Deep Learning Model

This notebook shows how to build a Convolutional Neural Network (CNN) to classify images from the Fashion MNIST dataset. The Fashion MNIST dataset consists of grayscale images of size 28x28 pixels belonging to 10 different clothing categories. We will use TensorFlow and Keras to build, train, and evaluate the model.

```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
```

```
# loading the dataset
(train_images, train_labels), (test_images, test_labels) = fashion_m

# Preprocess the data
# Normalize pixel values to be between 0 and 1
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-k
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-k
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-k
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-k
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Reshape the data to be compatible with the CNN (28, 28, 1)
train_images = train_images.reshape(train_images.shape[0], 28, 28,
test_images = test_images.reshape(test_images.shape[0], 28, 28, 1)
```

```
# one-hot encode the labels
train_labels = to_categorical(train_labels, num_classes=10)
test_labels = to_categorical(test_labels, num_classes=10)

# verify the shape of the data
print(f'Train images shape: {train_images.shape}')
print(f'Test images shape: {test_images.shape}')
print(f'Train labels shape: {train_labels.shape}')
print(f'Test labels shape: {test_labels.shape}')
```

```
Train images shape: (60000, 28, 28, 1)
Test images shape: (10000, 28, 28, 1)
Train labels shape: (60000, 10)
Test labels shape: (10000, 10)
```

Build the CNN Model

```
# build the CNN model
model = models.Sequential([
    # First convolutional layer
