CONVOLUTIONAL NEURAL NETWORK (CNN) FOR MNIST CLASSIFICATION

INTRODUCTION

The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0-9), each of size 28x28 pixels. Our goal is to build a Convolutional Neural Network (CNN) to classify these digits into the correct categories (0-9).

PREPROCESSING

- The images were loaded using Keras' built-in MNIST dataset loader.
- Pixel values were normalized to the range [0, 1] by dividing each pixel by 255.
- The images were reshaped to have an additional channel dimension, making them suitable for the CNN input (28x28x1).
- The labels were converted to one-hot encoded vectors, which is required for categorical classification.

MODEL ARCHITECTURE

The model was constructed using the Sequential API from Keras and follows a layered architecture designed for hierarchical feature extraction.

The layers of the model include:

- **Input Layer**: 28x28 images with 1 channel (grayscale).
- **Convolutional Layers**: Four convolutional layers with increasing filter sizes (32, 64, 128, and 256), each using a 3x3 kernel and ReLU activation. These layers are designed to extract spatial features from the images.

- **Max Pooling Layers**: Two max-pooling layers with a 2x2 window to downsample the feature maps, reducing dimensionality and computation.
- **Fully Connected Layers**: Three fully connected layers (128, 50, and 50 units) with ReLU activation. These layers act as the classifier based on the features extracted by the convolutional layers.
- **Dropout Layer**: A dropout layer (0.5 dropout rate) was added after the first fully connected layer to prevent overfitting.
- **Output Layer**: A softmax activation layer with 10 units to classify the images into one of the 10 digit categories (0-9).

TRAINING PROCESS

The model was compiled using the Adam optimizer and categorical cross-entropy loss, which is suitable for multi-class classification. The model was trained on the training set (60,000 images) for 10 epochs, with a batch size of 32, and validated on the test set (10,000 images).

The training process involved plotting the training and validation accuracy to monitor overfitting and adjust hyperparameters accordingly.

RESULTS

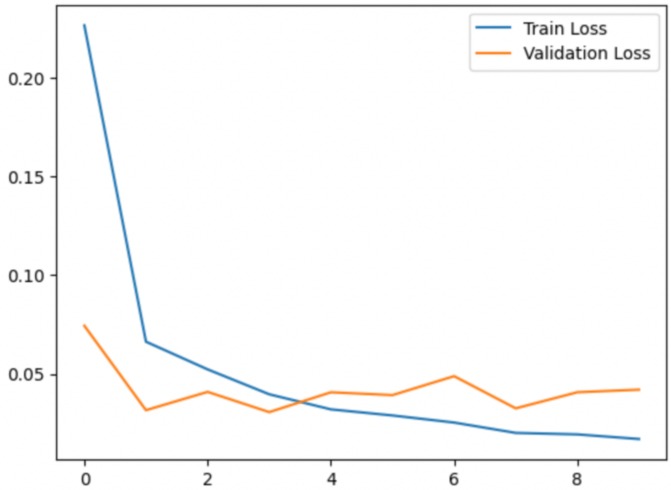
The CNN model was trained and evaluated on the MNIST dataset. The following metrics were observed:

- **Training Loss**: 0.0085
- **Training Accuracy**: 99.80%
- **Test Loss**: 0.0420
- **Test Accuracy**: 99.25%

These results indicate that the model generalizes well to the test data, as the test accuracy is only slightly lower than the training accuracy, with a small test loss value.

