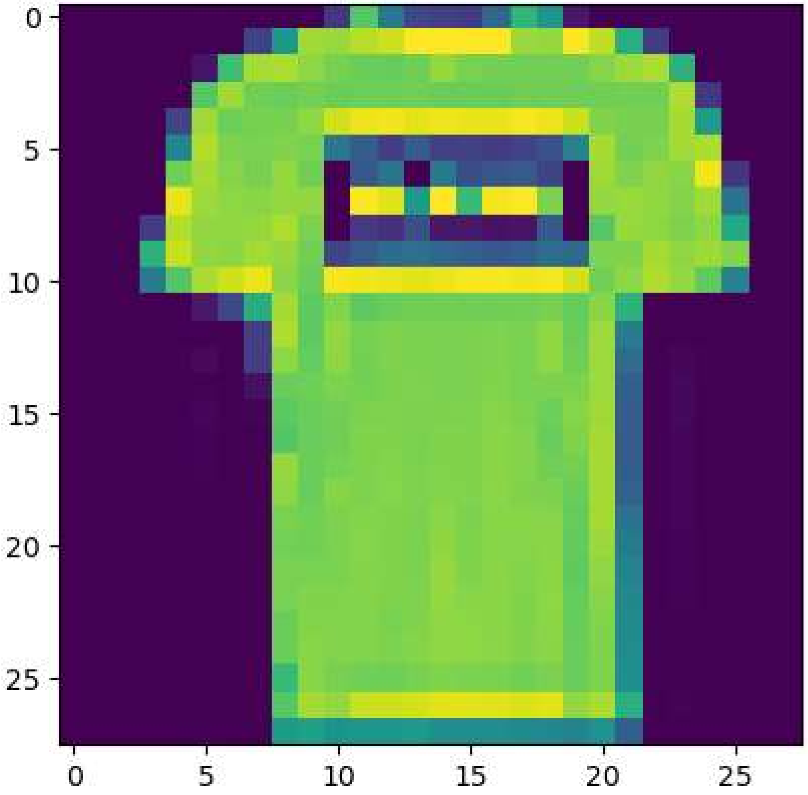
In [1]:

```
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(

# There are 10 image classes in this dataset and each class has a mapping corr

#0 T-shirt/top
#1 Trouser
#2 pullover
#3 Dress
#4 Coat
#5 sandals
#6 shirt
#7 sneaker
#8 bag
#9 ankle boot

# https://ml-course.github.io/master/09%20-%20Convolutional%20Neural%20Network
```

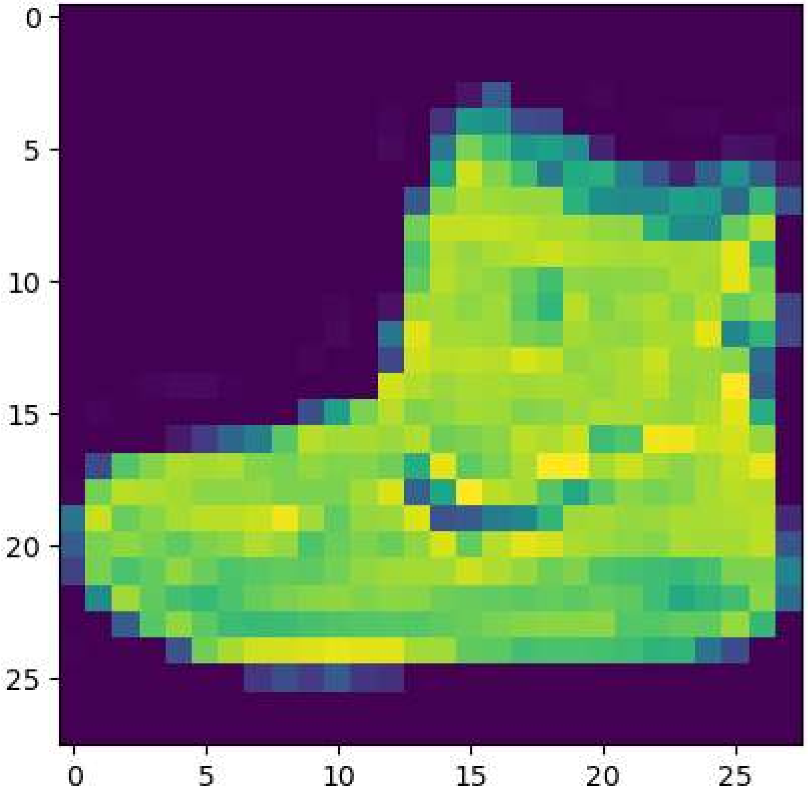
```
WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cr
oss_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cros
s_entropy instead.

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf
-keras-datasets/train-labels-idx1-ubyte.gz)
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf
-keras-datasets/train-images-idx3-ubyte.gz)
26421880/26421880 [==============================] - 5s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz)
5148/5148 [==============================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz)
4422102/4422102 [==============================] - 1s 0us/step
```

