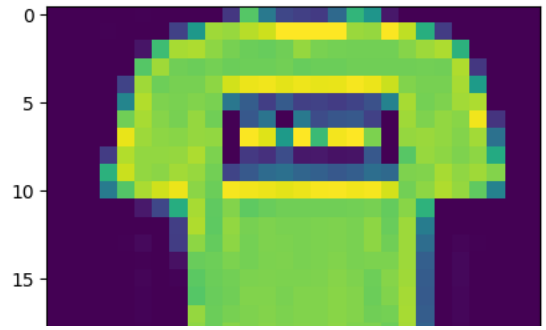
```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np


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


x_train, y_train, x_test, y_test
```

```
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              ...,

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8),
       array([9, 0, 0, ..., 3, 0, 5], dtype=uint8),
       array([[[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
```

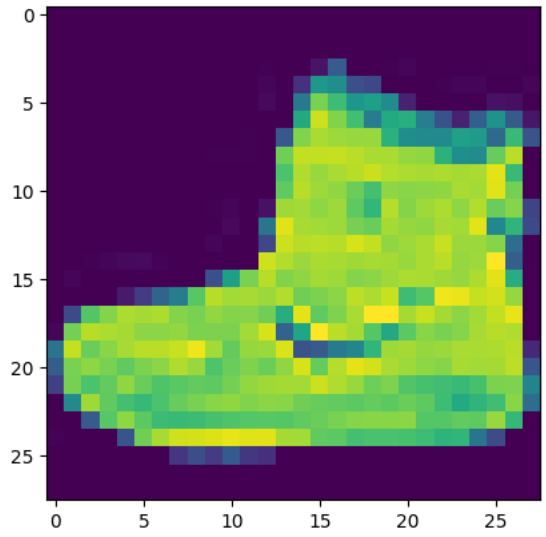
```
plt.imshow(x_train[1])
```

```
<matplotlib.image.AxesImage at 0x7f8974305690>
```




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

```
<matplotlib.image.AxesImage at 0x7f89446bc220>
```




```
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

```
x_train.shape
```

```
(60000, 28, 28)
```

```
x_test.shape
```

```
(10000, 28, 28)
```

```
y_train.shape
```

```
(60000,)
```

```
y_test.shape
```

```
(10000,)
```

```
model = keras.Sequential([
keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),

keras.layers.MaxPooling2D((2,2)),

keras.layers.Dropout(0.25),

keras.layers.Conv2D(64, (3,3), activation='relu'),

keras.layers.MaxPooling2D((2,2)),

keras.layers.Dropout(0.25),
keras.layers.Conv2D(128, (3,3), activation='relu'),
```

```
keras.layers.Flatten(),
keras.layers.Dense(128, activation='relu'),

keras.layers.Dropout(0.25),
keras.layers.Dense(10, activation='softmax')

])



model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 dropout_3 (Dropout)         (None, 13, 13, 32)        0

 conv2d_4 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 dropout_4 (Dropout)         (None, 5, 5, 64)          0

 conv2d_5 (Conv2D)           (None, 3, 3, 128)         73856

 flatten_1 (Flatten)         (None, 1152)              0

 dense_2 (Dense)             (None, 128)               147584

 dropout_5 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 241,546
Trainable params: 241,546
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

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

```
Epoch 1/5
1875/1875 [==============================] - 93s 49ms/step - loss: 0.2651 - accuracy: 0.9020 - val_loss: 0.2586 - val_accuracy: 0.9044
Epoch 2/5
1875/1875 [==============================] - 91s 49ms/step - loss: 0.2561 - accuracy: 0.9038 - val_loss: 0.2624 - val_accuracy: 0.9064
Epoch 3/5
1875/1875 [==============================] - 82s 44ms/step - loss: 0.2455 - accuracy: 0.9082 - val_loss: 0.2454 - val_accuracy: 0.9078
Epoch 4/5
1875/1875 [==============================] - 80s 43ms/step - loss: 0.2362 - accuracy: 0.9107 - val_loss: 0.2525 - val_accuracy: 0.9096
Epoch 5/5
1875/1875 [==============================] - 82s 44ms/step - loss: 0.2297 - accuracy: 0.9139 - val_loss: 0.2609 - val_accuracy: 0.9066
313/313 [==============================] - 3s 10ms/step - loss: 0.2609 - accuracy: 0.9066
Test accuracy: 0.9065999984741211
```

✓ 7m 29s completed at 2:04 PM