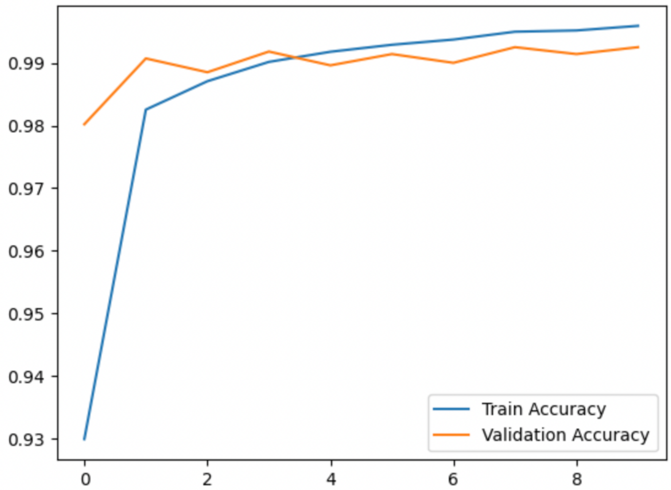
Training and Validation Loss:

The following graph shows the training and validation loss over 10 epochs. The training loss decreased steadily, while the validation loss remained consistently low, indicating that the model was not overfitting significantly.



Training and Validation Accuracy:

The graph below illustrates the training and validation accuracy over the same 10 epochs. The training accuracy increased steadily and reached 99.80%, while the validation accuracy stabilized around 99.25%. This demonstrates that the model achieved high generalization on unseen data.



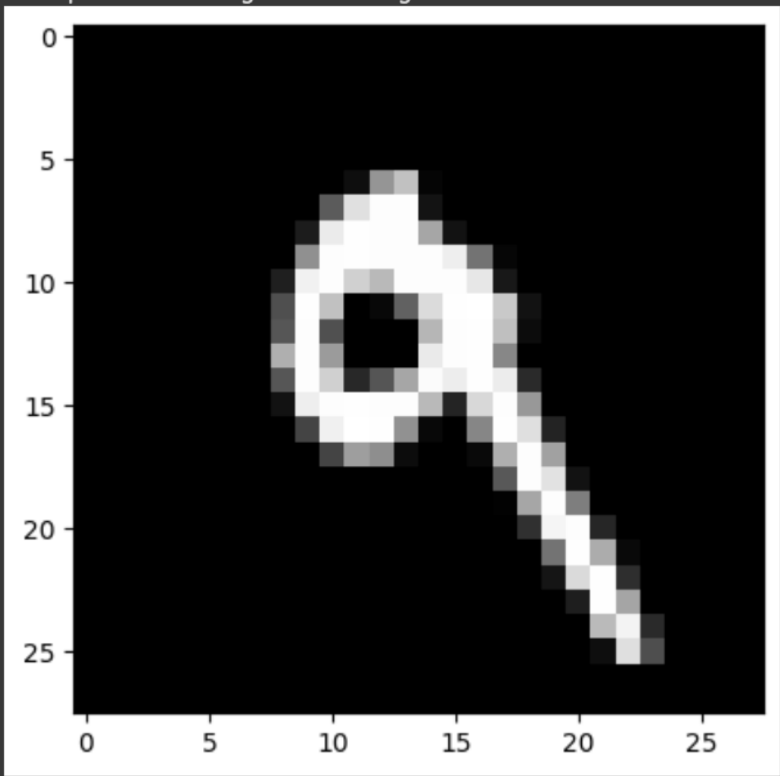
Example of Model Prediction:

To demonstrate the model's performance, an example prediction was made using an image from the test set. The image displayed was of the digit "9", and the model correctly predicted it as "9". This prediction showcases the model's effectiveness in classifying handwritten digits.

The model predicted the digit with high confidence, as shown below:

```
[23] plt.imshow(x_test[7],cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7d2234847f10>
```




```
[24] x = np.reshape(x_test[7], (1, 28, 28, 1))
     np.argmax(model.predict(x))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
9
```

CONCLUSION:

A Convolutional Neural Network (CNN) was designed to classify handwritten digits from the MNIST dataset. The model achieved excellent performance with **99.80% training accuracy** and **99.25% test accuracy**, indicating strong generalization to unseen data. The use of multiple convolutional layers and dropout helped to prevent overfitting, as reflected in the low test loss of **0.0420**.

An example prediction showed the model accurately classifying a digit "9" from the test set, further demonstrating its effectiveness. Therefore, this CNN architecture successfully solved the digit recognition task with high accuracy and reliability.