Analysis and Report of the implementation of CNN on MNIST dataset:

Github link - sainathvaddi/CNN_on_MNIST_data (github.com)

Understanding Dataset: The MNIST dataset is a widely used dataset for handwritten digit recognition. It consists of 28x28 grayscale images of handwritten digits (0 through 9). Each image is labeled with a digit.

Data Preparation:

Loading the Dataset: We used the provided method to fetch the MNIST dataset from the UCI repository.

Data Preprocessing: Normalization and reshaping are the preprocessing steps. Normalization provides that the pixel values are within the range between 0 and 1, which helps in faster convergence during training. Reshaping (to 8*8*1) is required to match the input shape expected by CNN.

CNN Architecture:

Convolutional Layers: These layers extract features from the input images. Each of these layers applies a set of kernels (filters) to the input image, resulting in feature maps. ReLU activation functions are applied after the convolution to introduce non-linearity.

Max Pooling: These layers simplify the feature maps by reducing their size. It's like zooming out, focusing only on the most important details only. This technique keeps the highest value in each region, which helps in capturing the most relevant features while making calculations easier.

Fully Connected Layers: After multiple convolutional and pooling layers, fully connected layers are added to perform classification based on the extracted features. The final fully connected layer is followed by a softmax activation function to produce probabilities of each class.

Model Architecture:

The CNN architecture here consists of 3 convolutional layers with ReLU activation functions. Each layer's parameters and dimensions need to be documented thoroughly.

This includes the size of the filters, the number of filters, padding, stride, and the dimensions of the feature maps at each layer. The architecture is as follows:

Convolutional Layer 1: In this layer, Parameters are 32 filters, kernel size (3, 3), Input Dimension: (8, 8, 1), Output Dimension: (8, 8, 32). After this, add zero padding (1, 1) to increase spatial dimensions.

Max Pooling Layer 1: This layer has Pool Size: (2, 2), Strides: (2, 2), Input dimension: (8, 8, 32), Output Dimension: (4, 4, 32)

Convolutional Layer 2: In this layer, Parameters are 64 filters, kernel size (3, 3), Input dimension: (4, 4, 32), output dimension: (4, 4, 64)

Max Pooling Layer 2: This layer has Pool Size: (2, 2), Strides: (2, 2), Input dimension: (4, 4, 64), output dimension: (2, 2, 64)

Convolutional Layer 3: In this layer, Parameters are 128 filters, kernel size (3, 3), Input Dimension: (2, 2, 64), Output Dimension: (2, 2, 128)

Layers Flatten: Flattening the output of the last convolutional layer, the input dimension is (2, 2, 128), and we get output dimension as (512,)

Fully Connected Layer: This layer is with 64 neurons and ReLU activation, with input dimension of (512,), and gives output dimension of (64,)