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 2
    layers.MaxPooling2D((2, 2)),
    # Second convolutional layer
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    # Third convolutional layer
    layers.Conv2D(64, (3, 3), activation='relu'),
    # Flatten the output
    layers.Flatten(),
    # Fully connected (dense) layer
    layers.Dense(64, activation='relu'),
    # Output layer with 10 units for 10 classes (Fashion MNIST has
    layers.Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutior
  super().__init__(activity_regularizer=activity_regularizer, **kwar
```

Compile and Train the Model

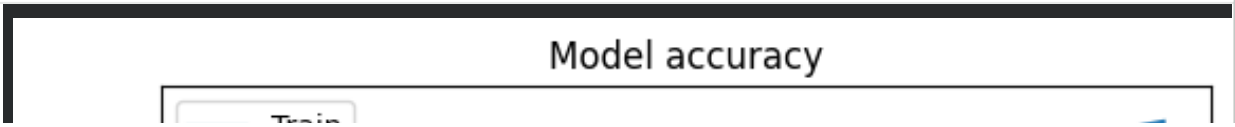
```
# compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
history = model.fit(train_images, train_labels, epochs=5, batch_siz
```

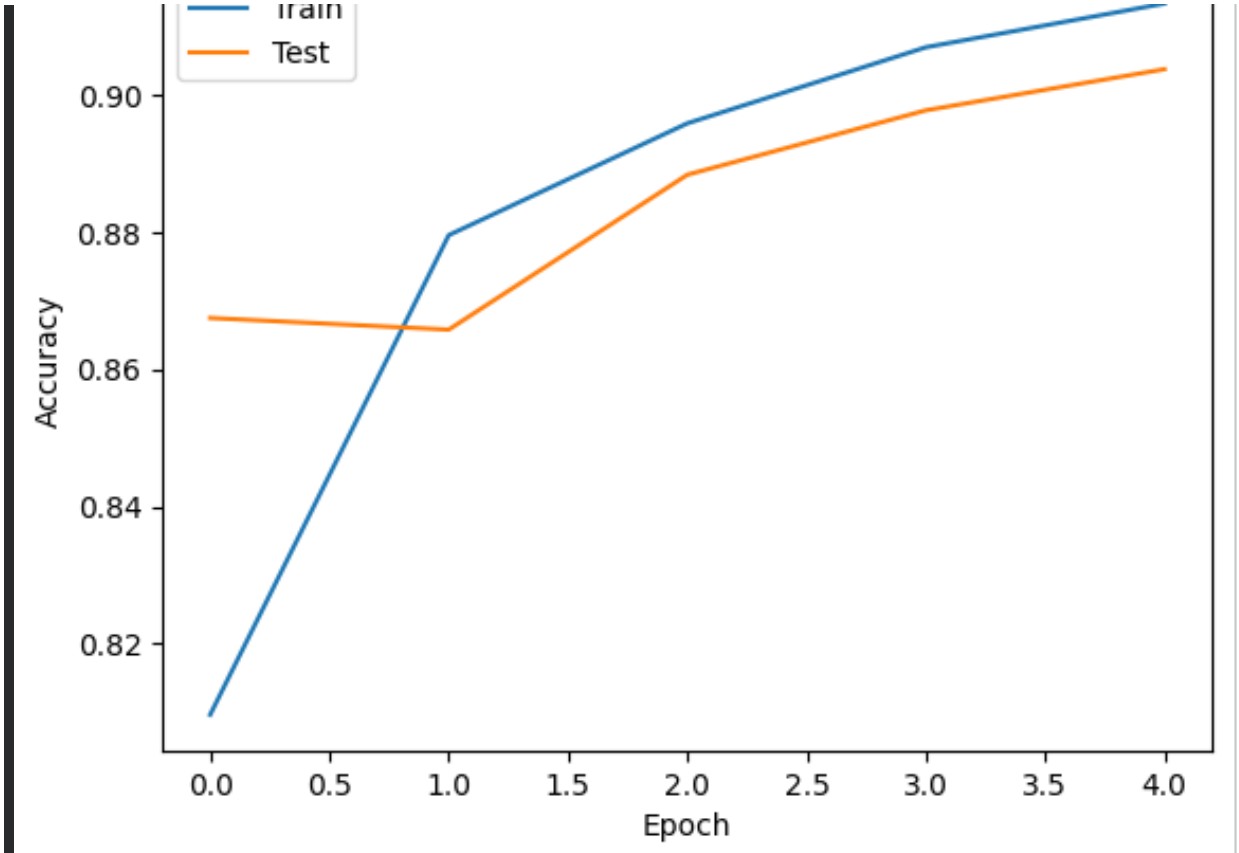
```
