SOURCE CODE :

```
# Import necessary libraries
Import tensorflow as tf
From tensorflow.keras.datasets import mnist
From tensorflow.keras.models import Sequential
From tensorflow.keras.layers import Dense, Flatten
From tensorflow.keras.utils import to_categorical
Import matplotlib.pyplot as plt

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize image data (0-1)
X_train, x_test = x_train / 255.0, x_test / 255.0

# One-hot encode labels
Y_train = to_categorical(y_train, 10)
Y_test = to_categorical(y_test, 10)

# Build a simple neural network model
Model = Sequential([
    Flatten(input_shape=(28, 28)),        # Flatten 28x28 images to 784 vector
    Dense(128, activation='relu'),        # Hidden layer
    Dense(10, activation='softmax')       # Output layer (10 classes for digits 0-9)
])

# Compile the model
```

```
Model.compile(optimizer='adam',
              Loss='categorical_crossentropy',
              Metrics=['accuracy'])

# Train the model
Model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

# Evaluate the model on test data
Loss, accuracy = model.evaluate(x_test, y_test)
Print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Save the model (optional)
Model.save("digit_recognizer_model.h5")

# Optional: Test a few predictions
Import numpy as np

# Predict and visualize 5 random test samples
Num_samples = 5
Indices = np.random.choice(len(x_test), num_samples, replace=False)

For I in indices:
    Img = x_test[i]
    Label = np.argmax(y_test[i])
    Prediction = np.argmax(model.predict(img.reshape(1, 28, 28)))

    Plt.imshow(img, cmap='gray')
```

```
Plt.title(f”True: {label}, Predicted: {prediction}”)
Plt.axis(‘off’)
Plt.show()
```

Output :

/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.

Super().__init__(**kwargs)

Epoch 1/5

1688/1688 □□□□□□□□□□□□□□□□□□□□ 10s 5ms/step – accuracy: 0.8700 – loss: 0.4594 – val_accuracy: 0.9663 – val_loss: 0.1260

Epoch 2/5

1688/1688 □□□□□□□□□□□□□□□□□□□□ 7s 4ms/step – accuracy: 0.9612 – loss: 0.1362 – val_accuracy: 0.9695 – val_loss: 0.1005

Epoch 3/5

1688/1688 □□□□□□□□□□□□□□□□□□□□ 10s 4ms/step – accuracy: 0.9734 – loss: 0.0866 – val_accuracy: 0.9717 – val_loss: 0.0909

Epoch 4/5

1688/1688 □□□□□□□□□□□□□□□□□□□□ 11s 4ms/step – accuracy: 0.9806 – loss: 0.0621 – val_accuracy: 0.9778 – val_loss: 0.0848

Epoch 5/5

1688/1688 □□□□□□□□□□□□□□□□□□□□ 11s 5ms/step – accuracy: 0.9862 – loss: 0.0442 – val_accuracy: 0.9785 – val_loss: 0.0780

313/313 □□□□□□□□□□□□□□□□□□□□ 1s 2ms/step – accuracy: 0.9701 – loss: 0.0908

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.

