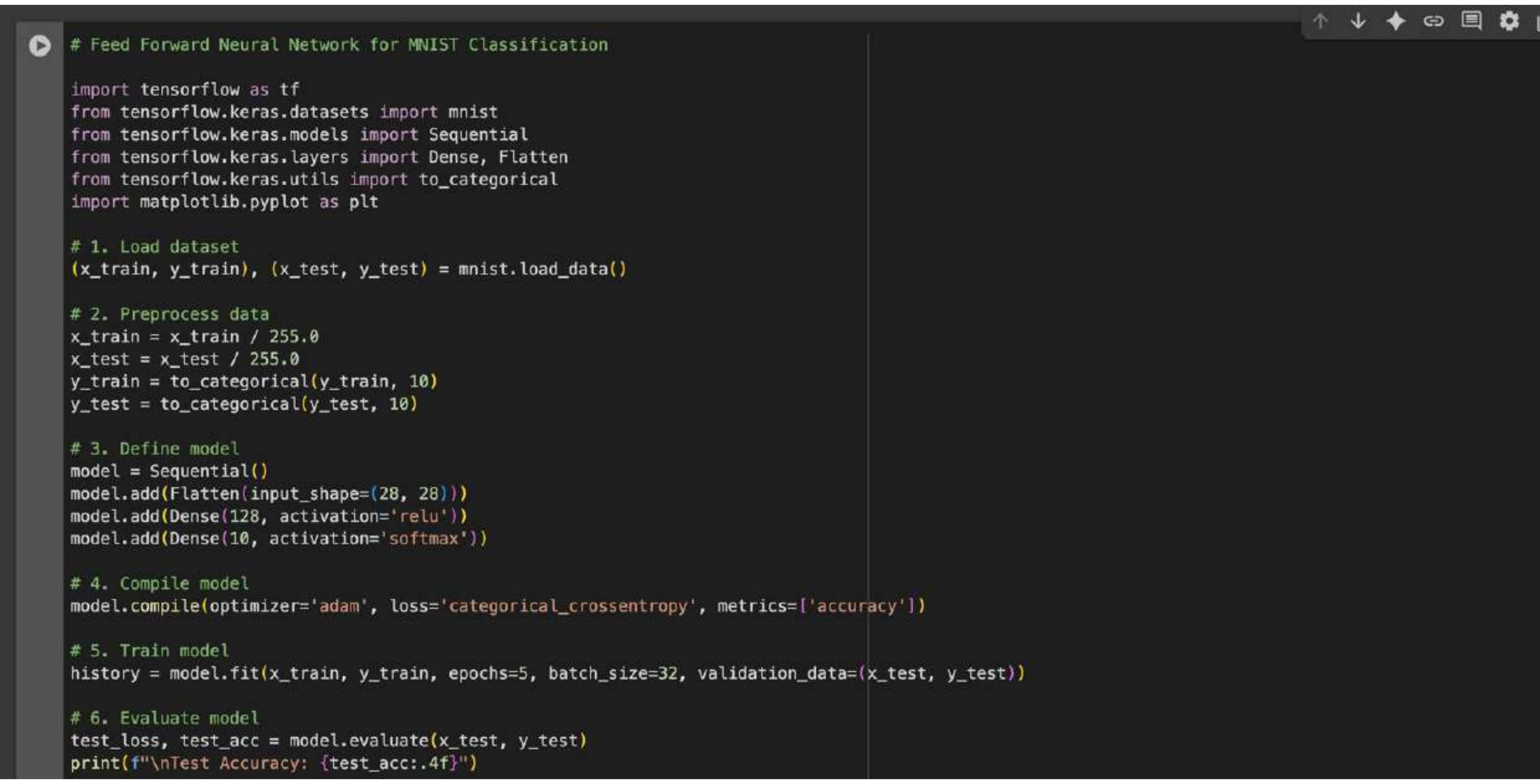
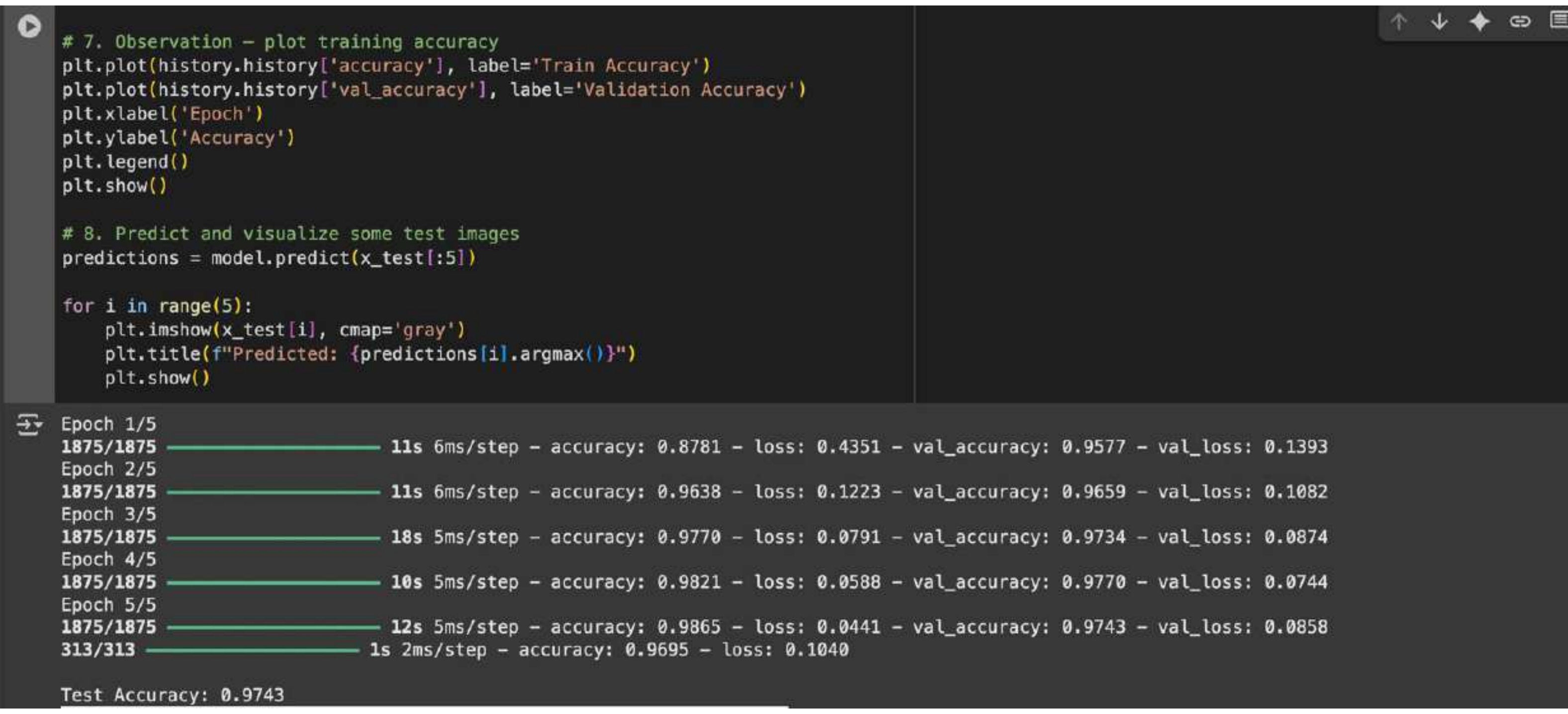
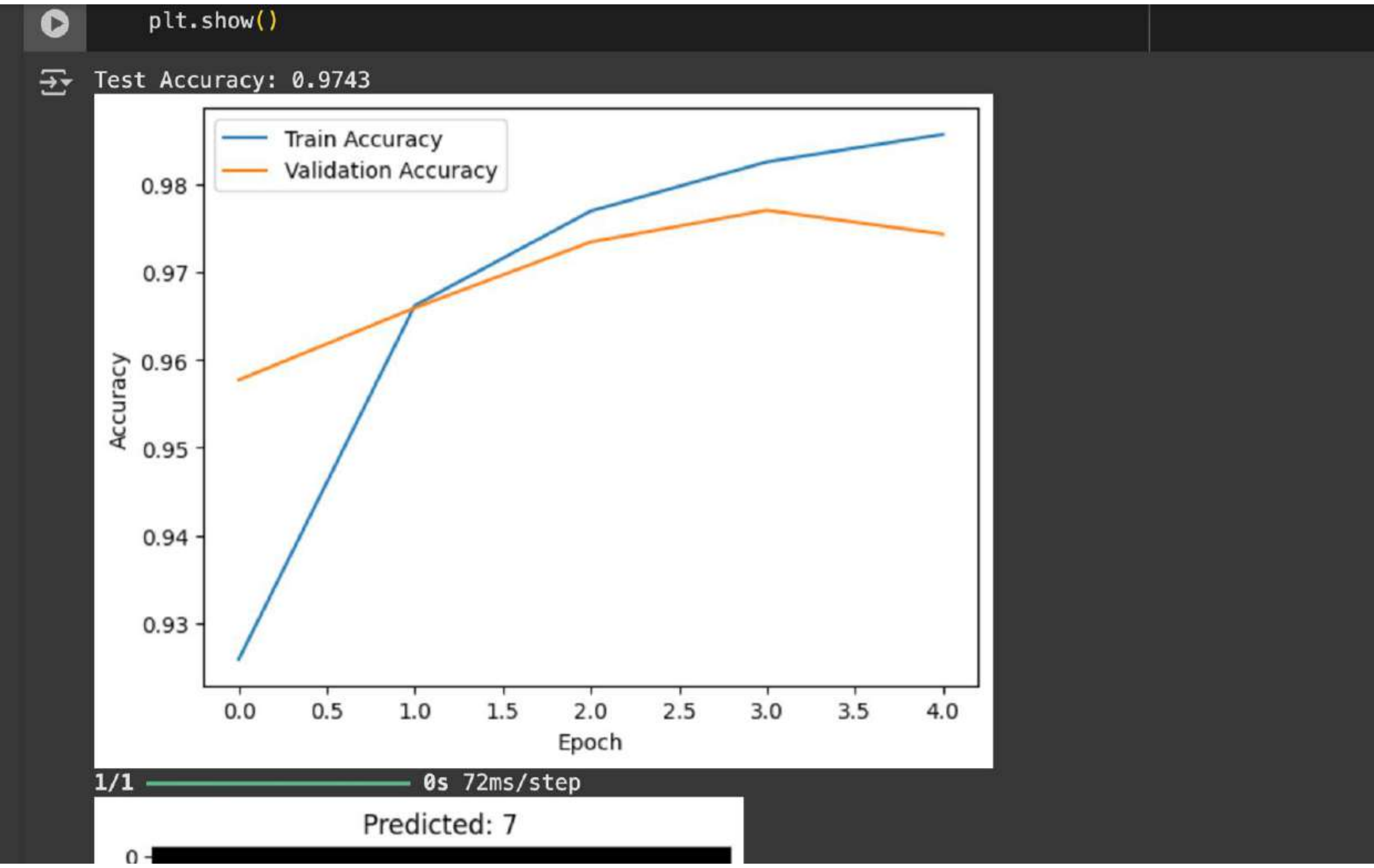


```python
# Feed Forward Neural Network for MNIST Classification

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

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

# 2. Preprocess data
x_train = x_train / 255.0
x_test = x_test / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# 3. Define model
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

# 5. Train model
history = model.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))

# 6. Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"\nTest Accuracy: {test_acc:.4f}")
```



```python
# 7. Observation - plot training accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# 8. Predict and visualize some test images
predictions = model.predict(x_test[:5])

for i in range(5):
    plt.imshow(x_test[i], cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}")
    plt.show()
```

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.8781 - loss: 0.4351 - val_accuracy: 0.9577 - val_loss: 0.1393