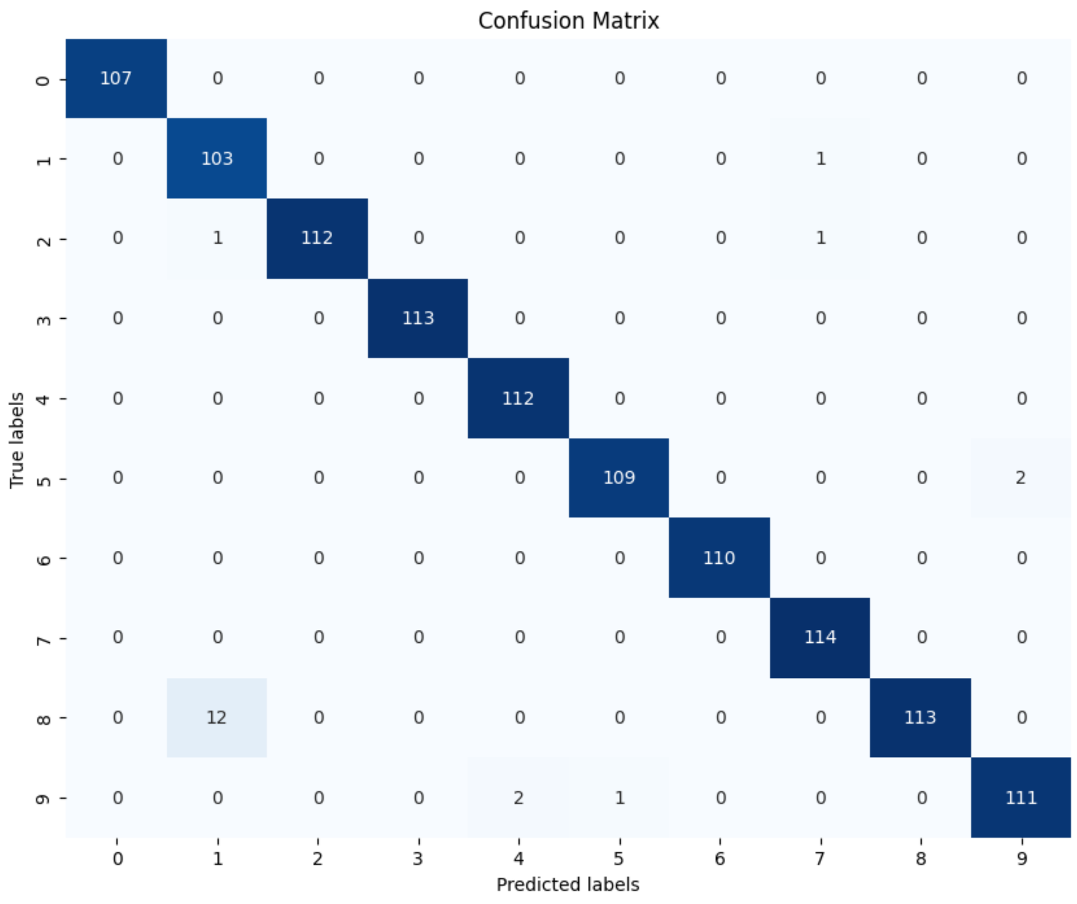
Output Layer: This layer is with 10 neurons for classification and softmax activation, with input dimension of (64,), output dimension of (10,)

Training and Evaluation: K-fold cross-validation with k=5 was performed to train and validate the model. The model was trained for 10 epochs with a batch size of 32. Performance metrics such as validation accuracy and loss were monitored across epochs. The overall accuracy of the model was calculated using the test set. Additionally, a classification report was generated to evaluate precision, recall, and F1-score for each class.

Results and Analysis: The average validation accuracy across folds was approximately 98.5%. Validation accuracy and loss were plotted across epochs to visualize the model's training process. The overall accuracy of the model on the test set is 98.49%.
The classification report revealed high precision, recall, and F1-score for each digit class, indicating robust performance across all classes. The classification report and Confusion matrix are provided below:

Classification Report

Class (digits)	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	107
1	0.89	0.99	0.94	104
2	1.00	0.98	0.99	114
3	1.00	1.00	1.00	113
4	0.98	1.00	0.99	112
5	0.99	0.98	0.99	111
6	1.00	1.00	1.00	110
7	0.98	1.00	0.99	114
8	1.00	0.90	0.95	125
9	0.98	0.97	0.98	114



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	107	0	0	0	0	0	0	0	0	0
1	0	103	0	0	0	0	0	1	0	0
2	0	1	112	0	0	0	0	1	0	0
3	0	0	0	113	0	0	0	0	0	0
4	0	0	0	0	112	0	0	0	0	0
5	0	0	0	0	0	109	0	0	0	2
6	0	0	0	0	0	0	110	0	0	0
7	0	0	0	0	0	0	0	114	0	0
8	0	12	0	0	0	0	0	0	113	0
9	0	0	0	0	2	1	0	0	0	111