

The goal of this project is to build a Convolutional Neural Network (CNN) to recognize handwritten digits from the MNIST dataset. The MNIST dataset contains 70,000 grayscale images of digits (0-9), each 28x28 pixels in size.

- ✓ Using TensorFlow and Keras, we will create and train a deep learning model to classify these digits accurately. The project involves data preprocessing, model building, training, and evaluation to understand CNNs and their application.

a. Load the MNIST dataset

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
```

```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
```

➡ Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.11490434/11490434> 0s 0us/step

b. Normalize the pixel values between 0 and 1

```
x_train=x_train.astype('float32')/255
x_test=x_test.astype('float32')/255

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

c. Build a CNN with Convolutional layers

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
```

```
→ /usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107:
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

d. Include MaxPooling layers

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

e. Add Dense layers and the correct output layer with 10 neurons and softmax activation

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

f. Use the 'adam' optimizer, Set the loss function to 'categorical_crossentropy'

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

g. Train the model and track both training and validation accuracy/loss

```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```


```
history = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_t
```

```
→ Epoch 1/10
1875/1875 ————— 30s 15ms/step - accuracy: 0.1098 - loss: 2.3019 - val_acc
Epoch 2/10
1875/1875 ————— 28s 15ms/step - accuracy: 0.1118 - loss: 2.3016 - val_acc
Epoch 3/10
1875/1875 ————— 41s 15ms/step - accuracy: 0.1125 - loss: 2.3010 - val_acc
Epoch 4/10
1875/1875 ————— 28s 15ms/step - accuracy: 0.1115 - loss: 2.3015 - val_acc
Epoch 5/10
1875/1875 ————— 41s 15ms/step - accuracy: 0.1106 - loss: 2.3016 - val_acc
Epoch 6/10
1875/1875 ————— 28s 15ms/step - accuracy: 0.1133 - loss: 2.3011 - val_acc
Epoch 7/10
1875/1875 ————— 29s 16ms/step - accuracy: 0.1127 - loss: 2.3015 - val_acc
Epoch 8/10
1875/1875 ————— 28s 15ms/step - accuracy: 0.1095 - loss: 2.3018 - val_acc
```

Epoch 9/10

1875/1875  **41s** 15ms/step - accuracy: 0.1130 - loss: 2.3013 - val_acc


Epoch 10/10

1875/1875  **28s** 15ms/step - accuracy: 0.1154 - loss: 2.3010 - val_acc

h. Evaluate the model on test data

```

from re import VERBOSE
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test Accuracy: {test_accuracy}')
```

**313/313**  **2s** 5ms/step - accuracy: 0.1160 - loss: 2.3012

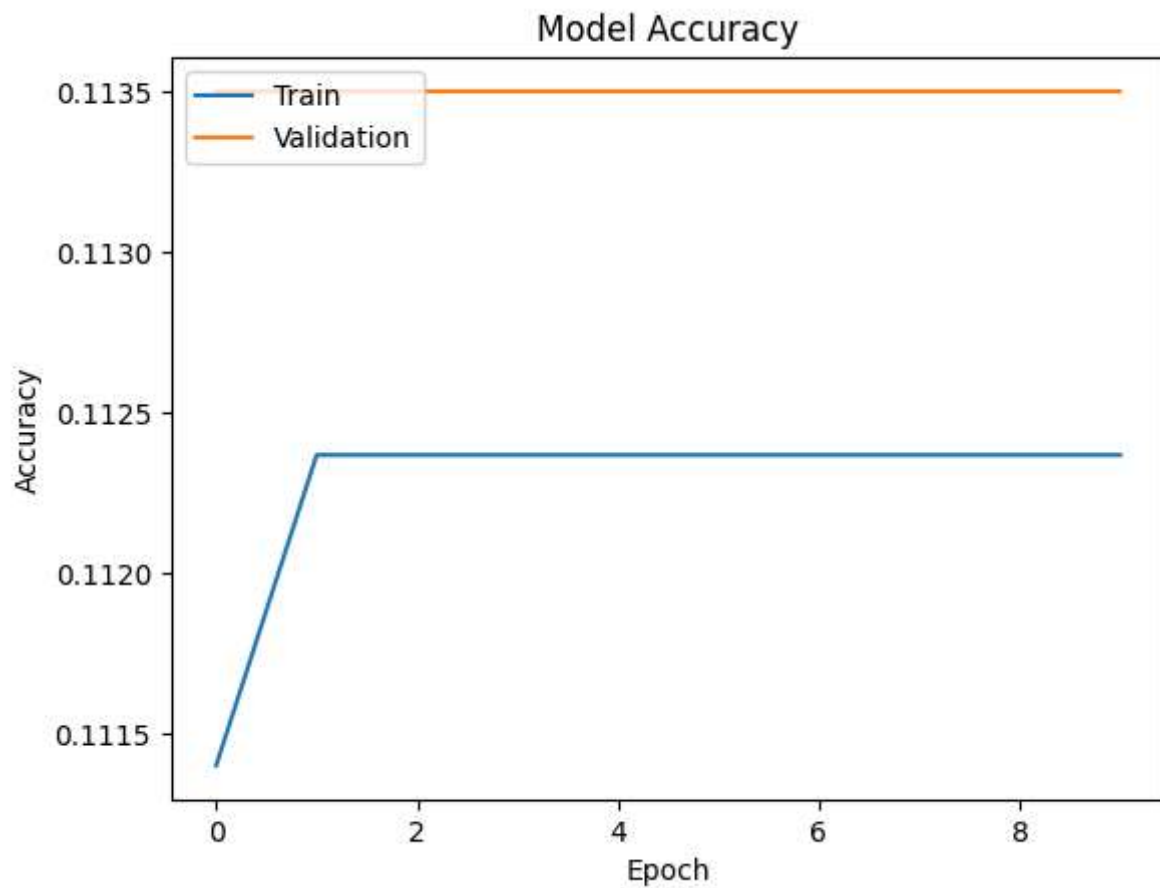
Test Accuracy: 0.11349999904632568

i. Plot accuracy and loss graphs

#Plot Accuracy

```

import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],label='Training Accuracy')
plt.plot(history.history['val_accuracy'],label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



#Plot Loss

```
plt.plot(history.history['loss'],label='Training Loss')
plt.plot(history.history['val_loss'],label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



This project aims to build a Convolutional Neural Network (CNN) using the MNIST dataset for recognizing handwritten digits (0-9). The dataset consists of 70,000 grayscale images, each of size 28x28 pixels, divided into 60,000 training images and 10,000 test images. The goal is to classify these images into the correct digit categories.

Data Preprocessing: Load the MNIST data, normalize the pixel values, and reshape the images as needed. **Model Architecture:** Build a CNN with convolutional layers, followed by max-pooling and dense layers. The final layer will use the softmax activation function to output probabilities for each digit class (0-9). **Training and Evaluation:** Train the model on the training data while validating on the validation set. Track accuracy and loss during training and evaluate the final model on the test data. **Visualization:** Plot the training/validation accuracy and loss graphs to analyze the model's performance over time.

Start coding or [generate](#) with AI.