Comparison and Analysis of Machine Learning, Deep Learning and Convolutional Neural Network on MNIST Data

Problem Definition

The purpose of this project is to apply and evaluate different traditional machine learning algorithms, neural network with multiple layers and convolutional neural network using MNIST Dataset. The problem this project addresses is to determine which model is most effective for this dataset considering the accuracy and other metrics score. This project also analyzes the computational resources and time taken for each model training. In addition, it also points how to increase the performance of the models.

Objectives

- To apply and evaluate different machine learning algorithms and combine the result of all through voting to improve the results.
- To implement and train deep neural network with multiple hidden layers.
- To analyze specialized deep learning algorithm named Convolutional Neural Network (CNN) for image data.
- To compare these models based on their training time, metrics score such as accuracy.

Data Description

MNIST is an image dataset loaded from Keras by default containing 60,000 train images and 10,000 test images. These images are in grayscale with pixel values between 0 and 255. The labels are digits between 0 and 9. In the preprocessing step, the values are normalized to be on the same scale and faster convergence. Additionally, data is further split to validation 10000 records from training data for parameter tuning. For machine learning and MLP (Multi layers perceptron) input shape compatibility, each image is flattened to 1D array.

Model Description

In this project mainly three algorithms from Machine learning are used. These include Decision Tree, Random Forest, KNN (K-nearest neighbors). For Deep learning, a multi-layer perceptron and convolutional neural network are used.

1). Machine Learning Models

To analyze the performance of machine learning models on image dataset, random forest, decision tree and KNN are applied.

1a) Decision Tree

Decision Tree splits the data based on the feature value to create the tree like structure for decision. It is used as baseline to see the performance of the model. It splits the data on the basis of Gini-index, entropy. The purer node is selected as head node. The Decision tree is fast to train and easy to implement but cannot capture the pixel relationship.

1b). Random Forest

Random forest combines multiple trees and then predict the output by voting from each tree. The reason to choose Random Forest is it works better on large and tabular data and produce better result by combining the output class prediction from all trees instead of a single tree. However, it is slower to train if number of trees are more and it also lacks spatial understanding of image data.

1c) K-Nearest Neighbors (KNN)

KNN classifies a sample based on the majority label of its k closest neighbors. It can perform well on clean and separated data but sensitive to noise.

In the end Votingclassifier of the three machine learning models is used to improve final result. This will collectively vote for the label and based on the majority, the output is predicted. Which leads to better performance compare to individual machine learning model.

2. Deep Learning (MLP)

The second algorithm used is a multi-layer deep neural network. For this project, two hidden layers with 256 and 128 nodes respectively are used. 1st hidden layer will capture edges and 2nd hidden layer will capture high level feature like shapes. Learning rate 0.001 is used as default for faster convergence and stability, the more increase in the learning rate will make the model to oscillate and overshoot while the lower value makes the model training slow. weight decay 0.0001 is used to regularize the weight to reduce overfitting, if higher value used, model can't learn and may underfit while for larger values, the model can't prevent overfitting. Batch size of 64 is used to balance training speed and model generalization. This size provides stable gradient updates. Additionally, we used Relu (rectifier linear unit) activation function which avoid vanishing gradient and fast training. In the end, to predict the class, we used SoftMax which will assign each class probability. Final layer has 10 nodes which is our class labels. Adam is used as optimizer for faster convergence, Spare categorical cross-entropy is the loss function suited for multi classification and does not require one-hot encoding. Early stopping and model checkpoints are used to see if the model is overfitting and did not improve, then it will save the best model weights. It can learn non-linear boundaries by hidden layers and activation functions. However, it also lacks spatial relationship with nearby pixels.

3). Convolutional Neural Network (CNN)

The final algorithm used is convolutional neural network. Learning rate with other batch size and weight decay are set similar to MLP. CNN captures the patterns in image using different convolutional filters. Here we used 128 filters of size 3 x 3 in first layer and 64 filters with 3 x 3 size. After convolution, max pooling is applied with 2 x 2 matrix and then flatten with one other hidden layer. It is also robust to translation, rotation as it captures the shape easily using filters. However, it requires more computational resources.

Experimental Environment

Platform and major libraries used in the project are given below.

Name	**Specification / Version**
Operating System	Window 10
Python	3.13.5
Tensorflow	2.20.0-rc0
scikit-learn	1.6.1
NumPy	2.1.3
Matplotlib	3.10.0
torchsummary	1.5.1
torch	2.8.0
torchvision	0.23.0
Hardware	Intel Core i5-6200 @ 2.30 GHz, RAM 12 GB, CUDA/GPU=0,

Table 1 Experimental Environment

Results and Analysis

The Decision Tree perform poorly because they treat each pixel as an independent feature, without considering the spatial context. This is the reason their performance is limited. However Random Forest has the advantage of multiple tree classification result and perform somehow better instead of individual tree. KNN also can calculate and assign the nearest pixel class which help in better result. Finally, The Voting Classifier showed that combining the strengths of different traditional models can improve performance.