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

Out[2]:
```
<matplotlib.image.AxesImage at 0x22d91216e10>
```



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

Out[3]: <matplotlib.image.AxesImage at 0x22d91357750>
```



```
In [4]: # Next, we will preprocess the data by scaling the pixel values to be between

        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)

        # 28, 28 comes from width, height, 1 comes from the number of channels
        # -1 means that the length in that dimension is inferred.
        # This is done based on the constraint that the number of elements in an ndarr

        # each image is a row vector (784 elements) and there are lots of such rows (L
```

```
In [5]: # converting the training_images array to 4 dimensional array with sizes 60000

        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]:
```
# We will use a convolutional neural network (CNN) to classify the fashion ite
# The CNN will consist of multiple convolutional layers followed by max poolin
# dropout, and dense layers. Here is the code for the model:

model = keras.Sequential([
    keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    # 32 filters (default), randomly initialized
    # 3*3 is Size of Filter
    # 28,28,1 size of Input Image
    # No zero-padding: every output 2 pixels less in every dimension
    # in Paramter shwon 320 is value of weights: (3x3 filter weights + 32 bias
    # 32*3*3=288(Total)+32(bias)= 320


    keras.layers.MaxPooling2D((2,2)),
    # It shown 13 * 13 size image with 32 channel or filter or depth.

    keras.layers.Dropout(0.25),
    # Reduce Overfitting of Training sample drop out 25% Neuron

    keras.layers.Conv2D(64, (3,3), activation='relu'),
    # Deeper layers use 64 filters
    # 3*3 is Size of Filter
    # Observe how the input image on 28x28x1 is transformed to a 3x3x64 featur
    # 13(Size)-3(Filter Size )+1(bias)=11 Size for Width and Height with 64 De
    # in Paramter shwon 18496 is value of weights: (3x3 filter weights + 64 bi
    # 64*3*3=576+1=577*32 + 32(bias)=18496

    keras.layers.MaxPooling2D((2,2)),
    # It shown 5 * 5 size image with 64 channel or filter or depth.

    keras.layers.Dropout(0.25),

    keras.layers.Conv2D(128, (3,3), activation='relu'),
    # Deeper layers use 128 filters
    # 3*3 is Size of Filter
    # Observe how the input image on 28x28x1 is transformed to a 3x3x128 featu
    # It show 5(Size)-3(Filter Size )+1(bias)=3 Size for Width and Height with
    # 128*3*3=1152+1=1153*64 + 64(bias)= 73856

    # To classify the images, we still need a Dense and Softmax layer.
    # We need to flatten the 3x3x128 feature map to a vector of size 1152
    # https://medium.com/@iamvarman/how-to-calculate-the-number-of-parameters-

    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    # 128 Size of Node in Dense Layer
    # 1152*128 = 147584

    keras.layers.Dropout(0.25),
    keras.layers.Dense(10, activation='softmax')
    # 10 Size of Node another Dense Layer
    # 128*10+10 bias= 1290
])
```

```
WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is dep
recated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.
max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [10]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

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

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 dropout_1 (Dropout)         (None, 5, 5, 64)          0

 conv2d_2 (Conv2D)           (None, 3, 3, 128)         73856

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 128)               147584

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 241546 (943.54 KB)
Trainable params: 241546 (943.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [11]:
```
# Compile and Train the Model
# After defining the model, we will compile it and train it on the training da