Epoch 1/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 51s 53ms/step - accuracy: 0.7346 - loss
Epoch 2/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 49s 52ms/step - accuracy: 0.8753 - loss
Epoch 3/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 87s 57ms/step - accuracy: 0.8941 - loss
Epoch 4/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 57s 61ms/step - accuracy: 0.9075 - loss
Epoch 5/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 58s 62ms/step - accuracy: 0.9114 - loss
```

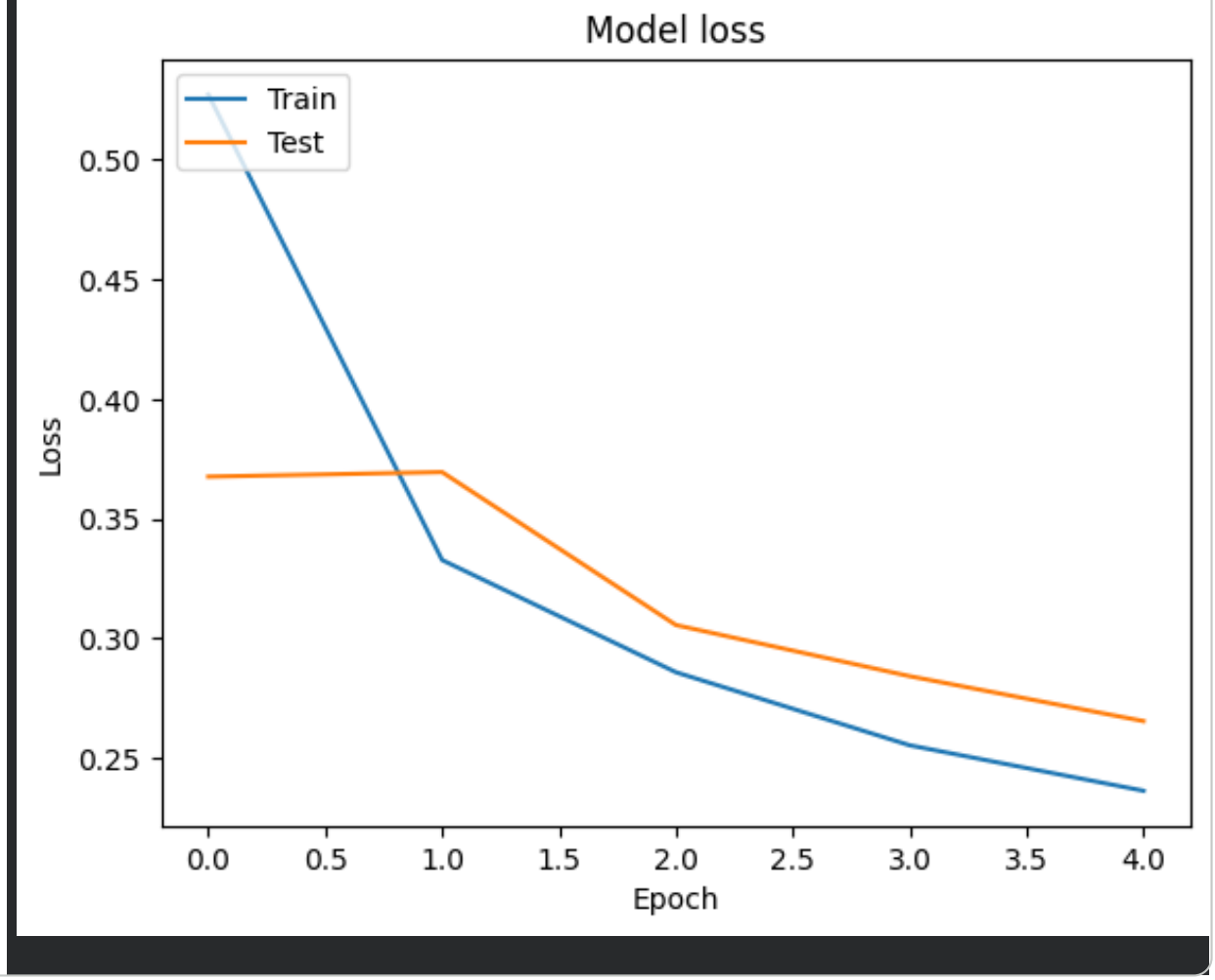
```
# evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verb
print(f'\nTest accuracy: {test_acc}')
```

```
313/313 - 3s - 9ms/step - accuracy: 0.9038 - loss: 0.2655

Test accuracy: 0.9038000106811523
```

```
# plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



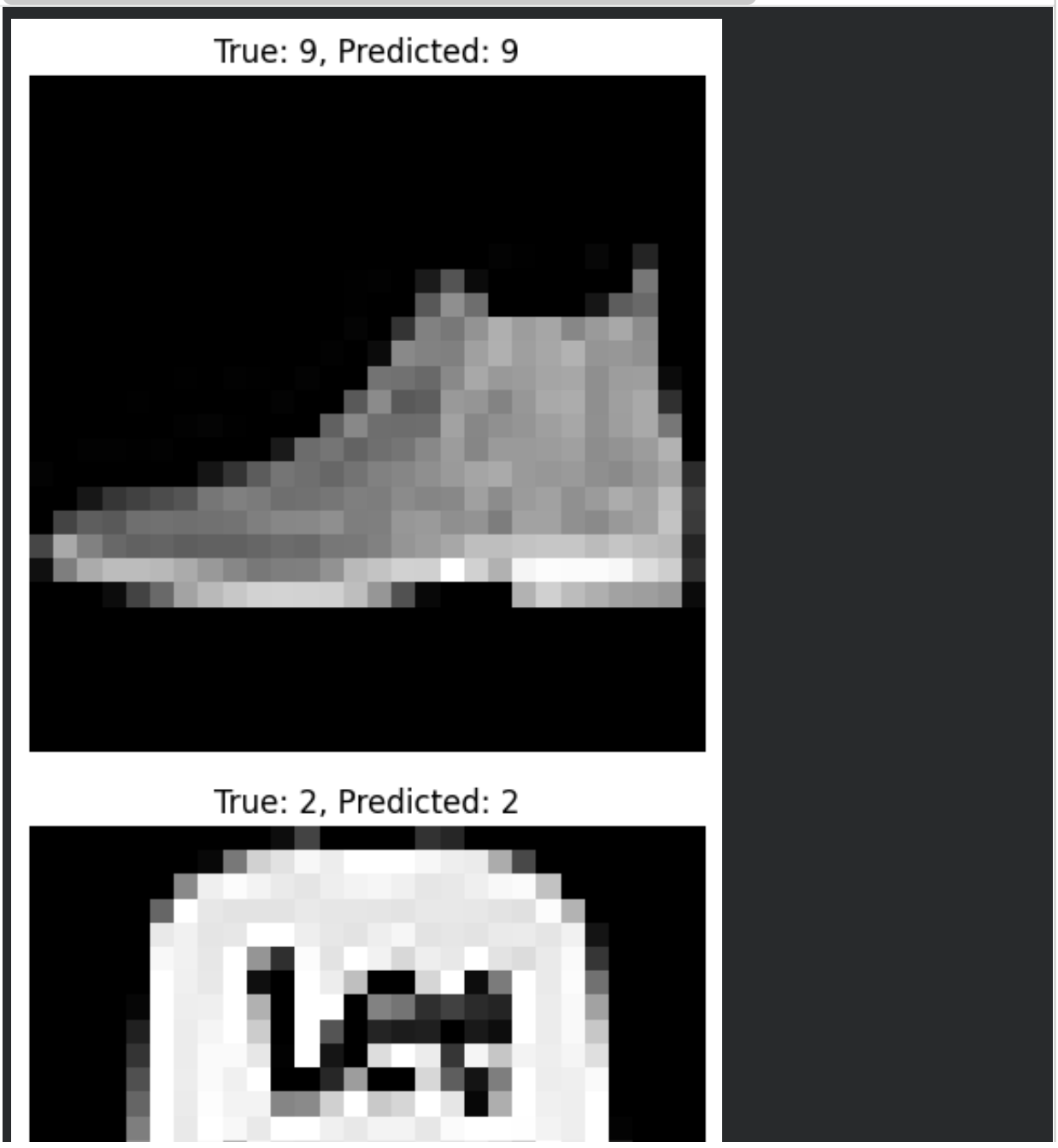

Train
Test
0.90
0.88
