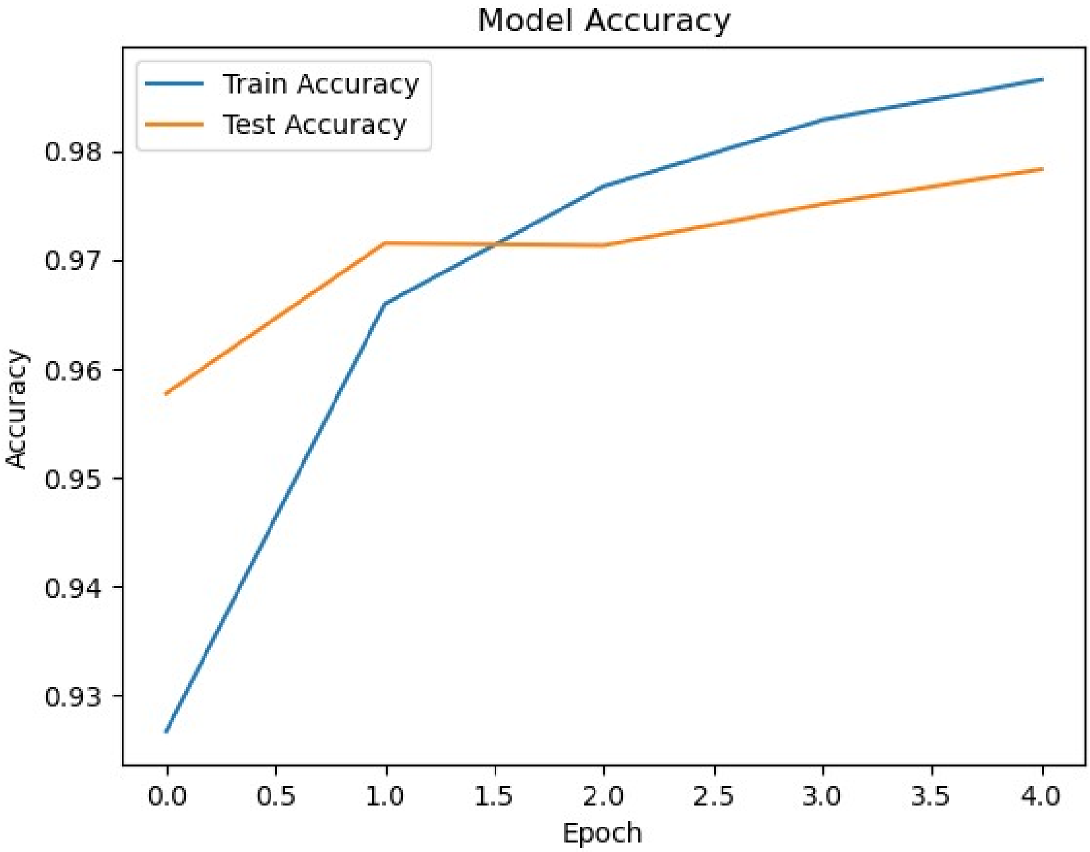
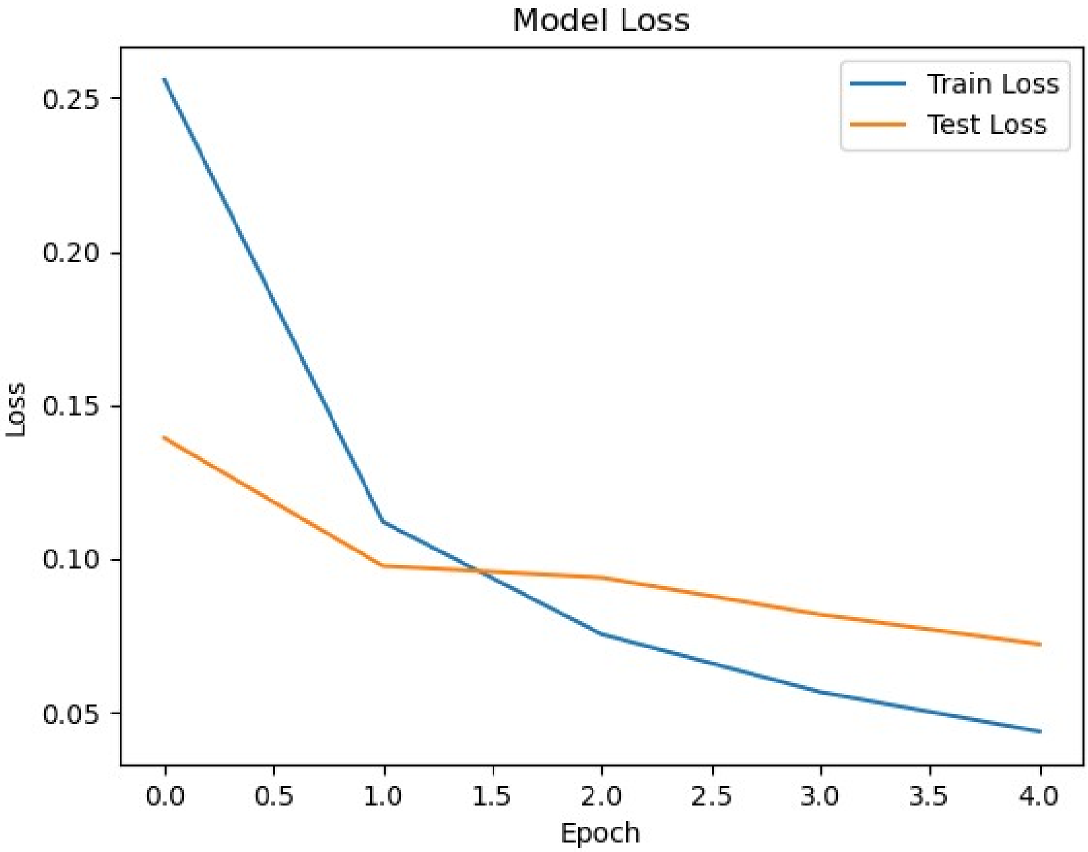
```python
# Import required libraries
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Step 1: Load the MNIST dataset
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()

# Normalize the pixel values to range [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0

# Step 2: Build a simple model
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),  # Flatten the 28x28 images
into 1D
    layers.Dense(128, activation='relu'),  # Dense layer with 128
neurons
    layers.Dense(10, activation='softmax')  # Output layer for 10
classes
])

# Step 3: Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Step 4: Train the model
history = model.fit(x_train, y_train, epochs=5,
validation_data=(x_test, y_test))

# Step 5: Visualize training results
# Accuracy plot
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Loss plot
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
# Step 6: Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test Accuracy: {test_acc:.4f}')

# Step 7: Predict on test data (optional)
predictions = model.predict(x_test)
print(f"Predicted class for first test image:
{tf.argmax(predictions[0]).numpy()}")
```

```
C:\Users\Niket Chauhan\OneDrive\Documents\Zoom\Lib\site-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
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 5ms/step - accuracy: 0.8768 - loss:
0.4317 - val_accuracy: 0.9577 - val_loss: 0.1392
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.9638 - loss:
0.1185 - val_accuracy: 0.9715 - val_loss: 0.0976
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.9787 - loss:
0.0712 - val_accuracy: 0.9713 - val_loss: 0.0938
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9830 - loss:
0.0549 - val_accuracy: 0.9751 - val_loss: 0.0817
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.9881 - loss:
0.0396 - val_accuracy: 0.9783 - val_loss: 0.0720
```


Model Accuracy
Train Accuracy
Test Accuracy
0.98
0.97
0.96
0.95
0.94
0.93
Accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch



```
313/313 - 1s - 4ms/step - accuracy: 0.9783 - loss: 0.0720
Test Accuracy: 0.9783
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Predicted class for first test image: 7

model.summary()

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ flatten (Flatten)               │ (None, 784)            │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense (Dense)                   │ (None, 128)            │
100,480 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_1 (Dense)                 │ (None, 10)             │
```

```
1,290 │
└──────────────────────────────────┴─────────────────────────┴──────────
───────┘

 Total params: 305,312 (1.16 MB)

 Trainable params: 101,770 (397.54 KB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 203,542 (795.09 KB)
```