Figure 1 Voting Classifier Metrics

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.97	1.00	0.99	1135
2	0.98	0.97	0.98	1032
3	0.97	0.97	0.97	1010
4	0.98	0.97	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.99	0.98	958
7	0.97	0.97	0.97	1028
8	0.98	0.95	0.97	974
9	0.96	0.97	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

From above Figure 1, we can see the model did well while predicting most of the digits. Suppose for "0" digit High Precision (0.97): the model predicts "0", it is correct 97% of the time. Very few false positives (other digits mistakenly predicted as 0). Recall (0.99): Out of all the true "1"s in the dataset, the model only catches 99%. some number of "0"s were missed (classified as another digit). F1-score is balance of precision and recall obtained by harmonic mean of precision and recall. It means the model did not confused with predicted 0 as other digit 98% time. However digit 9 is consfused with other digits reducing the f1-score to 0.96.

MLP is more accurate because its hidden layers can learn non-linear combinations of features, making it able to capture the underlying patterns in the data. However, it still processes the image as a flattened array, which means it loses the pixel relationship between adjacent pixels and also make the model parameter heavy. Like, a small shift in the digit's position can change the input vector leading to misclassification.

Figure 2 MLP Performance Metrics

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.98	0.98	0.98	1032
3	0.98	0.98	0.98	1010
4	0.98	0.98	0.98	982
5	0.98	0.98	0.98	892
6	0.99	0.98	0.98	958
7	0.98	0.98	0.98	1028
8	0.98	0.97	0.98	974
9	0.97	0.98	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

In **Figure 2**, we can see the MLP able to predict 0 correct of the times. Compare to Machine Learning models, it also did better on digit “9” where the f1 score is increased from 0.96 to 0.97.

The CNN's high performance is due to its use of convolutional layers, which apply filters to small sections of the image. This method allows the network to learn local features, such as edges and curves, and then combine them to recognize more complex patterns. This approach makes CNNs translation independent, where it can correctly classify a digit even if its position in the image is slightly shifted. The data augmentation technique further improves the CNN's ability to generalize by exposing it to a variety of rotated images during training, which makes it more robust to variations in image. Here is the for comparison of each model.

Figure 3 CNN Performance Metrics

	precision	recall	f1-score	support
0	1.00	1.00	1.00	980
1	0.99	1.00	1.00	1135
2	0.99	0.99	0.99	1032
3	0.99	1.00	0.99	1010
4	0.98	0.99	0.99	982
5	0.99	0.98	0.99	892
6	0.99	0.99	0.99	958
7	0.99	0.98	0.99	1028
8	0.99	0.99	0.99	974
9	0.99	0.98	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

In above **Figure 3**, the CNN perform best of all, compare two previous accuracy and other scores are better. Even the digit “9” which was misclassified by other is correctly predicted 99% time. This is because it can capture the shape by using different filters.

