MNIST Image Classification

Giri Sai kumar Pragada
B01027669

Introduction:

The demand for automatic image recognition systems has surged in the current digital age. Common applications include autonomous vehicles, robot assistants, and healthcare. A fundamental task in this field is the recognition of handwritten digits, which serves as a basis for many real time applications. In this project, my aim is to tackle the challenge of developing a deep learning model capable of accurately identifying handwritten digits from the MNIST dataset.

Proposal:

To address the handwritten digit recognition challenge, I propose using a deep learning approach. My project aims to develop a robust model that performs well in recognizing numbers in real world scenarios. The MNIST dataset, which is widely used for this purpose, will be imported from the Keras library.

MNIST Dataset:

The MNIST dataset (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits, widely used in image processing, data science, and machine learning. It contains 60,000 training images and 10,000 testing images, making it ideal for evaluating models and architectures. As it is a labeled dataset, developers can use it to test various models and establish baseline performance for new architectures.

Loading the Dataset:
For this project, the MNIST dataset is loaded using the import functionality from the Keras library.

Preparing the Dataset:

To ensure consistency between the training and testing phases, the dataset needs to be prepared. The MNIST dataset is already well structured, but we must reshape both the training and testing datasets into a format suitable for input into our model. Specifically, the images are reshaped to have dimensions of 28x28 pixels with a single channel (grayscale), and batch sizes are set to 60,000 for training and 10,000 for testing.

Since the dataset labels range from 0 to 9, we need to convert these labels into a format the model can interpret. One Hot encoding will be applied to transform the labels into an array of 10 elements, where each array contains one `1` and nine `0`s, representing the position of the respective digit.

Data Exploration:

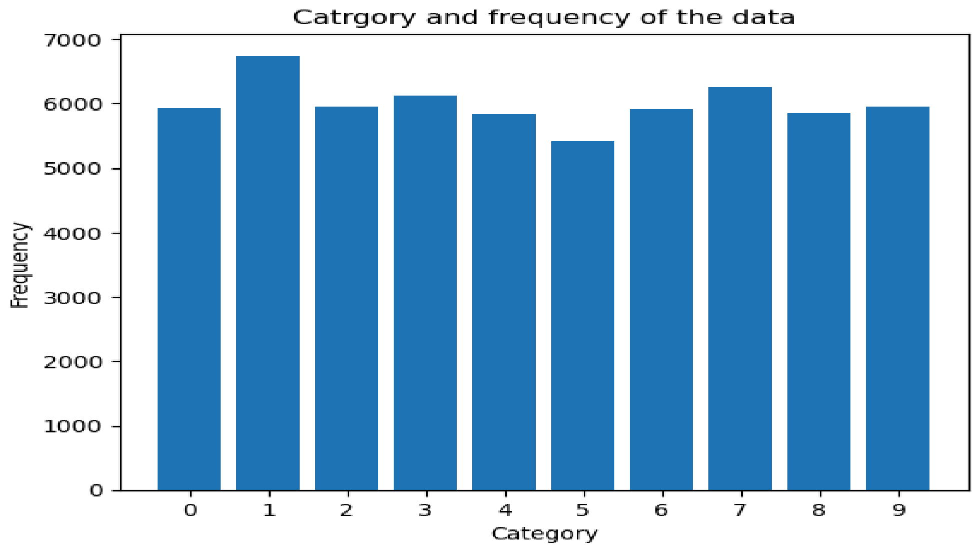
This is a crucial step in any data mining or machine learning project, as it helps in understanding the characteristics of the data from a variable perspective. Through data exploration, we can decide on the appropriate model architecture, activation functions, and other key parameters.

