MLNS Deep Learning Assignment Part -2 Report

- 2021101046

This project involves building a robust pipeline for predict sum of the digits given an image and its label value. The pipeline comprises two key modules:

1) Digit Extractor Module: Identifies and processes individual digits from input images.
2) Neural Network Classifier: Classifies the processed digits using a CNN model trained on the MNIST dataset.

Digit Extractor Module:

The digit extractor module employs contour-finding methods to isolate and extract individual digits from input images.

- Zero-padding was applied to images to ensure digits do not touch the borders. This step avoids boundary-related issues during contour analysis.
- Used contour finding to get boundaries of the digits. This initially led to 2 or more digits being returned. In order to handle this we check if the aspect ratio of the extracted digit matched the expected ratio. If not, the boundary was divided by a centerline, presuming that it had originally concatenated 2 digits together. For three connected digits, the boundary was divided into three equal parts.
- Applied a bilateral filter to the extracted digit images to reduce noise while preserving edges. Parameters: `7` for the diameter and `75*75` for intensity-related smoothing.
- Each extracted digit was resized to `28x28` pixels using a custom-defined `resize_to_fit()` function to match the input image size learned by the MNIST-trained neural network.

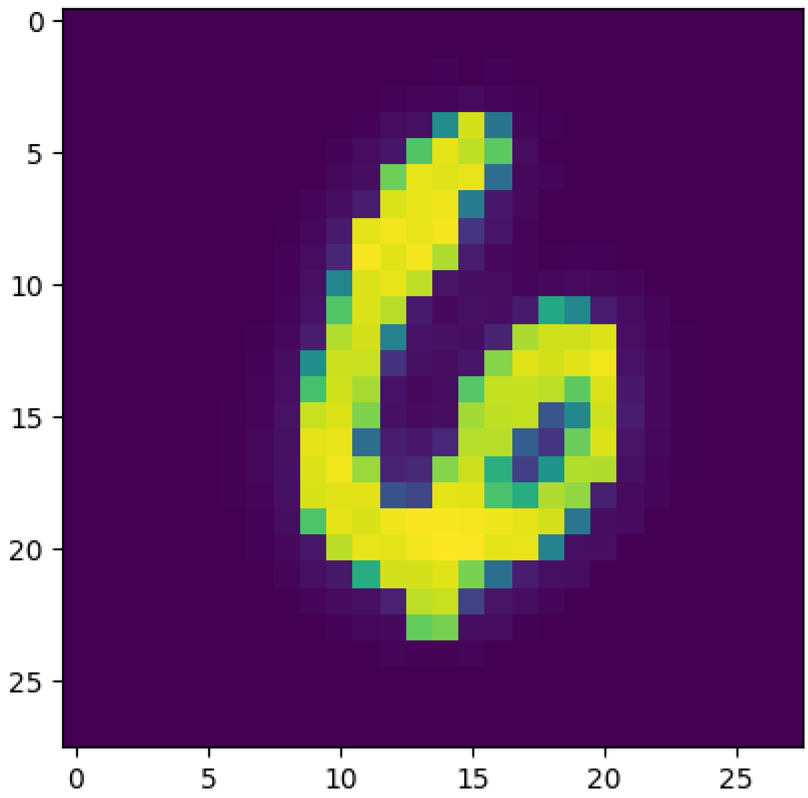
Neural Net Trained on MNIST Dataset:

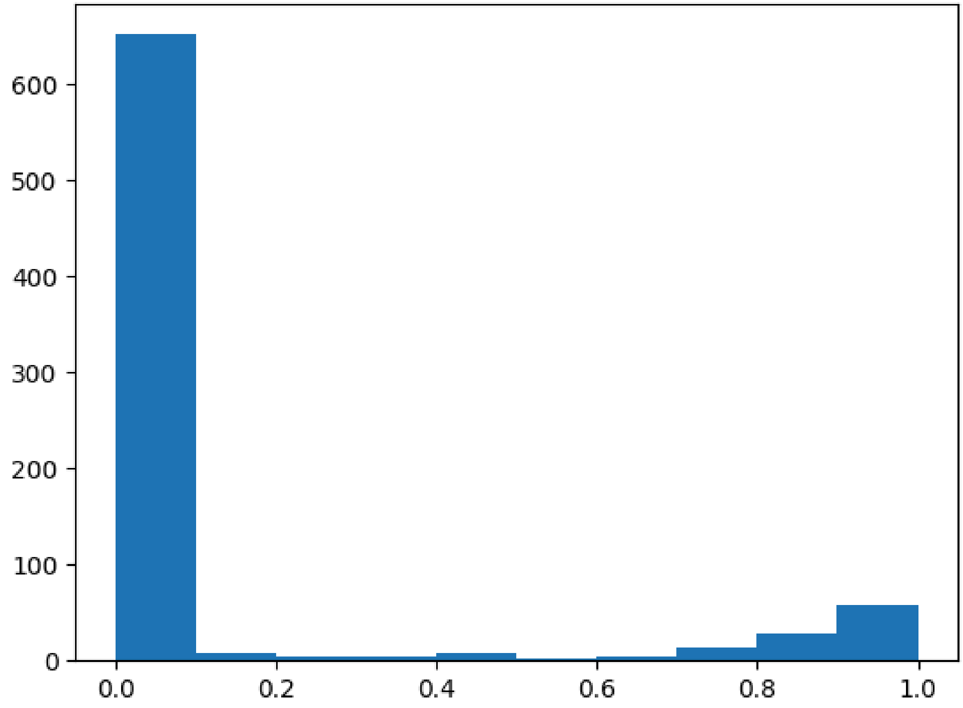
A CNN model was trained on the MNIST dataset, achieving 99% accuracy.

Model Architecture:

- Input shape: `(28, 28, 1)` for grayscale images.
- Two convolutional layers with ReLU activation and He initialization.
- MaxPooling layers to reduce dimensions.
- Dense layers with softmax activation for 10-class classification.

A discrepancy was observed in the pixel distribution intensity values for the MNIST dataset and the images extracted by the digit extractor module.



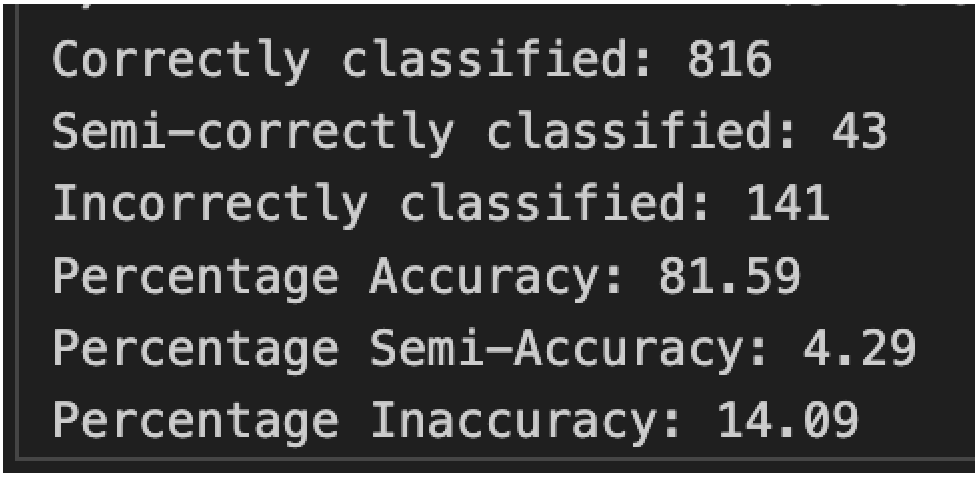


The extracted images lacked a sharp rise in frequency near 1.0 intensity (seen in MNIST images). Bilateral noise filtering was used to make the extracted images resemble the MNIST dataset.

