Handwritten Digits Recognition

Using Convolutional Neural Networks

Name: Bhushan Ingale

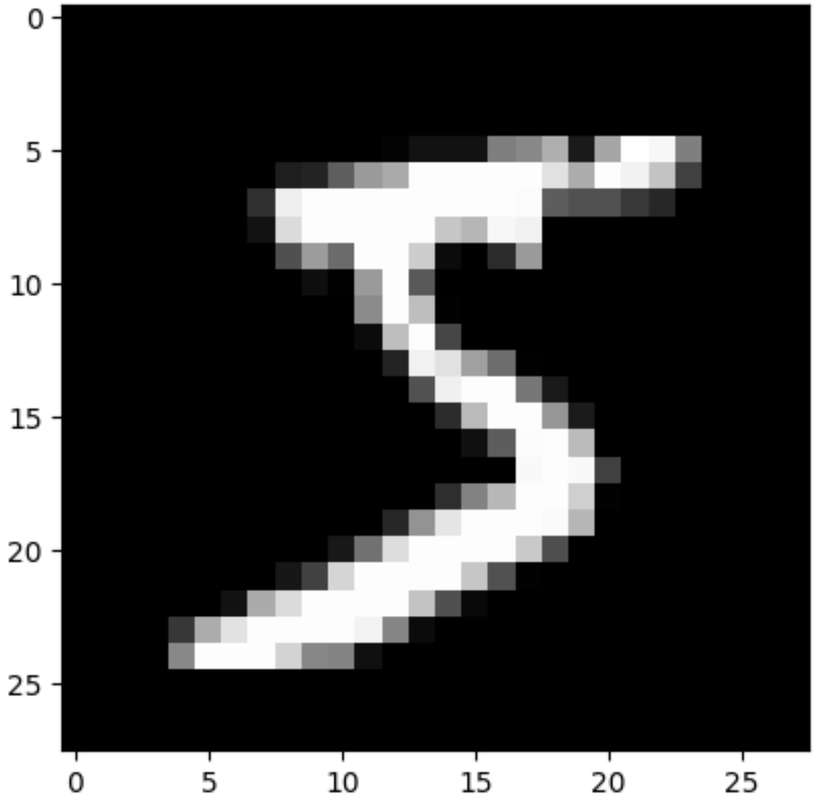
Date: 29/09/2023

Project ID: PRCP-1002-HandwrittenDigits | **Team ID:** PTID-CDS-SEP-23-1652

Objective

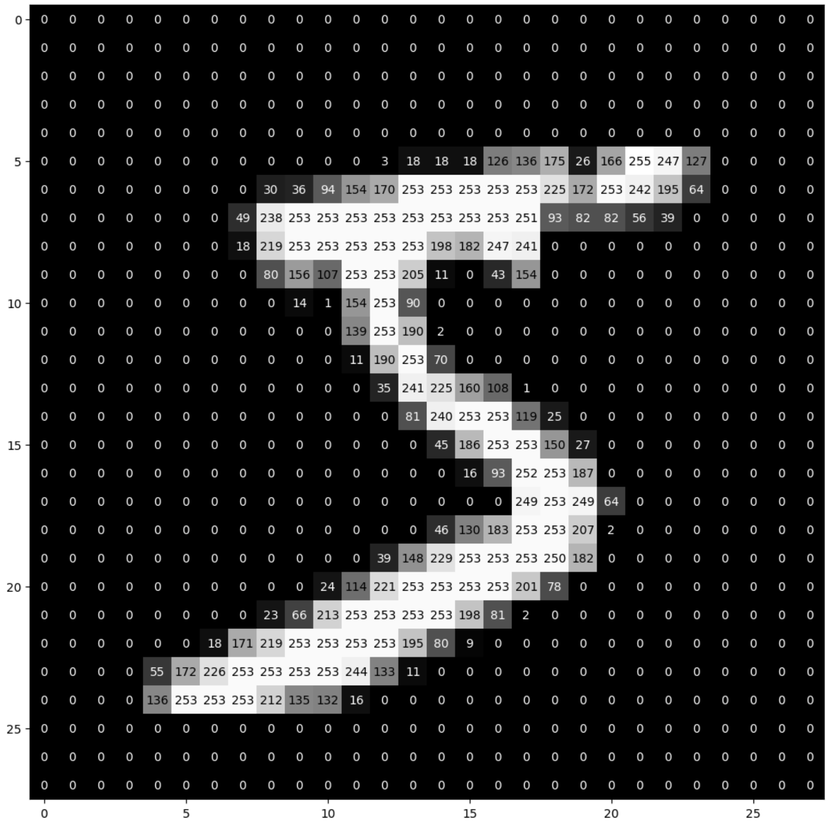
It was a digit recognition task. As such, there were ten digits (0 to 9) or ten classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy.

- Task 1: Prepared a complete data analysis report on the given data.
- Task 2: Classified a given image of a handwritten digit into one of the 10 classes representing integer values from 0 to 9.
- Task 3: Compared between various models and found the model that works better.



