Program 3: Design and implement a Convolutional Neural Network (CNN) for classification of image dataset

AIM

To **design and implement a Convolutional Neural Network (CNN)** for the **classification of an image dataset**

Algorithm

Step 1: Import Libraries

Step 2: Load Dataset

Step 3: Preprocess the Data

Step 4: Build the CNN Model

Step 5: Compile the Model

Step 6: Train the Model

Step 7: Evaluate the Model

Step 8: Make Predictions

Step 9: Visualize Training Performance

Program:

```
# Import necessary libraries

import tensorflow as tf
from tensorflow.keras import models, layers

import matplotlib.pyplot as plt

import numpy as np

# MNIST dataset: 28x28 grayscale images of handwritten digits (0-9)

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()


# Display the shape of dataset

print("Training data shape:", x_train.shape, y_train.shape)

print("Testing data shape:", x_test.shape, y_test.shape)
```

```
# Preprocess the data- Normalize pixel values from [0,255] to [0,1]
x_train, x_test = x_train / 255.0, x_test / 255.0

# CNNs expect input with channels -> reshape to (samples, height, width, channels)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

#Build CNN Model
model = models.Sequential([
    # Convolutional Layer 1
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),

    # Convolutional Layer 2
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Flatten the output from 2D to 1D
    layers.Flatten(),

    # Fully Connected Layer
    layers.Dense(64, activation='relu'),

    # Output Layer (10 classes for digits 0-9)
    layers.Dense(10, activation='softmax')
])

# Show model summary
```

```
model.summary()

# Step 4: Compile the Model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Step 5: Train the Model
history = model.fit(x_train, y_train, epochs=3, validation_data=(x_test, y_test))

# Step 6: Evaluate Model Performance
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest Accuracy: {accuracy:.4f}")

# Step 7: Make Predictions
predictions = model.predict(x_test[:5])

# Display first 5 test images, their true labels, and predicted labels
for i in range(5):
    plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
    plt.title(f"True: {y_test[i]} | Predicted: {np.argmax(predictions[i])}")
    plt.axis('off')
    plt.show()

# Step 8: Plot Training History (Loss & Accuracy)
plt.figure(figsize=(12, 4))

# Plot training & validation loss
```