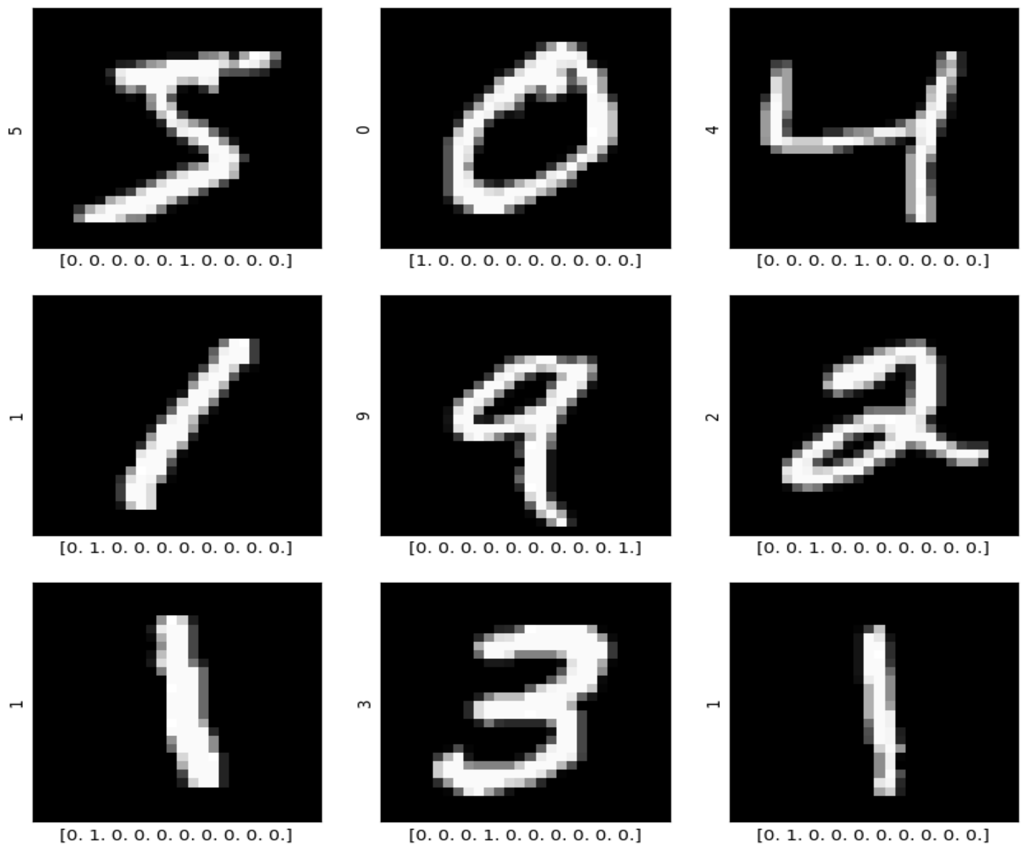
In this case, after loading the dataset, I printed the first values in `train_X` and `train_y`. As expected, `train_X` is a 3 dimensional array representing the grayscale layers of the image, where each pixel is converted into numerical values for analysis. The `train_y` array is one dimensional and follows the one hot encoded format with one `1` and nine `0`s.

A histogram was also plotted with the digit categories on the x axis and their respective frequencies on the y axis. The results show that the dataset is balanced, with roughly 6,000 samples in each category and no outliers.

To further visualize the dataset, I plotted nine images from the dataset in a 3x3 grid. The y axis corresponds to the numerical value of the digit, the x axis shows the one hot encoded label, and the image itself displays the handwritten digit. Cross Checking the images confirms that the digits and their labels match correctly.

This thorough exploration and preparation set the stage for building an effective deep learning model for handwritten digit recognition.





Methodology:

To deal the problem, I implemented three deep learning models using Keras:

1. Basic Neural Network (SNN)
2. Convolutional Neural Network (CNN)

1**) Basic Neural Network:**

For the simple neural network, I used a sequential model, which is a stack of layers.
The first layer is a flattening layer, which transforms the input images into a 1D vector.
Following the flatten layer, there are two fully connected (dense) layers, each with 128 neurons and using the ReLU activation function.
The output layer is another dense layer with 10 units (for the 10 digit categories), using the softmax activation function to output class probabilities.

For optimization, the model uses the "Adam" optimizer, and the loss function applied is "categorical binary cross entropy."

The model was trained for 10 epochs with a batch size of 128. After training, the model's performance was evaluated and the results were stored.

2**) Convolutional Neural Network:**

Based on the observations from the simple neural network, I built a CNN model to enhance the accuracy. A sequential Keras model was used to define the CNN architecture.