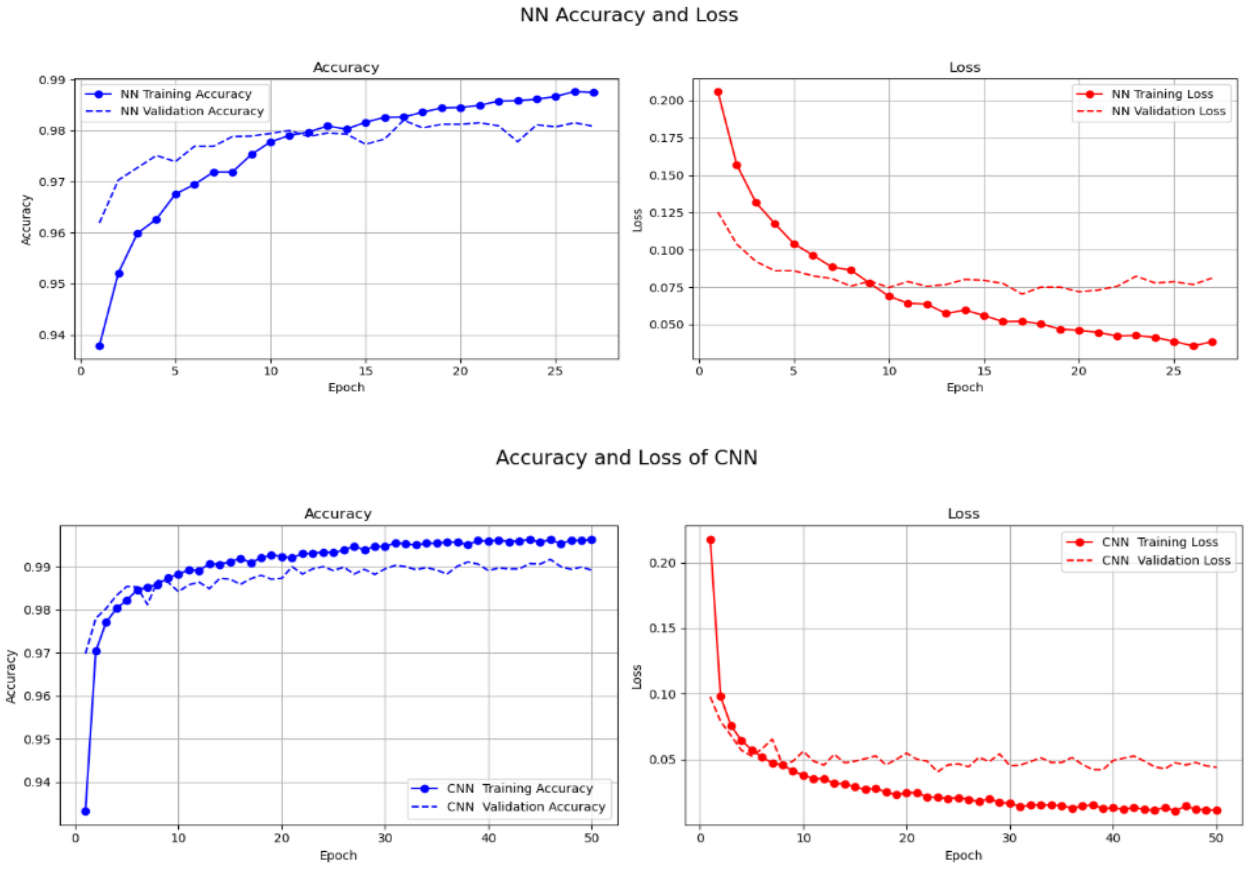

Figure 4 NN and CNN performance by epoch

From the above Figure 4, for NN, we can see that validation accuracy stopped to increase after 17 epochs. Similarly, the validation loss also stops to decrease and started again to increase after 17 epochs. On the other hand, CNN got higher accuracy in the initial epochs and steadily increasing over the epochs.

Hence, from misclassification patterns, the MLP was more likely to confuse digits that are visually similar and stroke is elongated, such as a 2 and a 3 as shown in below **Figure 5**. Due to bottom stroke elongation, NN unable to predict it 2, however CNN did due to various filters.

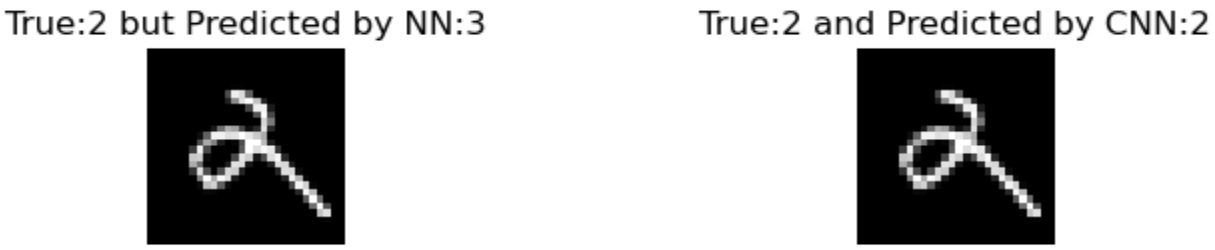

Figure 5 NN vs CNN Prediction

The overall accuracy score is shown in below **Table 2**. We can see from the above results that CNN performed well on image. However, the resources consumption and training time for CNN is also very high. Considering the resources provided in **Table 1**, here is the summarized overview of the major three models.

Table 2 Models Summary (TensorFlow)

	Model	#Params	Training Time (min)	Test Acc %	Macro-F1	avg time per epoch (min)
0	ML	N/A	1.06	97.42	0.97	N/A
1	MLP	235914	3.39	98.12	0.98	0.13
2	CNN	1219614	383.85	99.07	0.99	7.68

Similarly, for MLP and CNN, Pytorch is used and the performance is almost similar except the time taken per epoch which is much than tensorflow.

Table 3 Models Summary (Pytorch)

	Model	#Params	Training Time (min)	Test Acc %	Macro-F1	avg time per epoch (min)
0	MLP	235914	6.55	98.06	0.98	0.16
1	CNN	1632286	367.60	99.15	0.99	11.14

Conclusion and Future work

This project compared the three algorithms on MNIST Image dataset where CNN achieved the highest accuracy 99.07% followed by Multi-Layer Perceptron MLP 98.12% while machine learning voted classifier obtained 97.42% considering tensorflow implementation. MLP are less effective and more parameter heavy for image data but still performed better due to gradient descent optimization method. In the end, CNN takes the highest accuracy 99.07% as it is most appropriate for image data due to considering the neighboring pixel and also less parametric. Further, even if the pixel shifts slightly, the CNN captures those patterns inside the image, However, in case of MLP, the output can be different as it can be affected by pixel shift unlike CNN.

We trained the model with limited number of layers and without using pretrained models. However, in future, the results can be further improved by using transfer learning – pre trained model, data augmentation including translation, zooming and rotation of image. Further the number of hidden layers can be increased to capture complex shapes in the image.