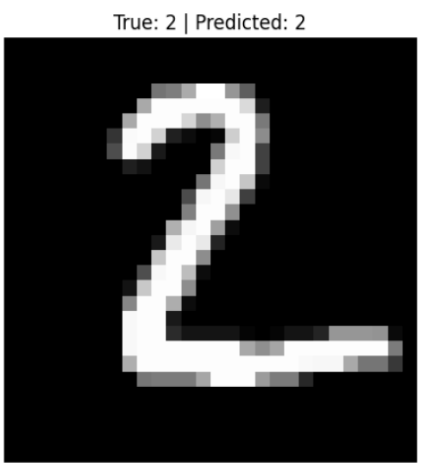
```
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot training & validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

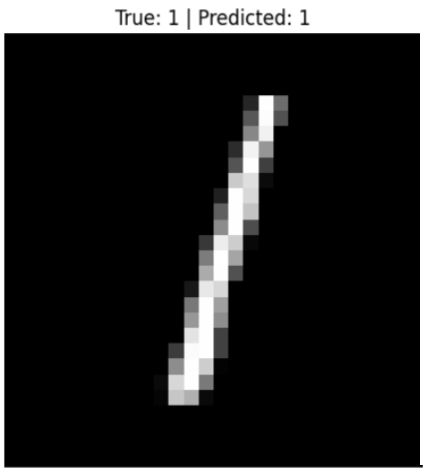
```
Training data shape: (60000, 28, 28) (60000,)
Testing data shape: (10000, 28, 28) (10000,)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_21"
```

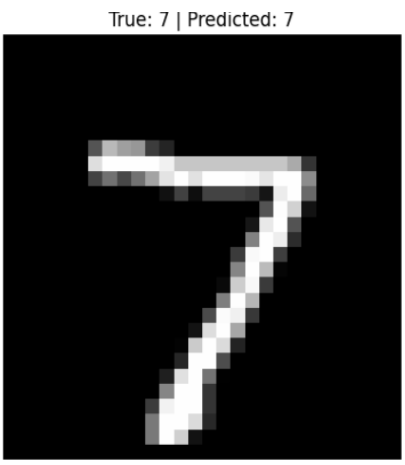
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_7 (Flatten)	(None, 1600)	0
dense_47 (Dense)	(None, 64)	102,464
dense_48 (Dense)	(None, 10)	650

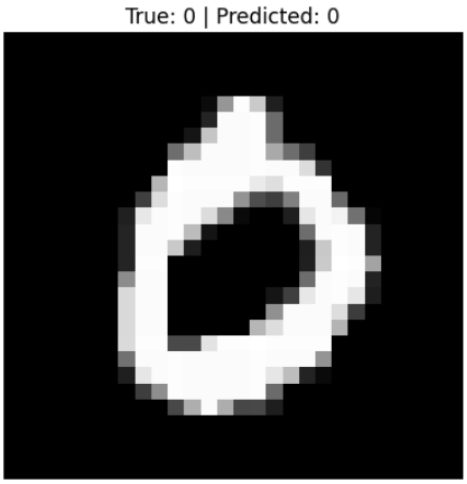
Total params: 121,930 (476.29 KB)
Trainable params: 121,930 (476.29 KB)
Non-trainable params: 0 (0.00 B)

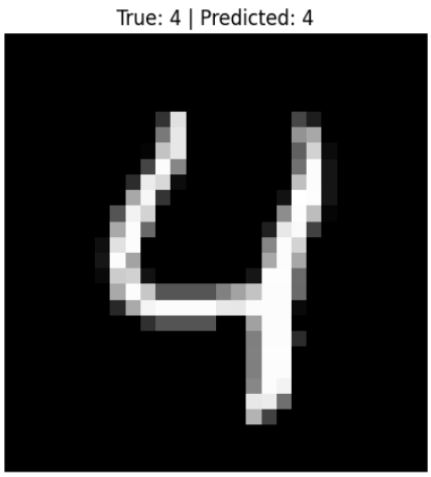
```
Epoch 1/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 32s 16ms/step - accuracy: 0.9033 - loss: 0.3094 - val_accuracy: 0.9859 - val_loss: 0.0438
Epoch 2/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 40s 16ms/step - accuracy: 0.9858 - loss: 0.0460 - val_accuracy: 0.9894 - val_loss: 0.0336
Epoch 3/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 41s 16ms/step - accuracy: 0.9895 - loss: 0.0330 - val_accuracy: 0.9896 - val_loss: 0.0291

Test Accuracy: 0.9896
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
```

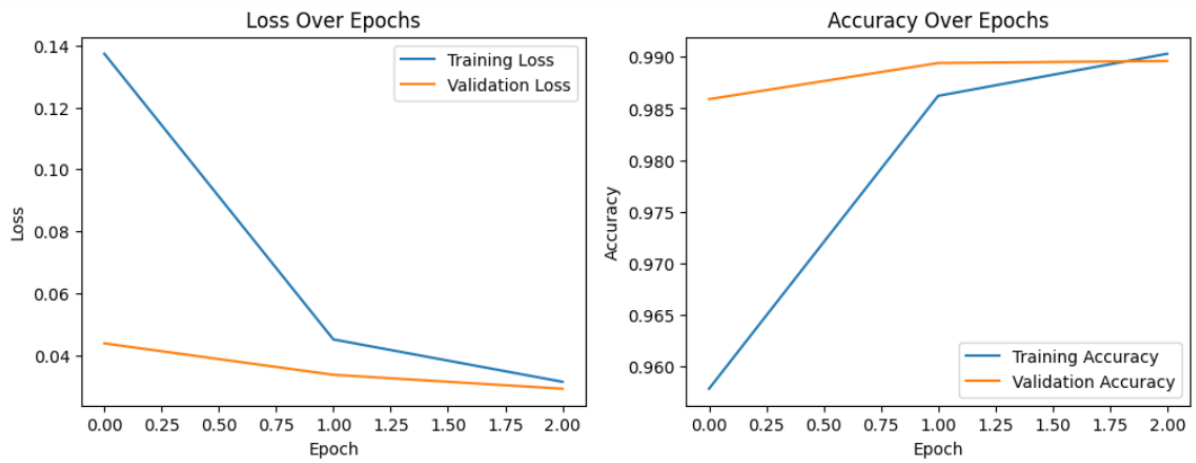











Loss Over Epochs
Training Loss
Validation Loss
Loss
Epoch
0.14
0.12
0.10
0.08
0.06
0.04
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Accuracy Over Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epoch
0.990
0.985
0.980
0.975
0.970
0.965
0.960