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.9638 - loss: 0.1223 - val_accuracy: 0.9659 - val_loss: 0.1082
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 18s 5ms/step - accuracy: 0.9770 - loss: 0.0791 - val_accuracy: 0.9734 - val_loss: 0.0874
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9821 - loss: 0.0588 - val_accuracy: 0.9770 - val_loss: 0.0744
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 5ms/step - accuracy: 0.9865 - loss: 0.0441 - val_accuracy: 0.9743 - val_loss: 0.0858
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9695 - loss: 0.1040

Test Accuracy: 0.9743
```



```
plt.show()
```

```
Test Accuracy: 0.9743
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
```

4. Build a sample feed forward neural network to recognise handwritten characters.

Aim:- To build a simple feed forward neural network to recognise handwritten characters.

Objective:-

1. To load and preprocess the MNIST dataset for neural network input.
2. To build feed forward neural network model with hidden Layers.
3. To train the model using stochostive gradient descent Optimizer and sparge categorical cross-entry loss.
4. Evaluate the trained model on test data and measure its accuracy.
5. To predict the class of given handwritten image.

Pseudocode:-

```
START:-
    Load MNIST dataset (training and testing data).
    ~~flatte~~ pattern each image from 28x28 to 784
                                          features.
    normalise pixel values to range [0,1]
create a sequential neural network.
Layer1: Dense (128 neurons, ReLU activation).
Layer 2: Dense (64 neurons, ReLU activation).
Output layer: Dense (10 neurons, softmax activation).
```

Compile model:-

optimizer = stathastic gradient descent

Loss = sparse categorical crossentropy

Metric = accuracy.

Train model on Training data for 5 epochs.

Evaluate model on testing data.

print test accuracy.

Observation:-

→ The loss decrease with each, showing that the model is learning.

→ Accuracy improves stedily during training.

Training:- X

epoch	Accuracy.	Loss.	
