

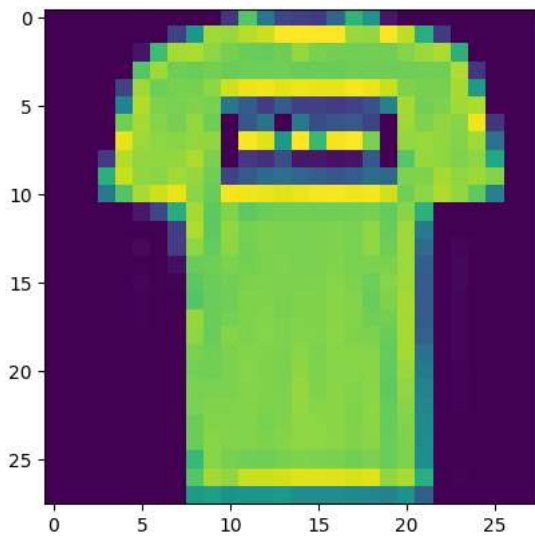
```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np
```

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

```
Downloading data from 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-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ————— 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ————— 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ————— 1s 0us/step
```

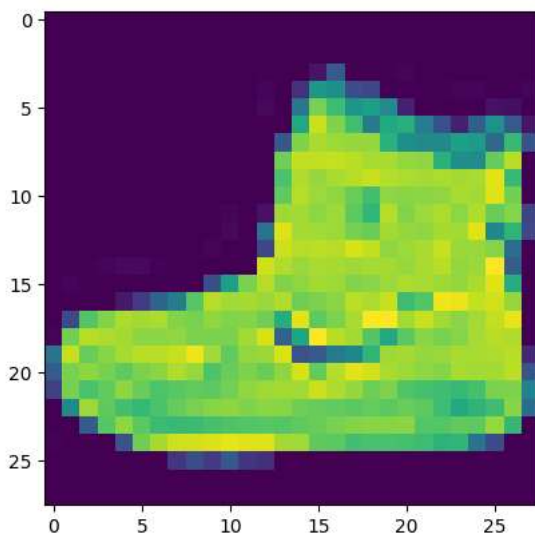
```
plt.imshow(x_train[1])
```

```
<matplotlib.image.AxesImage at 0x7ae0f119f4d0>
```



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

```
<matplotlib.image.AxesImage at 0x7ae0f0efcbd0>
```



```
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)
```

```


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 filters
# 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%
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 feature map
# 13(Size)-3(Filter Size )+1(bias)=11 Size for Width and Height with 64 Depth or filter or
# in Paramter shwon 18496 is value of weights: (3x3 filter weights + 64 bias) * 64 filters
# 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 feature map
# It show 5(Size)-3(Filter Size )+1(bias)=3 Size for Width and Height with 64 Depth or filtter or
# 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
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
])

```

 /usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. It has been deprecated. Use `keras.Input` objects to specify the input shape of the layer instead. (Parameter: 'input_shape')

```
model.summary()
```

 Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73,856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 128)	147,584
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 241,546 (943.54 KB)

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

```
Epoch 1/10
1875/1875 ————— 77s 40ms/step - accuracy: 0.7179 - loss: 0.7617 - val_accuracy: 0.8674 - val_loss: 0.3664
Epoch 2/10
1875/1875 ————— 80s 39ms/step - accuracy: 0.8625 - loss: 0.3694 - val_accuracy: 0.8810 - val_loss: 0.3189
Epoch 3/10
1875/1875 ————— 81s 38ms/step - accuracy: 0.8827 - loss: 0.3239 - val_accuracy: 0.8954 - val_loss: 0.2878
Epoch 4/10
1875/1875 ————— 80s 38ms/step - accuracy: 0.8879 - loss: 0.2998 - val_accuracy: 0.8991 - val_loss: 0.2808
Epoch 5/10
1875/1875 ————— 83s 38ms/step - accuracy: 0.8974 - loss: 0.2708 - val_accuracy: 0.8986 - val_loss: 0.2744
Epoch 6/10
1875/1875 ————— 82s 39ms/step - accuracy: 0.9035 - loss: 0.2533 - val_accuracy: 0.9062 - val_loss: 0.2564
Epoch 7/10
1875/1875 ————— 72s 38ms/step - accuracy: 0.9046 - loss: 0.2526 - val_accuracy: 0.9069 - val_loss: 0.2579
Epoch 8/10
1875/1875 ————— 84s 40ms/step - accuracy: 0.9118 - loss: 0.2388 - val_accuracy: 0.9077 - val_loss: 0.2536
Epoch 9/10
1875/1875 ————— 77s 37ms/step - accuracy: 0.9144 - loss: 0.2302 - val_accuracy: 0.9094 - val_loss: 0.2607
Epoch 10/10
1875/1875 ————— 85s 39ms/step - accuracy: 0.9149 - loss: 0.2278 - val_accuracy: 0.9104 - val_loss: 0.2447
```

```
# Evaluate the model on the test data
_, test_acc = model.evaluate(x_test, y_test, verbose=0) # verbose=0 for no output during evaluation
```

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

```
Test accuracy: 0.9103999733924866
```