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metric

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

# 1875 is a number of batches. By default batches contain 32 samles.60000 / 32
```

```
WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimi
zer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/10
WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTens
orValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue inste
ad.

WARNING:tensorflow:From C:\Users\Sakshi's PC\AppData\Roaming\Python\Python311
\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executin
g_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_
eagerly_outside_functions instead.

1875/1875 [==============================] - 29s 15ms/step - loss: 0.5689 - a
ccuracy: 0.7886 - val_loss: 0.3816 - val_accuracy: 0.8611
Epoch 2/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.3728 - a
ccuracy: 0.8630 - val_loss: 0.3128 - val_accuracy: 0.8855
Epoch 3/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.3241 - a
ccuracy: 0.8802 - val_loss: 0.3029 - val_accuracy: 0.8871
Epoch 4/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.2972 - a
ccuracy: 0.8905 - val_loss: 0.2729 - val_accuracy: 0.9000
Epoch 5/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.2802 - a
ccuracy: 0.8961 - val_loss: 0.2773 - val_accuracy: 0.8941
Epoch 6/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.2645 - a
ccuracy: 0.9014 - val_loss: 0.2655 - val_accuracy: 0.9037
Epoch 7/10
1875/1875 [==============================] - 27s 15ms/step - loss: 0.2521 - a
ccuracy: 0.9061 - val_loss: 0.2607 - val_accuracy: 0.9004
Epoch 8/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.2422 - a
ccuracy: 0.9095 - val_loss: 0.2558 - val_accuracy: 0.9069
Epoch 9/10
1875/1875 [==============================] - 27s 15ms/step - loss: 0.2324 - a
ccuracy: 0.9112 - val_loss: 0.2429 - val_accuracy: 0.9091
Epoch 10/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.2293 - a
ccuracy: 0.9137 - val_loss: 0.2543 - val_accuracy: 0.9088
```

In [12]:

```
# Finally, we will evaluate the performance of the model on the test data.

test_loss, test_acc = model.evaluate(x_test, y_test)

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

```
313/313 [==============================] - 2s 6ms/step - loss: 0.2543 - accur
acy: 0.9088
Test accuracy: 0.9088000059127808
```

In [13]:

```
pred = model.predict(x_test)
pred
```

```
313/313 [==============================] - 2s 5ms/step
```

Out[13]:

```
array([[4.72687944e-10, 4.70370028e-11, 2.51631355e-10, ...,
        4.28298517e-05, 4.35415731e-10, 9.99956131e-01],
       [7.08246034e-07, 2.29262217e-11, 9.99829650e-01, ...,
        1.00810624e-22, 1.14914656e-09, 2.48267197e-19],
       [7.83738519e-10, 1.00000000e+00, 3.29792235e-11, ...,
        9.74168265e-21, 4.09122953e-12, 1.89905846e-21],
       ...,
       [1.65549727e-10, 2.30544395e-16, 4.22858797e-11, ...,
        4.64661165e-12, 1.00000000e+00, 4.75911153e-14],
       [1.09682628e-11, 1.00000000e+00, 1.97278977e-12, ...,
        2.06001718e-21, 1.84952395e-13, 4.25730571e-20],
       [7.98275960e-06, 1.06374330e-07, 7.01506133e-06, ...,
        9.77163836e-02, 4.87984908e-05, 2.99475185e-04]], dtype=float32)
```

In [ ]: