Digit Recognition Competition on Kaggle

[1] *En-Tien Lin* (林恩田), [1,*]*Chiou-Shann Fuh* (傅楸善)

[1]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
[*]E-mail: entien.lin@gmail.com fuh@csie.ntu.edu.tw

ABSTRACT

Application of deep learning neural network and computer vision model for numerical digit prediction on the Modified National Institute of Standards and Technology's published data. The MNIST dataset has been recognized as a classic computer vision dataset, which includes pre-extracted features, for neural network and deep learning techniques. Overall, the goal is to reach high accuracy in the Kaggle competition with different computer vision prediction techniques and illustrate the differences between each of the classifiers or methods.

Competition link: https://www.kaggle.com/c/digit-recognizer

1. BACKGROUND AND METHOD

Program the computer vision model with Python and Pytorch library. The model consists of classifier such as support vector machine, Convolutional Neural Network (CNN), Principal Component Analysis (PCA), *K*-Nearest Neighbors classifiers, single-layer and multi-layer perceptron techniques to predict the numerical digit datasets provided from the Modified National Institute of Standards and Technology (MNIST).

2. AIMS AND OBJECTIVES

Utilized deep learning neural network techniques and other computer vision machine learning skills, we wish to achieve a high accuracy with our results to claim a top rank in the competition.
While each of the method might be used in prior submissions and published notebooks, we would first apply each of them to figure out each method's performance, and then either research for new applicable methods or combine the listed skills to achieve a better performance in a newly created model.

3. DATA

The MNIST data provided in this competition consists of three datasets: training dataset, testing dataset, and a sample submission file. For the training dataset, there are forty-two-thousand 28 pixels by 28 pixels image along with the corresponding label of the training images. For the testing data, there are ten thousand 28 pixels by 28 pixels image for prediction. Lastly, for submission, the file consists of two columns, which are the image identification number and the predicted label for the corresponding testing image. The images' pixel-value is an integer in the range between 0 and 255; the value indicates the brightness or darkness of the pixel, with higher number means darker, and vice versa for the gray-scaled images.

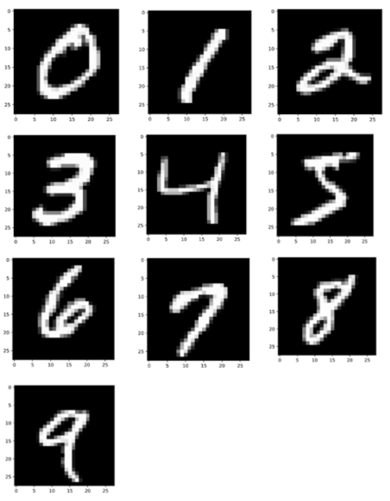


Fig. 1. The input images from digit 0 to digit 9 from MNIST.

4. METHODS AND RESULTS

4.1 Random Classifier

4.1.1 Method

Create a random classifier by setting the prediction based on a random pattern. Since there are 10 classes (from 0 to 9), the hypothesis is that the accuracy should be around 10 percent.

4.1.2 Result

By looking at the resulting confusion matrix (Fig. 11.), we can see that the probability is spread out, which means the prediction on each label diverges from all the classes. Furthermore, the accuracy of the prediction based on Kaggle submission is 0.1007, which closely matches our hypothesis 10 percent. Therefore, although the prediction accuracy of random classifier is extremely low, it is as expected, and its behavior matches the theoretical results.

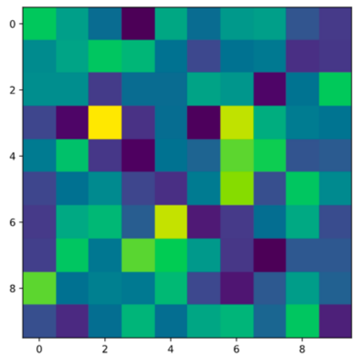


Fig. 2. The confusion matrix result of random classifier.

4.2 K-Nearest Neighbors (KNN)

4.2.1 Method

For KNN classifier, we utilized the existed library from scikit-learn python module for the implementation. In KNN classification, the prediction is simply classifying the image to the class that is most common amongst its k neighbors, which the distance metric is Euclidean in pixel space. In this experiment, k is set to 3, so images will be predicted to assign to the most common class within its 3 neighbors.

4.2.2 Result

Different from the random classifier's confusion matrix (Fig. 12.), the result of the KNN classifier shows a clear indication of high accuracy. More specifically, the confusion matrix has yellow value blocks on the diagonal line, which means the labels are predicted at a high accuracy. By looking at the raw values of the confusion matrix (Fig. 13.), we can find out that the values on the diagonal line are all close to 1, and the rest are either very small values or zeros. In other words, the diagonal values mean the probability of the actual label matches the predicted values, and the higher the accuracy is, the yellower block displays, and vice versa for the lower accuracy. Overall speaking, this KNN classifier has an accuracy of 97.05 percent, and intuitively, this result indicates that the classifier performs well just as how KNN is known for predicting labels. However, while we realize the high accuracy of KNN classifier, we also notice that it takes a significant amount of time to complete the prediction. By using the tdqm library during the computation, KNN classifier records an over-22 minutes runtime. Therefore, we came up with the idea of modifying the classifier to improve its efficiency but also maintain its high accuracy.

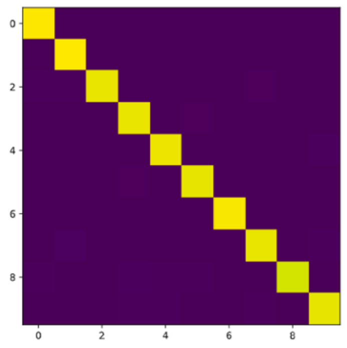
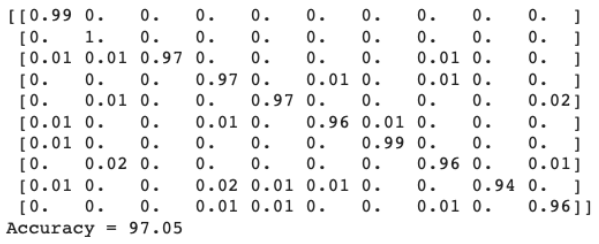


Fig. 3. The confusion matrix result of KNN classifier.



```
[[0.99 0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   1.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.01 0.01 0.97 0.   0.   0.   0.   0.01 0.   0.  ]
 [0.   0.   0.   0.97 0.   0.01 0.   0.01 0.   0.  ]
 [0.   0.01 0.   0.   0.97 0.   0.   0.   0.   0.02]
 [0.01 0.   0.   0.01 0.   0.96 0.01 0.   0.   0.  ]
 [0.01 0.   0.   0.   0.   0.   0.99 0.   0.   0.  ]
 [0.   0.02 0.   0.   0.   0.   0.   0.96 0.   0.01]
 [0.01 0.   0.   0.02 0.01 0.01 0.   0.   0.94 0.  ]
 [0.   0.   0.   0.01 0.01 0.   0.   0.01 0.   0.96]]
Accuracy = 97.05
```

Fig. 4. The values presented in the confusion matrix result of KNN classifier.

4.3 Principal Component Analysis (PCA) K-Nearest Neighbors (KNN)

4.3.1 Method

With the issue mentioned above, we decide to combine PCA technique with the already-tested KNN classifier. For this PCA-KNN classifier, we utilized the PCA and singular value decomposition (svd) implementations from the scikit-learn library. Furthermore, we continue to set the k value to 3 and principal component count to 25 for this mixed classifier. With the dimension change, the hypothesis is that the runtime should drop significantly while the accuracy should remain at a high level.

4.3.2 Result

Like what we expected, the PCA-KNN classifier is notably faster than the previous KNN classifier by a huge amount. From the tdqm result, the runtime is only 10 seconds for the entire process. Compare to the 22 minutes result, the model showed that we have accomplished our goal of minimizing the runtime. By further analysis on the difference during the computation, we understand that the remarkable speedup is achieved by reducing the dimension from 784 to the 25 we set. On the other side, the accuracy is measured at 97.31 percent, which is slightly higher than the previous result as well. Although the slight increase in runtime can differ for every trial, our second objective of maintaining the high accuracy is also completed. As a result, with the speedup and accuracy, this combined model is considered successful.

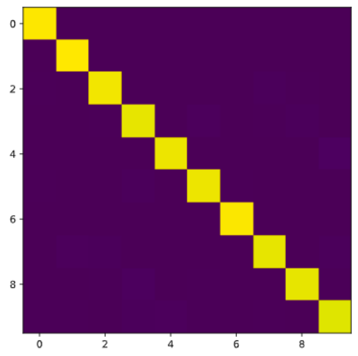


Fig. 5. The confusion matrix result of the PCA-KNN classifier.

4.4 Linear Classifier

4.4.1 Method

From this point on, we will implement deep learning neural network techniques on our prediction, and the first one will the simple linear classifier. The deep neural network method consists of initialize the gradient on data, forward propagation, compute error, back propagation, and update the parameters. Following the DNN training operations, the linear classifier simply predicts the label by connecting the input to the output. In addition, the linear classifier is also known as single layer perceptron, which the information simply feed-forward based on a transfer function. We can observe the output weight forward of the single layer perceptron for its feature detection.

4.4.2 Result

By looking at the confusion matrix result (Fig. 15.), we can see that the accuracy of linear classifier is not as high as KNN classifier. However, we can still notice the yellow blocks across the diagonal line, and the accuracy still manage to reach 92.46%. Besides, the single perceptron output weights (from Fig. 16. to Fig. 24.) present primary feature of each digit. More importantly, the neural learns how each digit looks like, so we can see the actual shape of the digit,

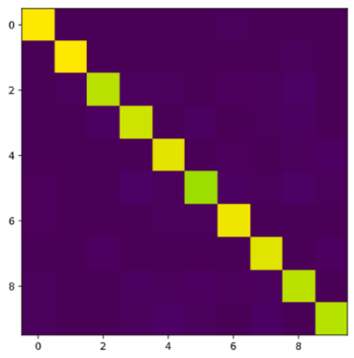


Fig. 6. The confusion matrix result of the linear classifier.

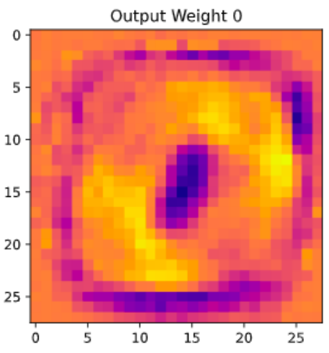


Fig. 7. The output weight of digit 0 of single layer perceptron.

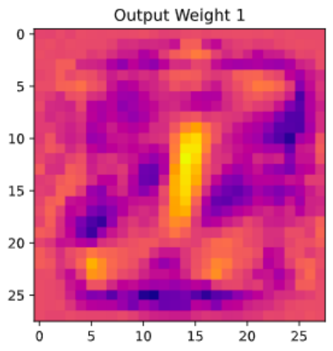


Fig. 8. The output weight of digit 1 of single layer perceptron.

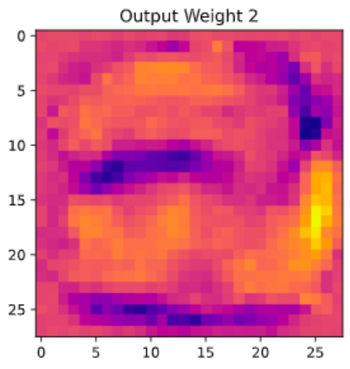


Fig. 9. The output weight of digit 2 of single layer perceptron.

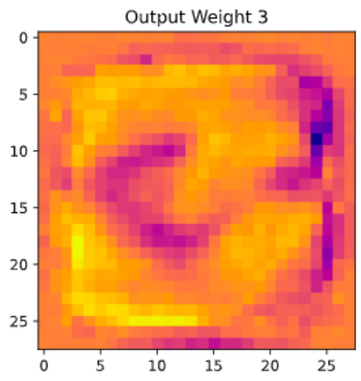


Fig. 10. The output weight of digit 3 of single layer perceptron.

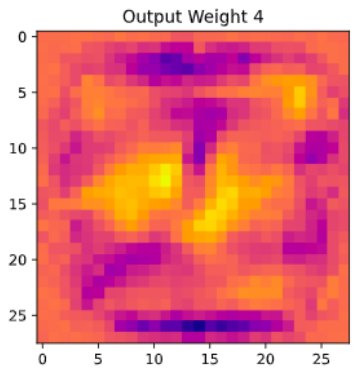


Fig. 11. The output weight of digit 4 of single layer perceptron.

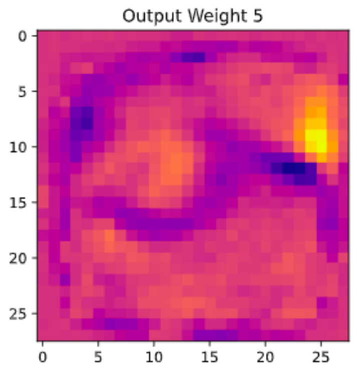


Fig. 12. The output weight of digit 5 of single layer perceptron.

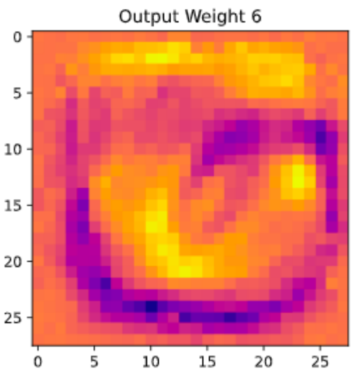


Fig. 13. The output weight of digit 6 of single layer perceptron.

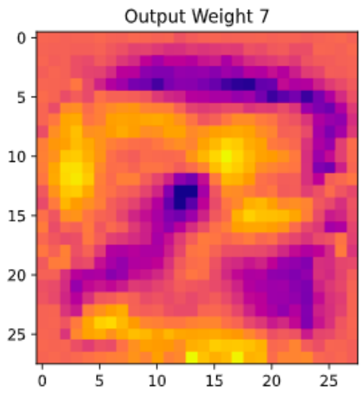


Fig. 14. The output weight of digit 7 of single layer perceptron.

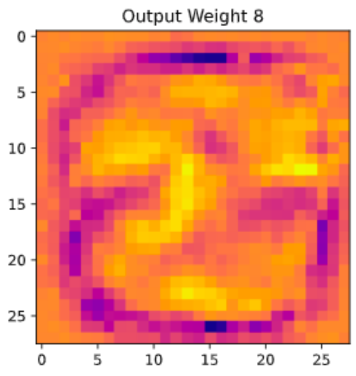


Fig. 15. The output weight of digit 8 of single layer perceptron.

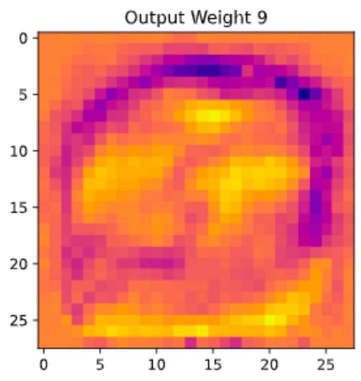


Fig. 16. The output weight of digit 9 of single layer perceptron.

4.5 Multi-Layer Perceptron (MLP)

4.5.1 Method

The MLP classifier is another deep neural network technique that consists of three layers, input layer, hidden layer, and finally, the output layer. In this model, the MLP classifier includes a matrix multiplication layer and bias offset layer, and we set the operation to have 10 epoch and a batch size of 50. Moreover, since the prediction is calculated through more layers from MLP then from linear classifier, the hypothesis is the MLP will reach a higher accuracy.

4.5.2 Result

As we expected, this MLP model results a 96.95% accuracy, which is higher than the linear classifier and strictly lower than the KNN classifier. Furthermore, as we look at the output weight filters of the MLP, we can see a huge difference from the display. Different from the single layer perceptron, the MLP weight outputs (from Fig. 25. To Fig. 35.) do not show the actual number, but partial feature the digit holds. We conclude these differences exists due to the multi layers it contains, which separate the features into pieces.