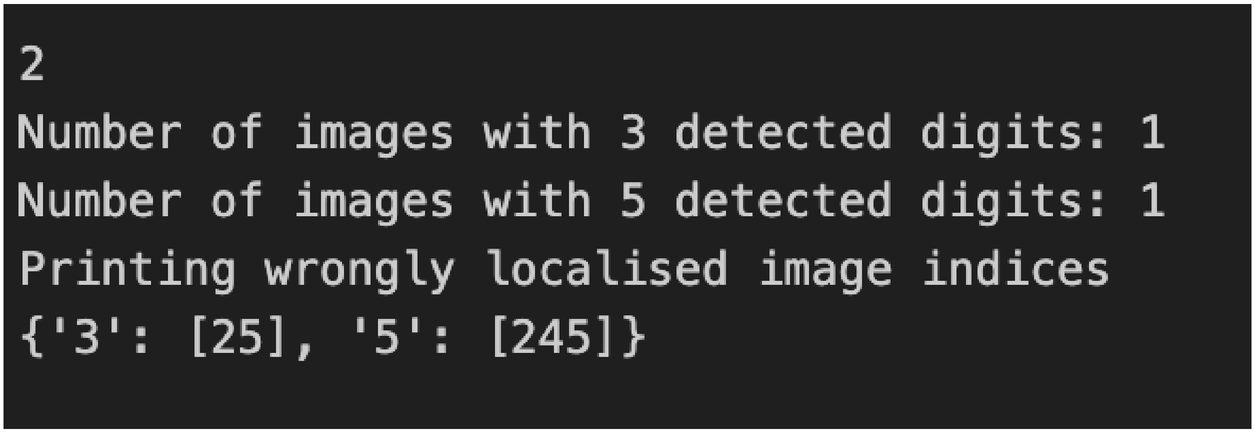
Pipeline Summary:

Image Processing: Contours were detected, digits were extracted, and preprocessed using bilateral noise filtering, resizing, and normalization.
Prediction: Processed digits were fed into the MNIST-trained CNN for classification. The predicted digits for the 4 extracted images were added and the sum thus calculated is reported as the final label for each image.
Performance: Achieved 81.59% accuracy, with 816 images correctly classified.
Error Analysis: Images with 3 or 5 digits detected due to incorrect contour splitting/segmentation or poor image quality leading to incorrect predictions were logged.

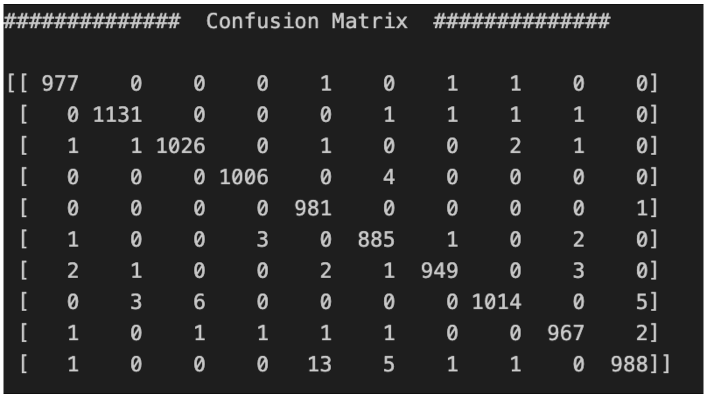
Metrics Report:



```
Correctly classified: 816
Semi-correctly classified: 43
Incorrectly classified: 141
Percentage Accuracy: 81.59
Percentage Semi-Accuracy: 4.29
Percentage Inaccuracy: 14.09
```

