ML Algorithm Selection

Nijat Rustamov

Introduction

The problem I focused on in this work is predicting handwritten digits using several different Machine Learning algorithms. Specifically, I implemented Random Forest, Support Vector Machines, Multi-layer Perceptron and Convolutional Neural Networks. The dataset I used for this work was MNIST. There have been numerous studies [1, 2] on this dataset. Which was the main reason why I picked MNIST. It is easier to find studies and blog posts [3] that help to understand which models work the best and how to tune them for better performance. Dataset has been described as too simple for modern capabilities of ML. However, it is still considered useful for getting started.

Dataset Description

MNIST dataset contains 70,000 handwritten digit samples. The images are grayscale 28x28 in size. There are no missing values, and the labels are the digits. The dataset was split into three sets: train set with 49,000 samples, validation set with 7,000 samples and test set with 14,000 samples. In python the data can easily be loaded using "mnist" package. Looking at Fig.1 it is clear that the distribution of labels is almost uniform which is extremely helpful to avoid imbalanced learning.

Experimental Setup

The code was written in Python using scikit-learn, Tensorflow and Keras libraries. The choice of language and packages are subjective. I tend to use whatever I am already comfortable with. The data was normalized between minimum and maximum pixel values which are 0 and 255 respectively. Initially, Principial Component Analysis was

applied to reduce the dimensions of linearized images for certain ML models. However, I found that raw pixels result in better performance. I selected 4 classifiers as I mentioned in the Introduction section. Random forest and SVM were set up using scikit-learn and ML and CNN were set up using Tensorflow and Keras.

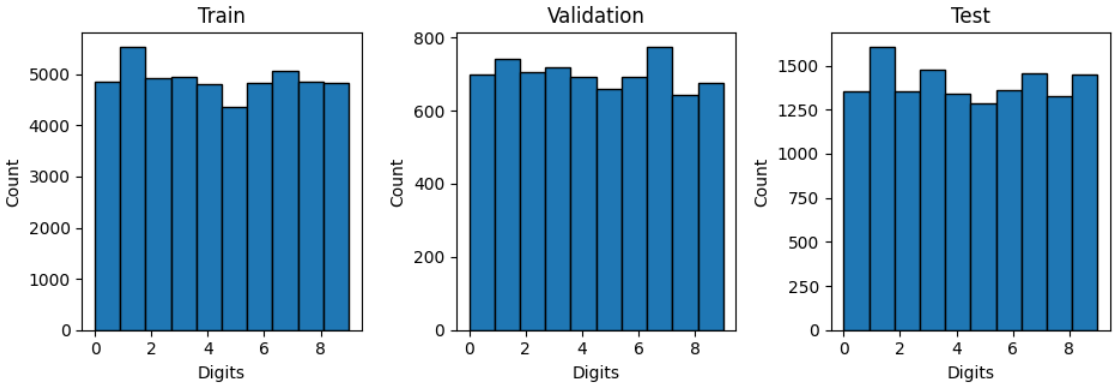

Figure 1. Distribution of class labels in different sets.

Results and Discussion

The summary of model training time and performance are provided in Tables 1 and 2 respectively. The fastest model to train was Random Forest and while the slowest model to train was MLP. However, Random Forest performed better than MLP. Random Forest demonstrated 96 and 97% accuracy on validation and test data and while MLP demonstrated 91-92% accuracy on validation and test data. Although the best performance came from CNN as expected with 98% accuracy, considering the runtimes and minor differences in performance between Random Forest and CNN, I would choose Random Forest over CNN for this task. SVM on the other hand, was neither the fastest nor the most accurate. In fact, MLP and SVM were very close to each other in terms of performance.

Fig. 2 shows confusion matrix on test data. Although there is seemingly nothing unusual, closer look at the plots reveal that every model is slightly confused between 3 and 5 and, 7 and 9.