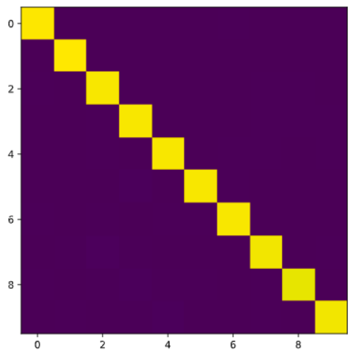


Fig. 17. The confusion matrix result of the MLP classifier.

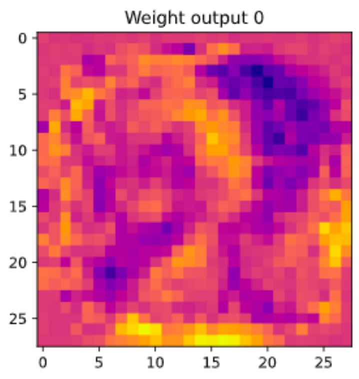


Fig. 18. The output weight of digit 0 of MLP classifier.

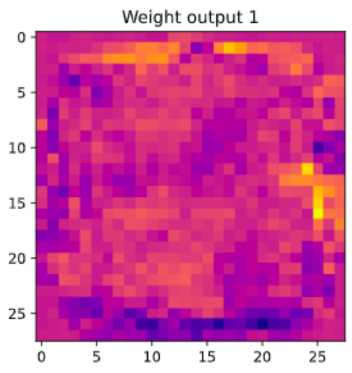


Fig. 19. The output weight of digit 1 of MLP classifier.

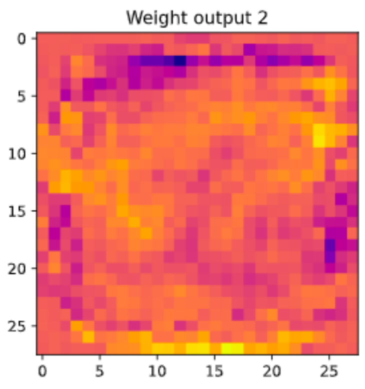


Fig. 20. The output weight of digit 2 of MLP classifier.

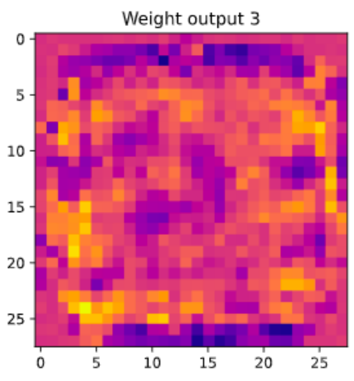


Fig. 21. The output weight of digit 0 of MLP classifier.

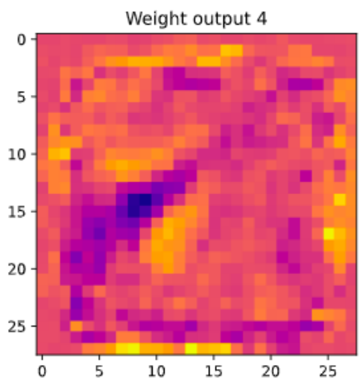


Fig. 22. The output weight of digit 4 of MLP classifier.

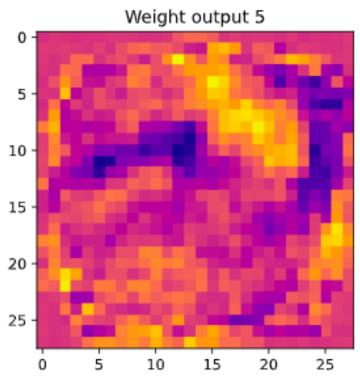


Fig. 23. The output weight of digit 5 of MLP classifier.

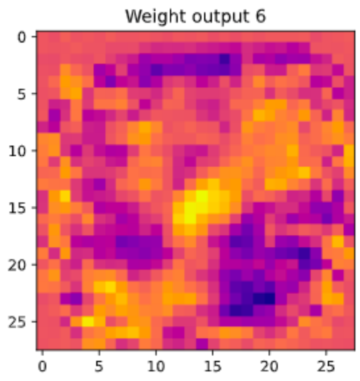


Fig. 24. The output weight of digit 6 of MLP classifier.

