In [1]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

```
2.4.1
```

In [3]:

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images,train_labels),(test_images,test_labels)=fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-l
abels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-i
mages-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-la
bels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-im
ages-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

In [4]:

```
# 0 top 1 trouser 2 pullover 3 dress 4 coat 5 sandal 6 shirt 7 sneaker 8 bag 9 ankle boo
t
```

In [5]:

```
class_names = ['top','trouser','pullover','dress','coat','sandal','shirt','sneaker','bag
','ankle boot']
```

In [7]:

```
train_images.shape
```

Out[7]:

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

In [8]:

```
len(train_labels)
```

Out[8]:

```
60000
```

In [9]:

```
train_labels
```

Out[9]:

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

In [10]:

```
test_images.shape
```

Out[10]:

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

In [11]:

```
len(test_labels)
```

Out[11]:

```
10000
```

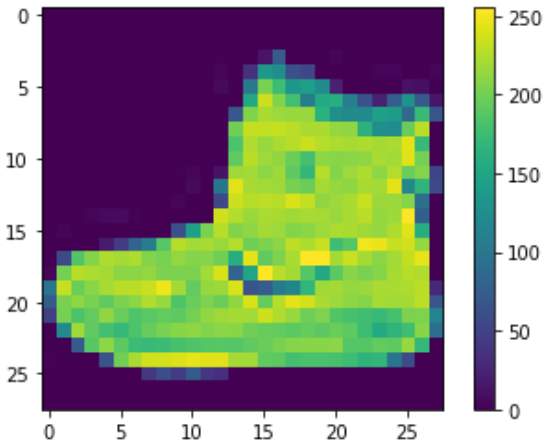
In [12]:

```
print(test_labels)
```

```
[9 2 1 ... 8 1 5]
```

In [13]:

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```

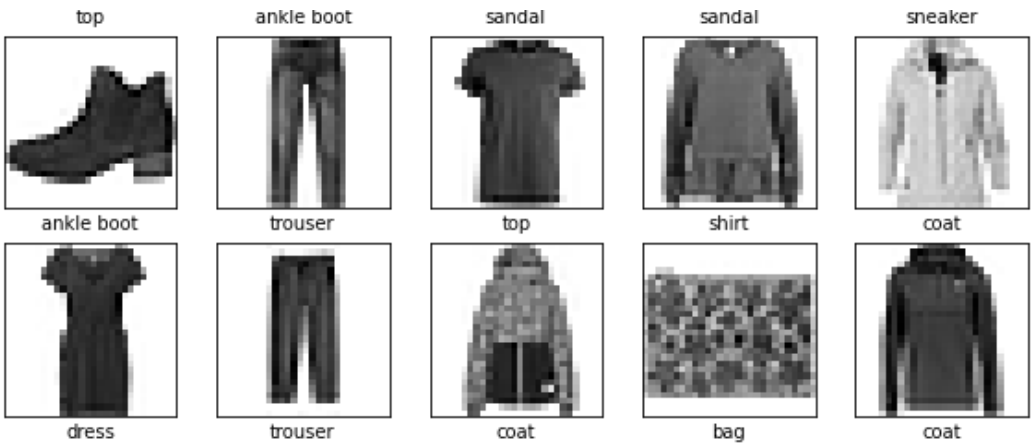


In [15]:

```
train_images=train_images/255.0
test_images=test_images/255.0

plt.figure(figsize=(10,10))

for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(train_images[i],cmap=plt.cm.binary)
  plt.xlabel(class_names[train_labels[i]])
plt.show()
```



In [18]:

```
model = tf.keras.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                             tf.keras.layers.Dense(128,activation='relu'),
                             tf.keras.layers.Dense(10)])
```

In [20]:

```
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
ogits=True),metrics=['accuracy'])
```

In [21]:

```
model.fit(train_images,train_labels,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 1.5271 - accuracy: 0.524
6
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6766 - accuracy: 0.756
1
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5815 - accuracy: 0.788
9
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5344 - accuracy: 0.809
2
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5065 - accuracy: 0.820
0
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4826 - accuracy: 0.829
7
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4622 - accuracy: 0.835
9
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4570 - accuracy: 0.841
0
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4363 - accuracy: 0.846
1
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4280 - accuracy: 0.845
8
```

Out[21]:

```
<tensorflow.python.keras.callbacks.History at 0x7fc87996d190>
```

In [22]:

```
test_loss,test_acc=model.evaluate(test_images,test_labels,verbose=2)
print('\nTest Accuracy : ',test_acc)
```

```
313/313 - 0s - loss: 0.4566 - accuracy: 0.8354
```

```
Test Accuracy :  0.8353999853134155
```

In [23]:

```
probability_model=tf.keras.Sequential([model,tf.keras.layers.Softmax()])
```

In [24]:

```
prediction=probability_model.predict(test_images)
prediction[0]
```

Out[24]:

```
array([4.2278185e-07, 1.2860928e-08, 5.1697789e-06, 4.5121728e-06,
       4.4703684e-06, 1.4013626e-01, 1.1814831e-05, 2.9970965e-01,
       5.0052893e-03, 5.5512244e-01], dtype=float32)
```

In [25]:

```
np.argmax(prediction[0])
```

Out[25]:

```
9
```

In [35]:

```

```

In [ ]:

```

```