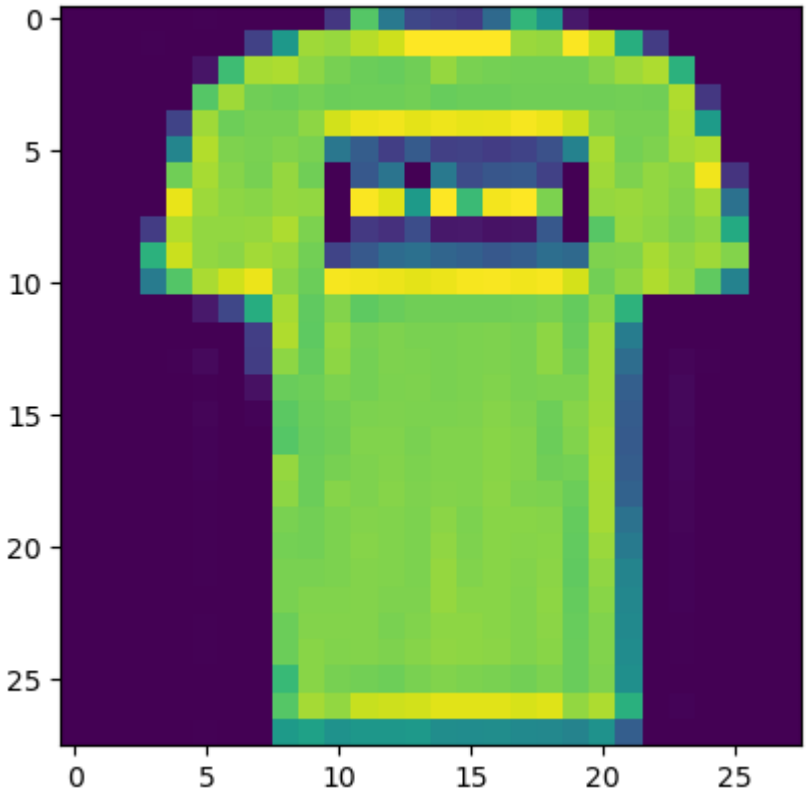
In [1]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np

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

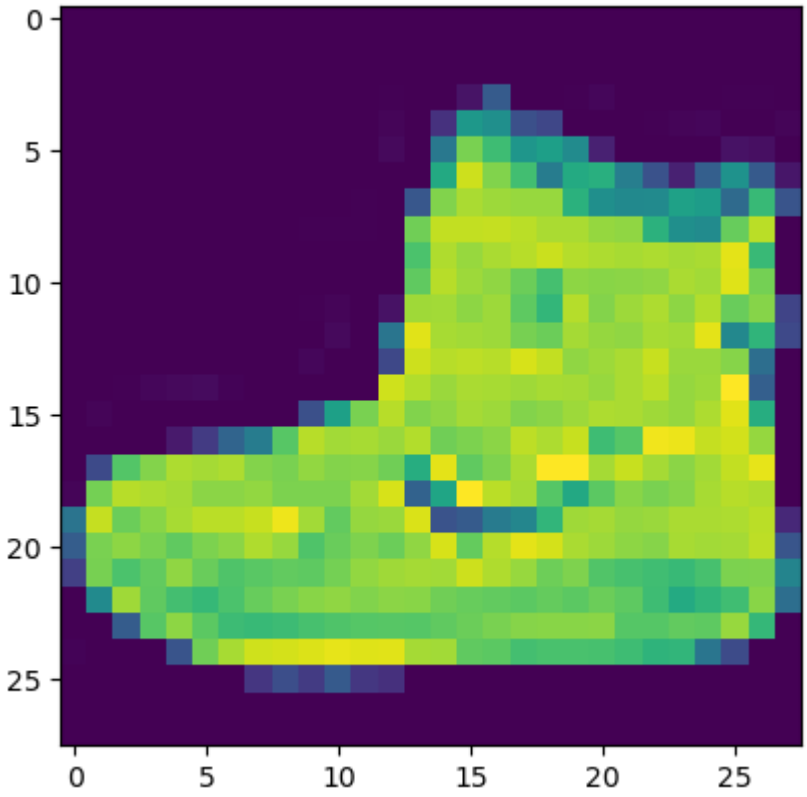
In [2]:
```
plt.imshow(x_train[1])
```

Out[2]: <matplotlib.image.AxesImage at 0x289c8d21ac0>



In [3]:
```
plt.imshow(x_train[0])
```

Out[3]: <matplotlib.image.AxesImage at 0x289dd859130>



In [4]:
```
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
```

In [5]:
```
x_train.shape
```

Out[5]: (60000, 28, 28, 1)

In [6]:
```
x_test.shape
```

Out[6]: (10000, 28, 28, 1)

In [7]:
```
y_train.shape
```

Out[7]: (60000,)

In [8]:
```
y_test.shape
```

Out[8]: (10000,)

In [9]:
```
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')

])
```

```
C:\Users\sukhad\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: 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__(activity_regularizer=activity_regularizer, **kwargs)
```

In [10]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	
conv2d (Conv2D)	(None, 26, 26, 32)	
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	
dropout (Dropout)	(None, 13, 13, 32)	
conv2d_1 (Conv2D)	(None, 11, 11, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	
dropout_1 (Dropout)	(None, 5, 5, 64)	
conv2d_2 (Conv2D)	(None, 3, 3, 128)	
flatten (Flatten)	(None, 1152)	
dense (Dense)	(None, 128)	
dropout_2 (Dropout)	(None, 128)	
dense_1 (Dense)	(None, 10)	

Total params: 241,546 (943.54 KB)

Trainable params: 241,546 (943.54 KB)

Non-trainable params: 0 (0.00 B)

In [11]:
```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=

history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 25s 12ms/step - accuracy: 0.7113 - loss: 0.7817 -
val_accuracy: 0.8620 - val_loss: 0.3839
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.8610 - loss: 0.3799 -
val_accuracy: 0.8826 - val_loss: 0.3094
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 11ms/step - accuracy: 0.8805 - loss: 0.3234 -
val_accuracy: 0.8916 - val_loss: 0.2937
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.8935 - loss: 0.2888 -
val_accuracy: 0.8998 - val_loss: 0.2732
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.8984 - loss: 0.2756 -
val_accuracy: 0.9009 - val_loss: 0.2687
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.9016 - loss: 0.2660 -
val_accuracy: 0.9088 - val_loss: 0.2566
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 23s 12ms/step - accuracy: 0.9060 - loss: 0.2477 -
val_accuracy: 0.9058 - val_loss: 0.2547
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 11ms/step - accuracy: 0.9097 - loss: 0.2392 -
val_accuracy: 0.9080 - val_loss: 0.2512
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.9095 - loss: 0.2410 -
val_accuracy: 0.9082 - val_loss: 0.2571
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 11ms/step - accuracy: 0.9154 - loss: 0.2249 -
val_accuracy: 0.9117 - val_loss: 0.2529
```

In [12]:
```
test_loss, test_acc = model.evaluate(x_test, y_test)

print('Test accuracy:', test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9118 - loss: 0.2600
Test accuracy: 0.9117000102996826
```

In [ ]:
```

```