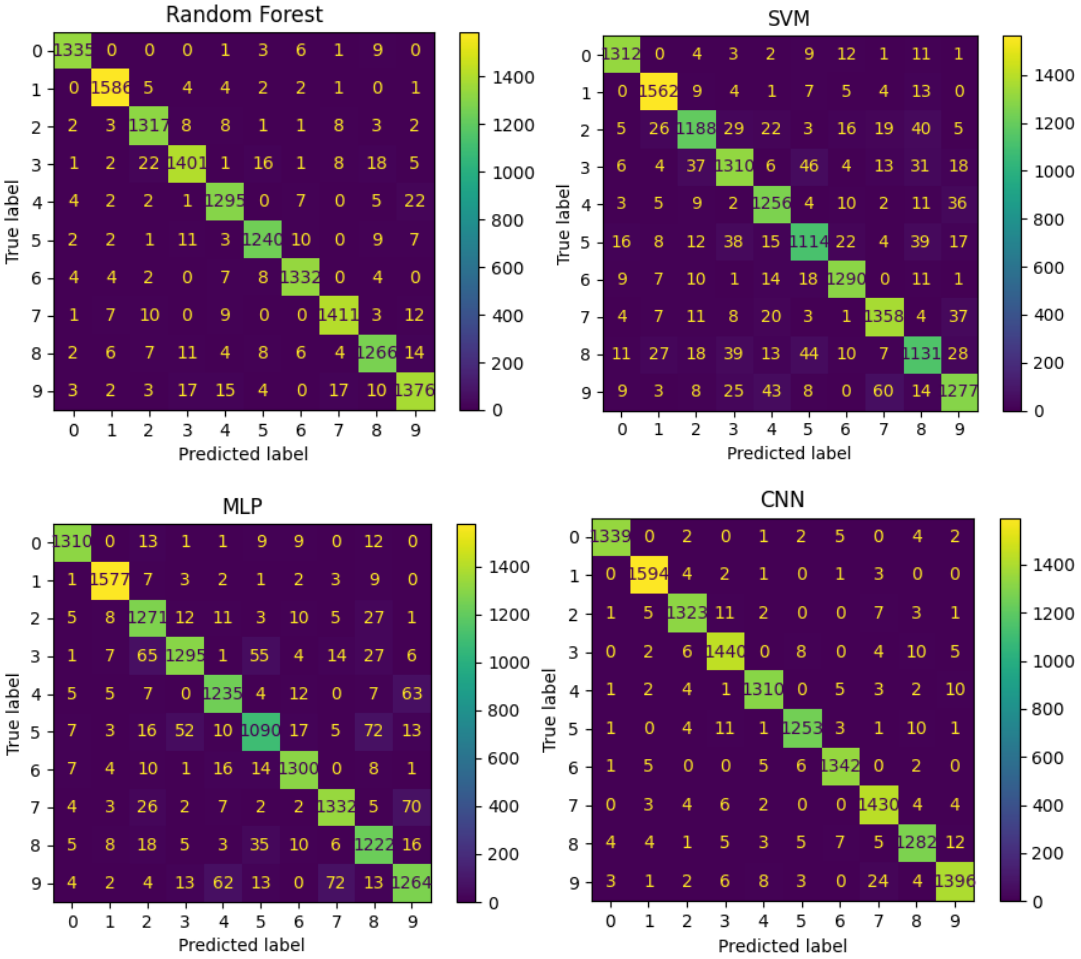

Figure 2. Confusion matrix of ML models

Table 1. Training time of ML models

Model	Training Time (seconds)
Random Forest	3
SVM	37.6
MLP	550
CNN	293

Table 2. Performance of ML models

Model	Train Accuracy	Validation Acc.	Test Accuracy
Random Forest	1.0	0.96	0.97
SVM	0.93	0.91	0.91
MLP	0.94	0.91	0.92
CNN	0.98	0.98	0.98

References

1. Kussul, E. and Baidyk, T. (2004) 'Improved method of handwritten digit recognition tested on MNIST database', *Image and Vision Computing*, 22(12), pp. 971–981. doi:10.1016/j.imavis.2004.03.008.
2. Baldominos, A., Saez, Y. and Isasi, P. (2019) 'A survey of handwritten character recognition with mnist and EMNIST', *Applied Sciences*, 9(15), p. 3169. doi:10.3390/app9153169.
3. Etzold, D. (2022) *MNIST-dataset of handwritten digits*, *Medium*. Available at: https://medium.com/mlearning-ai/mnist-dataset-of-handwritten-digits-f8cf28edafe (Accessed: 15 October 2023).
4. anujdutt9 (no date) *ANUJDUTT9/handwritten-digit-recognition-using-deep-learning: Handwritten digit recognition using machine learning and Deep Learning, GitHub*. Available at: https://github.com/anujdutt9/Handwritten-Digit-Recognition-using-Deep-Learning (Accessed: 18 October 2023).