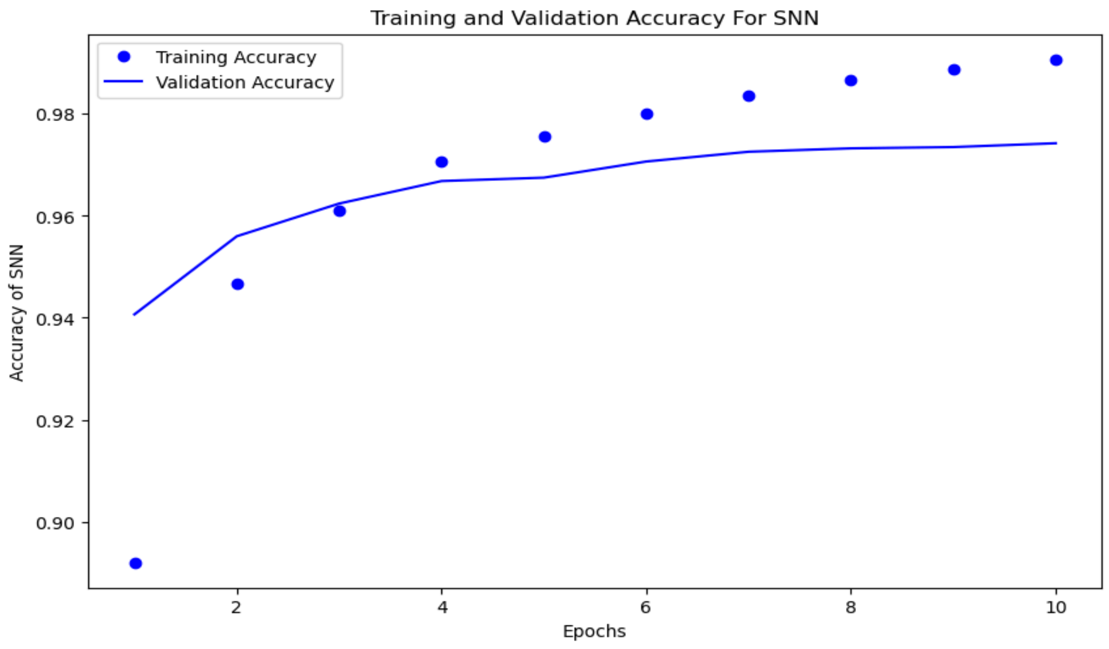
The CNN consists of two convolutional layers with 32 and 64 filters, respectively, both using the ReLU activation function.After each convolutional layer, a max pooling layer is applied to downsample the feature maps, reducing the spatial dimensions and capturing the most important features.After the convolutional and pooling layers, I added two fully connected dense layers. The first dense layer has 128 units and uses ReLU activation, while the output layer has 10 units and uses softmax activation.

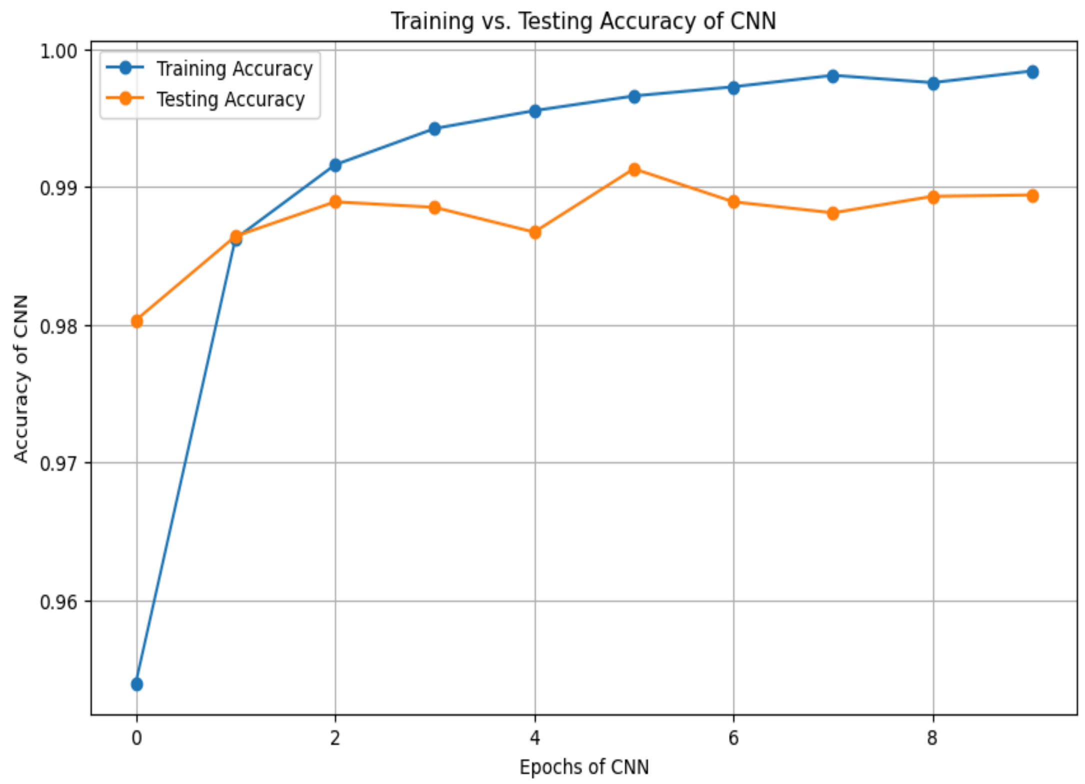
The configuration of the dense layers is similar to the basic neural network model, but the added convolutional layers improve the model's ability to extract important image features, making CNN more effective for image classification.