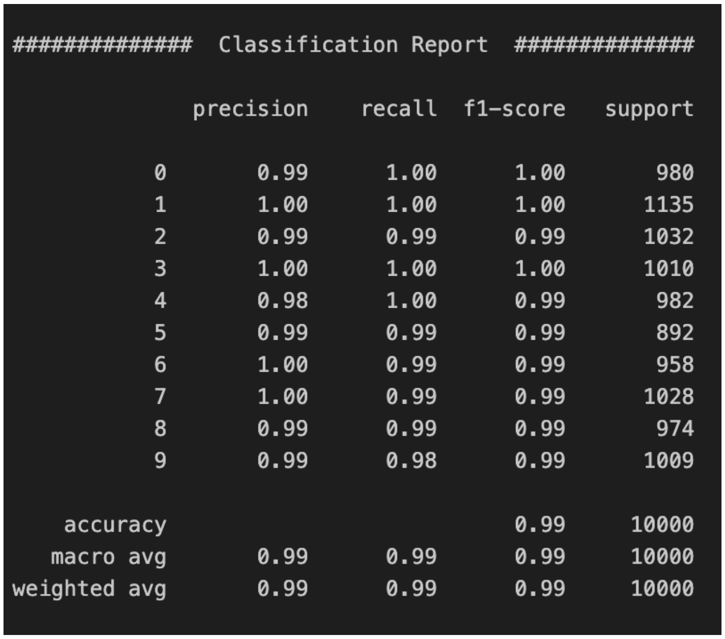


```
2
Number of images with 3 detected digits: 1
Number of images with 5 detected digits: 1
Printing wrongly localised image indices
{'3': [25], '5': [245]}
```



```
#############  Confusion Matrix  #############

[[ 977    0    0    0    1    0    1    1    0    0]
 [   0 1131    0    0    0    1    1    1    1    0]
 [   1    1 1026    0    1    0    0    2    1    0]
 [   0    0    0 1006    0    4    0    0    0    0]
 [   0    0    0    0  981    0    0    0    0    1]
 [   1    0    0    3    0  885    1    0    2    0]
 [   2    1    0    0    2    1  949    0    3    0]
 [   0    3    6    0    0    0    0 1014    0    5]
 [   1    0    1    1    1    1    0    0  967    2]
 [   1    0    0    0   13    5    1    1    0  988]]
```

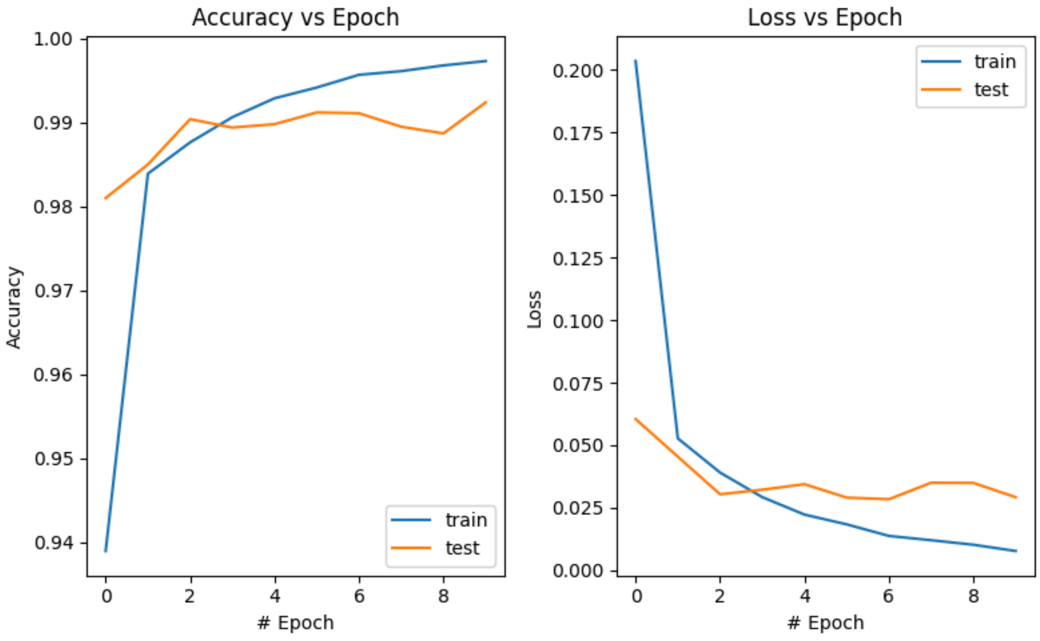


```
##############  Classification Report  ##############

              precision    recall  f1-score   support

           0       0.99      1.00      1.00       980
           1       1.00      1.00      1.00      1135
           2       0.99      0.99      0.99      1032
           3       1.00      1.00      1.00      1010
           4       0.98      1.00      0.99       982
           5       0.99      0.99      0.99       892
           6       1.00      0.99      0.99       958
           7       1.00      0.99      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
```


Accuracy vs Epoch
Accuracy
# Epoch
train
test
Loss vs Epoch
Loss
# Epoch
train
test