EDA

- Explored the various images in MNIST Dataset.
- The Dataset consists of 70000 Handwritten Images of 0 to 9 Digits.
- Each image is a 28×28-pixel square (784 pixels total). A standard split of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model, and a separate set of 10,000 images are used to test it.

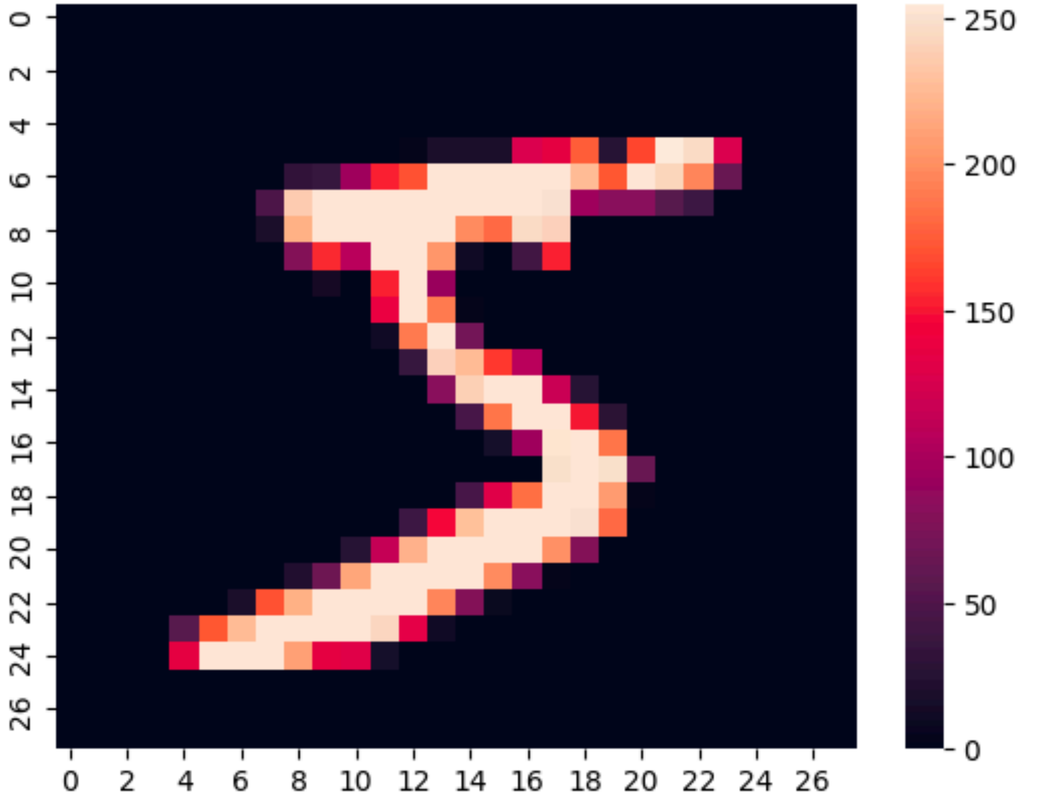


Data Preprocessing

Reshape:
The training dataset is structured as a 3-dimensional array of instance, image width, and image height. For a multi-layer perceptron model, we reduced the images down into a vector of pixels. In this case, the 28×28-sized images became 784 pixel input values.

Normalization:
To reduce the impact of illumination differences on image analysis, we performed grayscale Normalization.

Label Encoding:
To use categorical labels in our neural network, encoded them as Numerical values.



Modelling Neural Networks

Simple Neural Network
We created a Simple Neural Network with one hidden layer with the same number of neurons as there are inputs (784). A rectifier activation function `relu` is used for the neurons in the hidden layer. A `softmax` activation for output layer, `categorical_crossentropy` for loss and `adam` optimiser to learn the weights.

Simple Convolutional Neural Network
We will now create a simple CNN for MNIST dataset that demonstrates how to use all the aspects of a CNN implementation, including Convolutional layers, Pooling layers, and Dropout layers.

Larger CNN
A Convolutional Network with 9 layers.

Larger CNN Architecture

1. Convolutional layer with 30 feature maps of size 5×5
2. Pooling layer taking the max over 2*2 patches
3. Convolutional layer with 15 feature maps of size 3×3
4. Pooling layer taking the max over 2*2 patches
5. Dropout layer with a probability of 20%
6. Flatten layer
7. Fully connected layer with 128 neurons and rectifier activation
8. Fully connected layer with 50 neurons and rectifier activation
9. Output layer

Model Comparison

Model	Number of Layers	Learning Rate	Accuracy	Error Rate
Simple Neural Network	2	0.001	0.9982	1.96%
Simple Convolutional Neural Network	6	0.001	0.9878	1.22%
Modified Larger CNN	**9**	**0.001**	**0.9904**	**0.77%**

Conclusion

We have a **Convolutional Neural Network** model with **9 Layers** that has produced **99.04%** Accuracy with the minimal error of **0.77%.** This model's performance can further be enhanced by tuning the Learning Rate, Model Depth and Pixel Scaling etc.