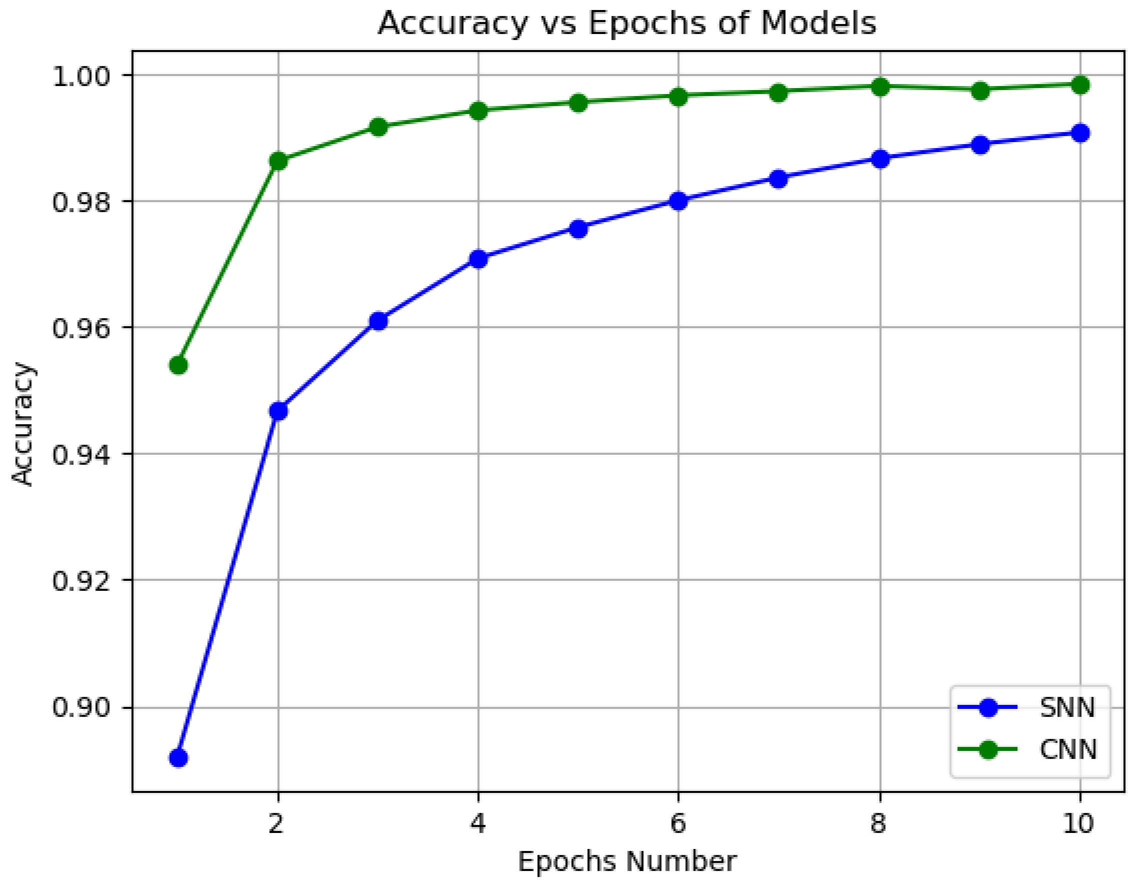
The model is compiled using categorical cross entropy as the loss function and the "Adam" optimizer. The output is evaluated based on the accuracy during testing.

By comparing the performance of both models, CNN is expected to outperform the simple neural network due to its ability to capture spatial hierarchies in the image data.

Results and comparisons:




Training vs. Testing Accuracy of CNN
Training Accuracy
Testing Accuracy
Accuracy of CNN
Epochs of CNN
1.00
0.99
0.98
0.97
0.96
0
2
4
6
8


Accuracy vs Epochs of Models
SNN
CNN
Accuracy
Epochs Number
1.00
0.98
0.96
0.94
0.92
0.90
2
4
6
8
10

```
   S.No              Neural Network Model  Accuracy
0     1             Simple Neural Network    0.9762
1     2  Convolutional Neural Network    0.9894
```

Based on the results of the two models, the accuracies achieved are as follows:

As expected, the Convolutional Neural Network (CNN) outperforms the Simple Neural Network (SNN), achieving a higher accuracy of 98.94%, compared to 97.62% for the SNN. This demonstrates the effectiveness of CNNs in image recognition tasks by capturing spatial features more efficiently.

conclusion:

The basic neural network consisted of a single hidden layer with 128 neurons and a softmax output layer with 10 neurons. After 10 training epochs, the model achieved a test accuracy of 0.9762. Although the accuracy is high, this model's performance is limited by its relatively simple architecture.

The CNN model included two convolutional layers with 32 and 64 filters, followed by two fully connected dense layers with 128 and 10 neurons, respectively. After 10 training epochs, the CNN model achieved a higher test accuracy of 0.9894. This improvement can be attributed to CNN's ability to capture spatial features and patterns in the image data, which the simple neural network could not fully exploit.

These findings demonstrate that while the Simple Neural Network (SNN) performs well, the Convolutional Neural Network (CNN) offers superior accuracy due to its more complex architecture. The results indicate that CNN is better suited for image recognition tasks, making it a more effective model for handwritten digit identification.