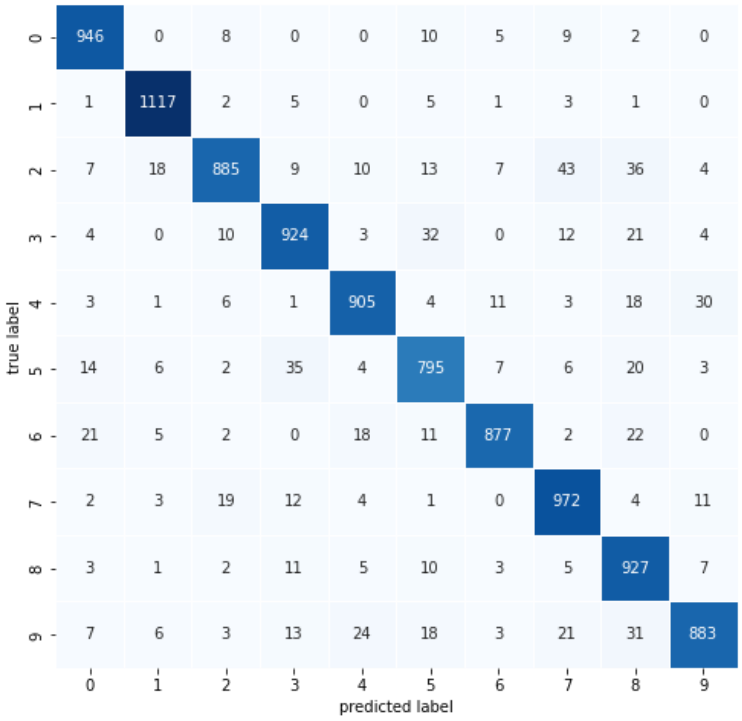

Figure 9: Experiment 5 – Random Forest Dimensionality Reduction Result