0.86
0.84
0.82
Accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Model loss
Train
Test
0.50
0.45
0.40
0.35
0.30
0.25
Loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

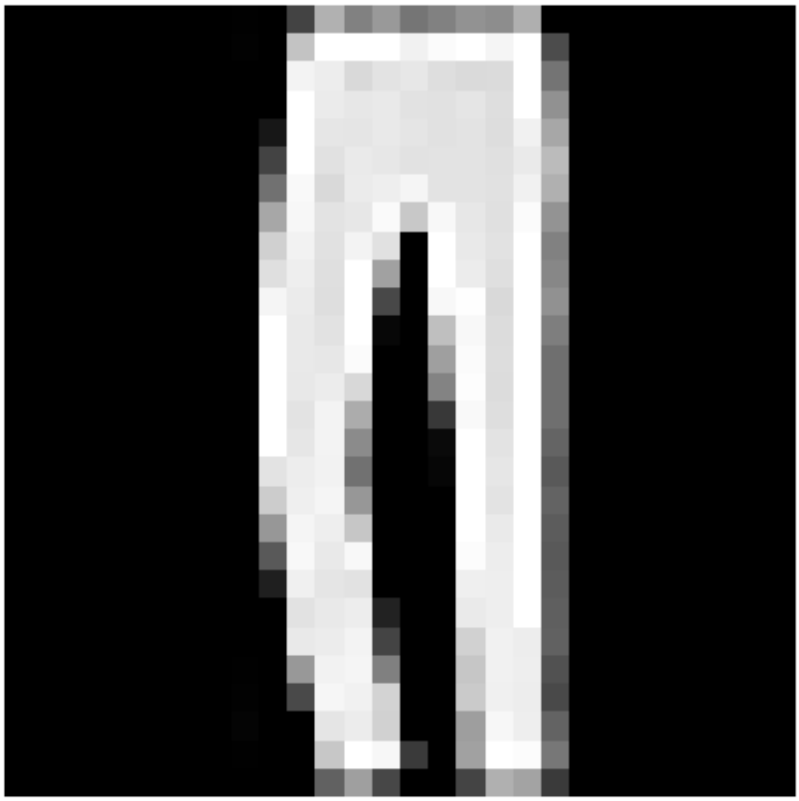
```
# make predictions
predictions = model.predict(test_images)
predicted_classes = tf.argmax(predictions, axis=1)
true_classes = tf.argmax(test_labels, axis=1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
```

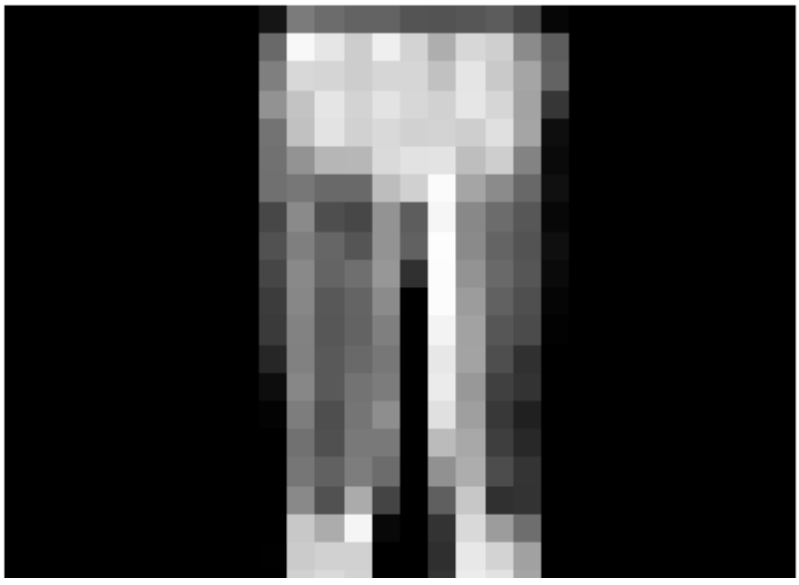
```
# display the first 5 images along with their predicted and actual
for i in range(5):
    plt.imshow(test_images[i].reshape(28, 28), cmap='gray')
    plt.title(f'True: {true_classes[i].numpy()}, Predicted: {predic
    plt.axis('off')
    plt.show()
```

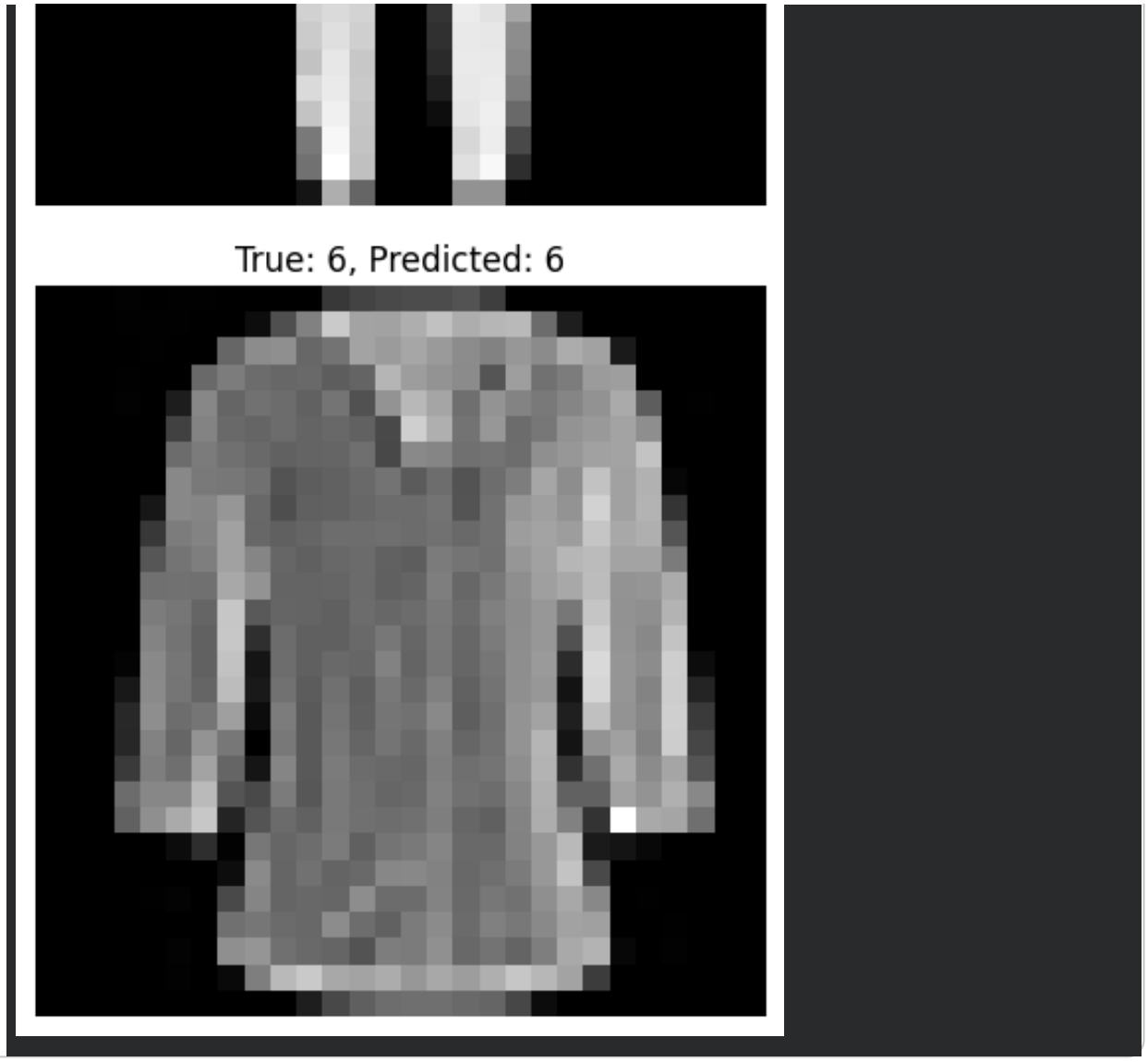


True: 1, Predicted: 1



True: 1, Predicted: 1




True: 6, Predicted: 6