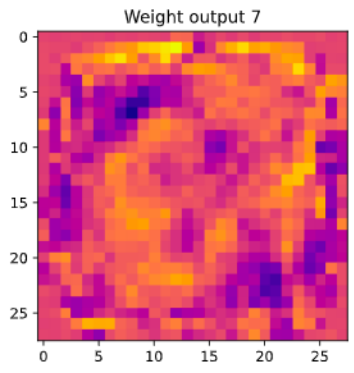


Fig. 25. The output weight of digit 7 of MLP classifier.

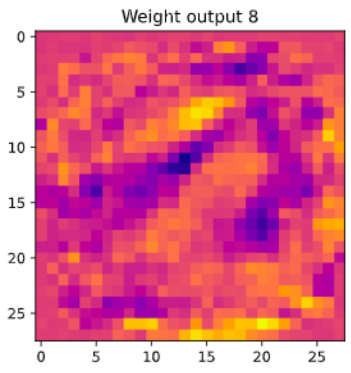


Fig. 26. The output weight of digit 8 of MLP classifier.

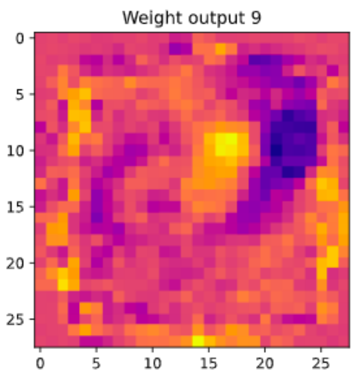


Fig. 27. The output weight of digit 9 of MLP classifier.

4.6 Convolutional Neural Network (CNN)

4.6.1 Method

CNN is a deep neural network technique that is most used in computer vision and analyzing visual imagery. Different from the MLP model, CNN model does not have full connectivity between layers to layers, which avoid the data from being overfitted. In the final version of our model, the CNN classifier consists of 2 convolutional layers, a fully connected hidden layer, and end with an output layer. Since CNN does not have the problem of overfitting, we hypothesize the result to be better than any other model we experienced.

4.6.2 Result

After all the revisions and modifications, our final CNN classifier results an accuracy of 99.425%. By looking at its confusion matrix result (Fig. 35.), we can notice a consistent bright yellow diagonal line, which indicates the high precision the model holds.

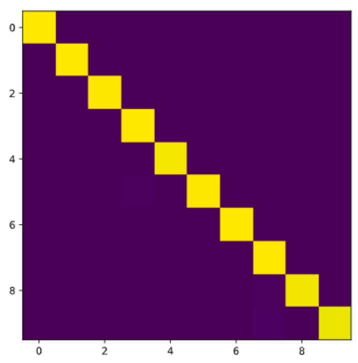


Fig. 28. The confusion matrix result of our CNN classifier.

5. CONCLUSION

5.1 Overall Discussion

As listed in Table 1, we can see that the CNN classifier holds the highest accuracy of all. While the top 120 predictions are 100% with non-provided dataset that includes testing labels, our prediction from CNN classifier still manage to rank 567 out of 5,837 participants (top 10%) in the digit recognition competition.

Table 1. Overall Performance.

	Accuracy
Random Classifier	10.07
KNN Classifier	97.05
PCA-KNN Classifier	97.31
Linear Classifier & Single Layer Perceptron	92.46
MLP Classifier	96.95
CNN Classifier	99.425

5.2 Future Works

Despite having a decent ranking in the competition, we still believe there are space for improvement. By looking at the top ranked public submissions, we noticed that other participants used leaf classifications, transfer learning, and other techniques that we did not implement. Furthermore, we should be able to achieve an even high accuracy by trying different combination of epochs and batch size in CNN or different k-value in KNN. Overall, this competition is a great learning experience, and we learned a great deal about image recognitions and computer vision deep learning techniques.

REFERENCES

[1] Kaggle, “Learn Computer Vision Fundamentals with the Famous MNIST Data,” https://www.kaggle.com/c/digit-recognizer, 2021.
[2] Kaggle, “Learn Computer Vision Fundamentals with the Famous MNIST Data,” https://www.kaggle.com/c/digit-recognizer/code, 2021.
[3] Dr. Benjamin Ochoa’s Computer Vision course at UCSD.