Test Accuracy: 97.46%

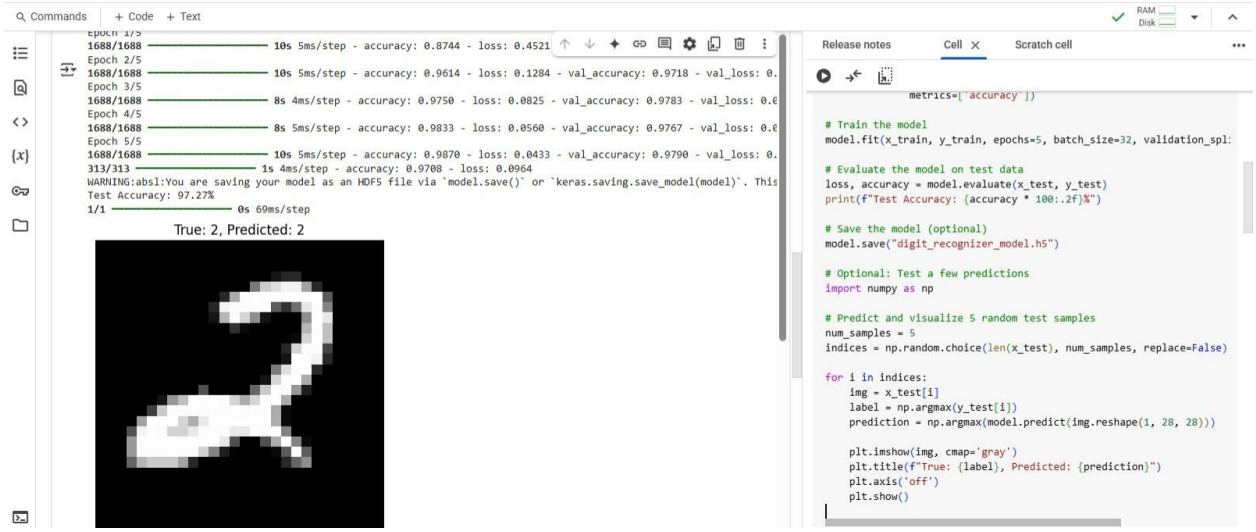
1/1 □□□□□□□□□□□□□□□□□□□□ 0s 73ms/step

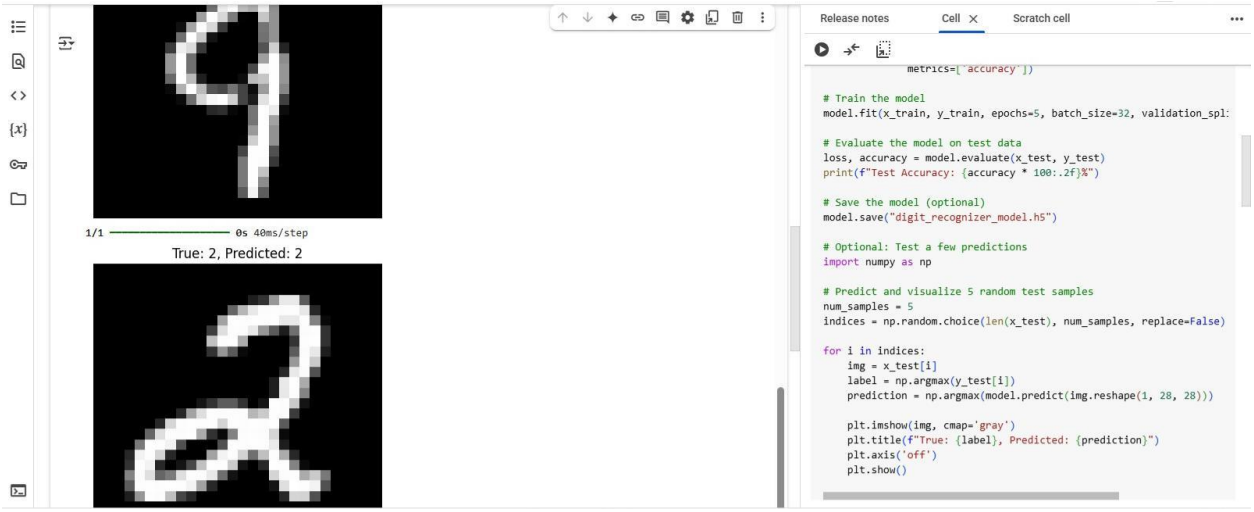
1/1 □□□□□□□□□□□□□□□□□□□□ 0s 47ms/step

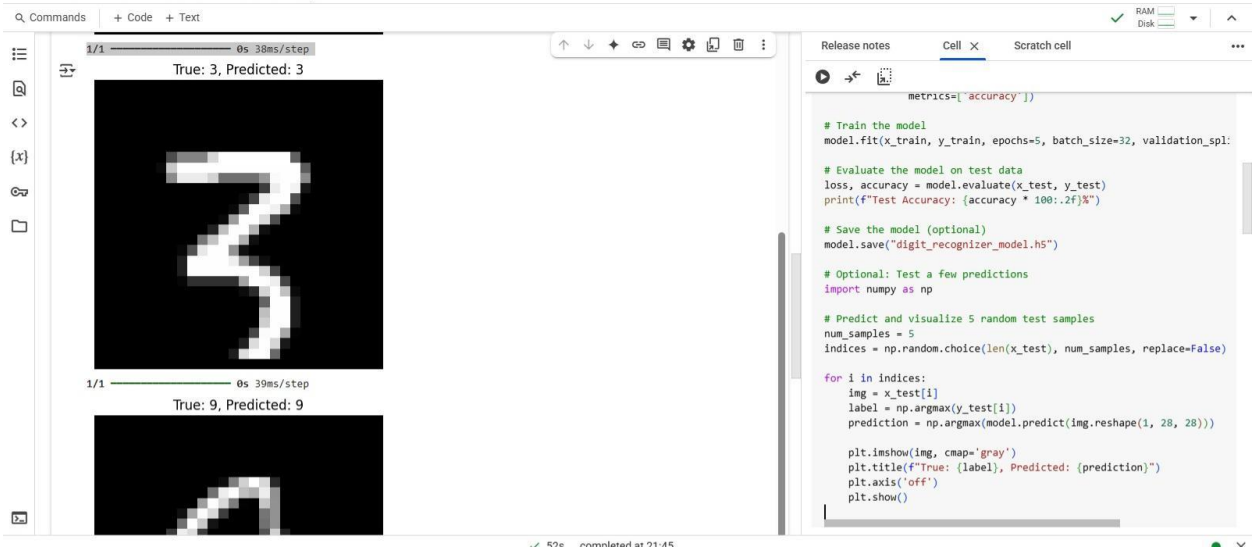
1/1 □□□□□□□□□□□□□□□□□□□□ 0s 40ms/step

1/1 □□□□□□□□□□□□□□□□□□□□ 0s 40ms/step

1/1 □□□□□□□□□□□□□□□□□□□□ 0s 39ms/step




1/1 0s 40ms/step
True: 2, Predicted: 2
Release notes
Cell
Scratch cell
metrics=['accuracy'])
# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=32, validation_spl
Evaluate the model on test data
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
# Save the model (optional)
model.save("digit_recognizer_model.h5")
# Optional: Test a few predictions
import numpy as np
# Predict and visualize 5 random test samples
num_samples = 5
indices = np.random.choice(len(x_test), num_samples, replace=False)
for i in indices:
img = x_test[i]
label = np.argmax(y_test[i])
prediction = np.argmax(model.predict(img.reshape(1, 28, 28)))
plt.imshow(img, cmap='gray')
plt.title(f"True: {label}, Predicted: {prediction}")
plt.axis('off')
plt.show()


Commands
+ Code
+ Text
RAM
Disk
1/1 0s 38ms/step
True: 3, Predicted: 3
1/1 0s 39ms/step
True: 9, Predicted: 9
Release notes
Cell
Scratch cell
metrics=['accuracy'])
# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=32, validation_spl
Evaluate the model on test data
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
# Save the model (optional)
model.save("digit_recognizer_model.h5")
# Optional: Test a few predictions
import numpy as np
# Predict and visualize 5 random test samples
num_samples = 5
indices = np.random.choice(len(x_test), num_samples, replace=False)
for i in indices:
img = x_test[i]
label = np.argmax(y_test[i])
prediction = np.argmax(model.predict(img.reshape(1, 28, 28)))
plt.imshow(img, cmap='gray')
plt.title(f"True: {label}, Predicted: {prediction}")
plt.axis('off')
plt.show()