

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np
```

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

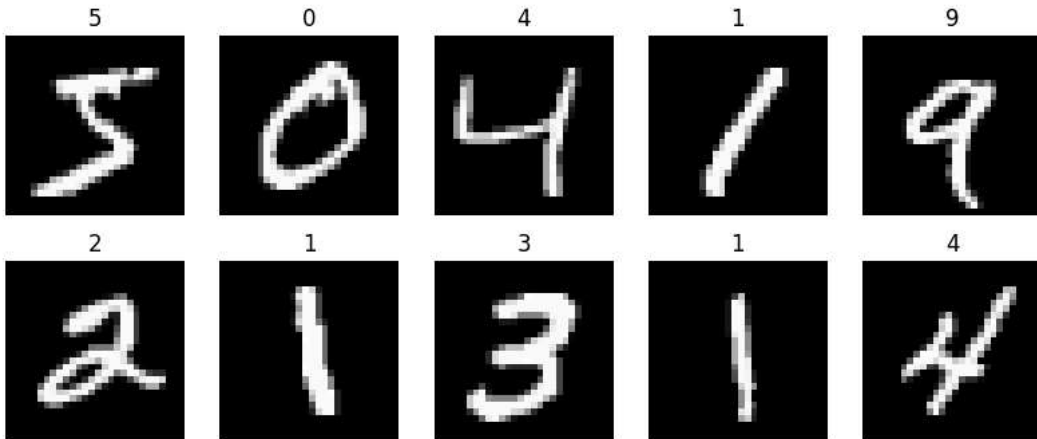
```
print("Training images:", x_train.shape)
print("Testing images:", x_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 — 0s 0us/step
Training images: (60000, 28, 28)
Testing images: (10000, 28, 28)
```

```
plt.figure(figsize=(10,4))

for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(y_train[i])
    plt.axis('off')

plt.show()
```



```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
```

```
x_train = x_train.reshape(-1,28,28,1)
x_test = x_test.reshape(-1,28,28,1)

print(x_train.shape)
```

```
(60000, 28, 28, 1)
```

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

```
datagen = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.10,
    width_shift_range=0.10,
    height_shift_range=0.10
)

datagen.fit(x_train)
```

```

model = Sequential()

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

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

model.add(Dense(128,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(10,activation='softmax'))

```

```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

```

```

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

```

```

history = model.fit(
    datagen.flow(x_train,y_train,batch_size=64),
    epochs=5,
    validation_data=(x_test,y_test)
)

```

```

Epoch 1/5
938/938 ————— 65s 67ms/step - accuracy: 0.8645 - loss: 0.4277 - val_accuracy: 0.9813 - val_loss: 0.0549
Epoch 2/5
938/938 ————— 61s 65ms/step - accuracy: 0.9474 - loss: 0.1738 - val_accuracy: 0.9873 - val_loss: 0.0356
Epoch 3/5
938/938 ————— 62s 66ms/step - accuracy: 0.9603 - loss: 0.1312 - val_accuracy: 0.9907 - val_loss: 0.0263
Epoch 4/5
938/938 ————— 60s 64ms/step - accuracy: 0.9683 - loss: 0.1087 - val_accuracy: 0.9924 - val_loss: 0.0247
Epoch 5/5
938/938 ————— 62s 66ms/step - accuracy: 0.9710 - loss: 0.0953 - val_accuracy: 0.9936 - val_loss: 0.0201

```

```

loss, accuracy = model.evaluate(x_test,y_test)

```

```

print("Test Accuracy =", accuracy)
print("Test Loss =", loss)

```

```

313/313 ————— 2s 7ms/step - accuracy: 0.9936 - loss: 0.0201
Test Accuracy = 0.9936000108718872
Test Loss = 0.02005731873214245

```

