Name : Harshada Mhaske

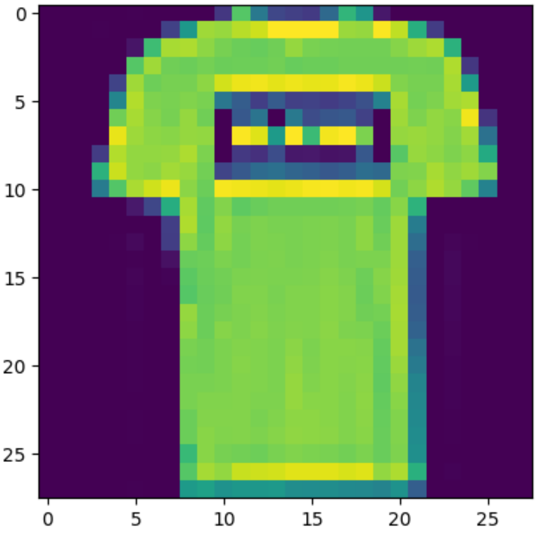
Div : B

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np
from tensorflow.keras import layers
```

```
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
```

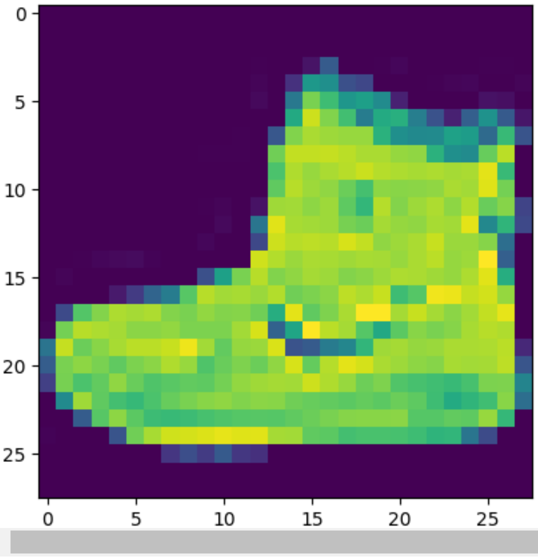
```
# There are 10 image classes in this dataset and each class has a mapping corresponding to the following labels:
#0 T-shirt/top #1 Trouser
#2 pullover
#3 Dress
#4 Coat
#5 sandals
#6 shirt
#7 sneaker
#8 bag
#9 ankle boot plt.imshow(x_train[1])
plt.imshow(x_train[1])
```

```
<matplotlib.image.AxesImage at 0x787050af9050>
```



```
plt.imshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x78703972e7d0>
```



```
# Normalize the input data to [0, 1]
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
# Reshape the input data to add a channel dimension (for CNNs)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

y_train = y_train.reshape(-1)
y_test = y_test.reshape(-1)



model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])


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

# Train the model
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 58s 30ms/step - accuracy: 0.7618 - loss: 0.6602 - val_accuracy: 0.8726 - val_loss: 0.3585
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 30ms/step - accuracy: 0.8812 - loss: 0.3288 - val_accuracy: 0.8845 - val_loss: 0.3171
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 30ms/step - accuracy: 0.8978 - loss: 0.2812 - val_accuracy: 0.9003 - val_loss: 0.2780
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 29ms/step - accuracy: 0.9126 - loss: 0.2405 - val_accuracy: 0.8893 - val_loss: 0.2970
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 55s 30ms/step - accuracy: 0.9194 - loss: 0.2181 - val_accuracy: 0.9008 - val_loss: 0.2735
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 30ms/step - accuracy: 0.9285 - loss: 0.1919 - val_accuracy: 0.9050 - val_loss: 0.2593
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 57s 30ms/step - accuracy: 0.9344 - loss: 0.1743 - val_accuracy: 0.9069 - val_loss: 0.2638
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 87s 33ms/step - accuracy: 0.9406 - loss: 0.1591 - val_accuracy: 0.9109 - val_loss: 0.2625
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 77s 31ms/step - accuracy: 0.9463 - loss: 0.1416 - val_accuracy: 0.9134 - val_loss: 0.2537
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 30ms/step - accuracy: 0.9538 - loss: 0.1271 - val_accuracy: 0.9133 - val_loss: 0.2682
```

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9116 - loss: 0.2923
Test accuracy: 0.9107999801635742
```