1	~~0.9929~~ 91.86%	~~0.0232~~	0.2801
2	~~0.9968~~ 96.65%	0.0128	0.1203
3	~~0.9976~~ 97.68%	0.0099	0.0807
4	~~0.9976~~ 98.21%	0.0088	0.0583
5	~~0.9987~~ 98.74%	0.0058	0.0424

Overall accuracy of the model on the entire dataset = ~~0.9828~~ 0.9754.

epoch	Training accuracy	Training loss	val-accuracy	validation loss.
1	0.8773	0.4370	0.9578	0.1380
2	0.9662	0.1170	0.9702	0.0966
3	0.9769	0.0790	0.9749	0.0805
4	0.9828	0.0564	0.9753	0.0790
5	0.9875	0.417	0.9754	0.0816.

Result:-

Sucessfully built a Simple feed forward neural. network to recognize handwritten characters.

[illegible] 14/8/25

ML Libraries here

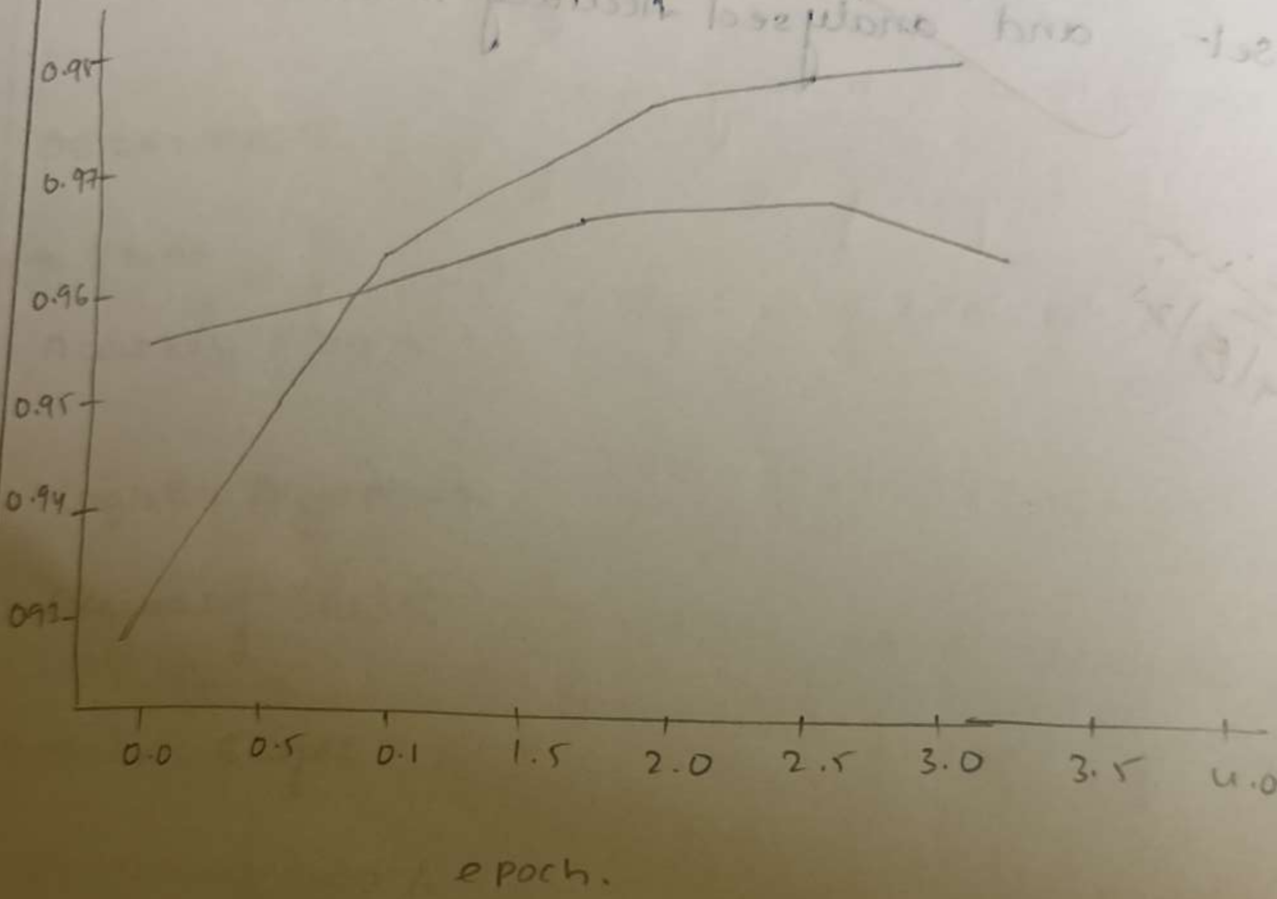
Tensorflow

pyTorch

so-no manual downloading

Feed forward.

goes input [illegible] output las


epoch.