```

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

```

```

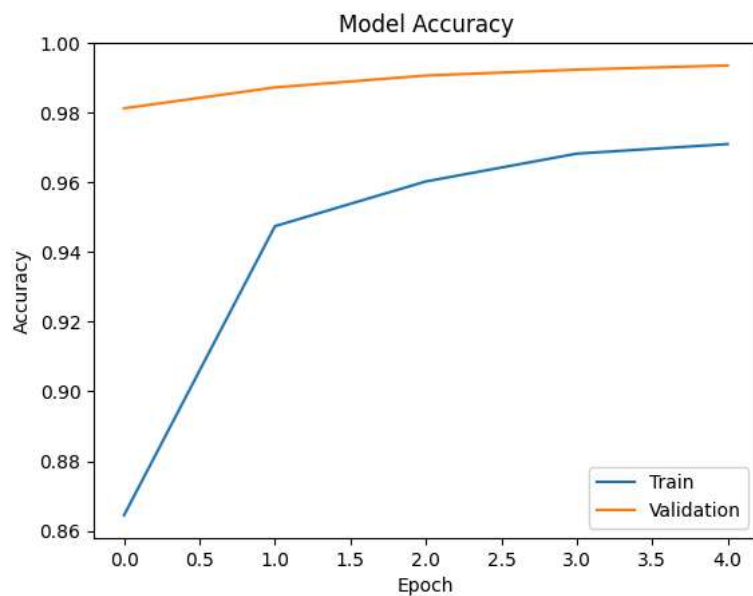
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Train","Validation"])

```

```

plt.show()

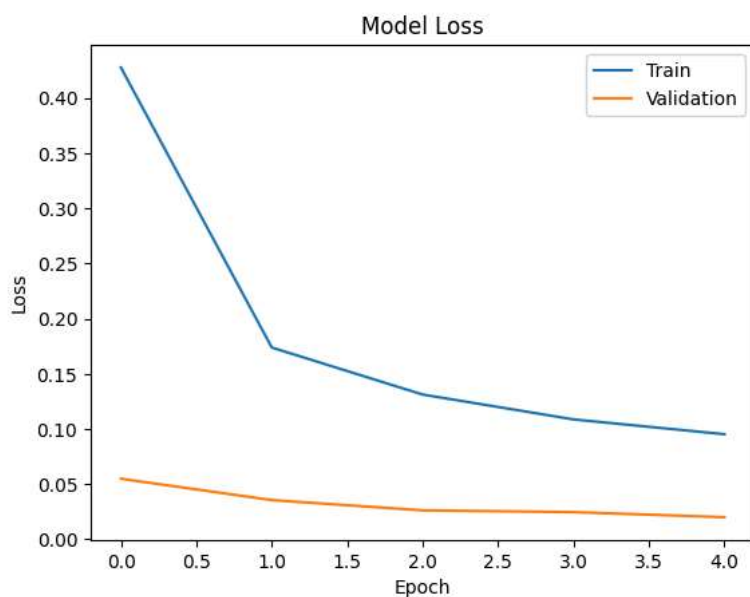
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title("Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Train", "Validation"])

plt.show()
```

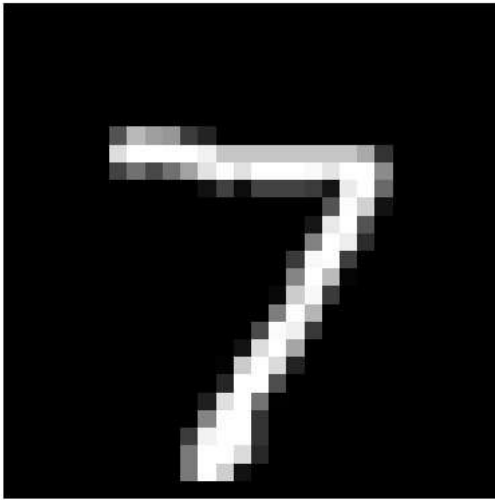


```
prediction = model.predict(x_test)

plt.imshow(x_test[0].reshape(28,28), cmap='gray')
plt.axis('off')

print("Predicted Digit =", np.argmax(prediction[0]))
```

313/313 — 2s 7ms/step
Predicted Digit = 7



```
model.save("MNIST_CNN_Model.h5")  
print("Model saved successfully.")
```