Practical 5 - Dhruv Patel BDA 17162121014

In [2]:

```python
import tensorflow as tf
from tensorflow import keras


import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:

```python
dataset = keras.datasets.fashion_mnist

(train_images,train_labels),(test_images,test_labels) = dataset.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz)
32768/29515 [=================================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz)
26427392/26421880 [==============================] - 3s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz)
8192/5148 [==============================================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz)
4423680/4422102 [==============================] - 1s 0us/step
```

Loading the dataset returns four NumPy arrays:

The train_images and train_labels arrays are the training set—the data the model uses to learn. The model is tested against the test set, the test_images, and test_labels arrays. The images are 28x28 NumPy arrays, with pixel values ranging from 0 to 255. The labels are an array of integers, ranging from 0 to 9. These correspond to the class of clothing the image represents:

Label Class

0 T-shirt/top

1 Trouser

2 Pullover

3 Dress

4 Coat

5 Sandal

6 Shirt

7 Sneaker

8 Bag

9 Ankle boot

Each image is mapped to a single label. Since the class names are not included with the dataset, store them here to use later when plotting the images:

In [7]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

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

Out[8]:
```
(60000, 28, 28)
```

In [9]:
```
test_images.shape
```

Out[9]:
```
(10000, 28, 28)
```

In [10]:
```
train_labels.shape
```

Out[10]:
```
(60000,)
```

In [11]:
```
test_labels.shape
```

Out[11]:
```
(10000,)
```

In [12]:
```
train_labels
```

Out[12]:
```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

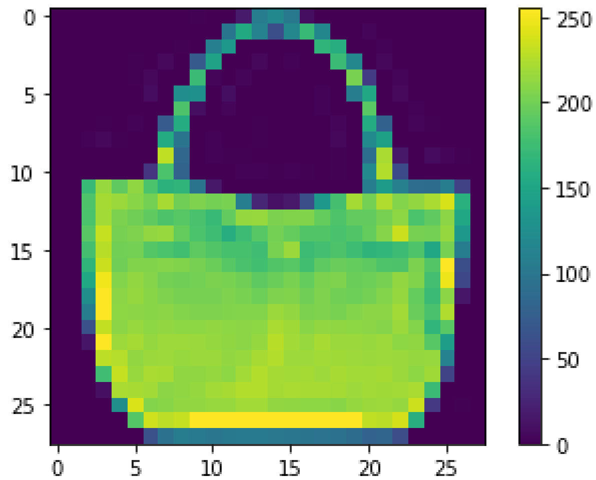
In [20]:
```
len(train_images) , len(test_images) , len(test_labels) ,len(train_labels)
```

Out[20]:
```
(60000, 10000, 10000, 60000)
```

Preprocess the data

In [19]:

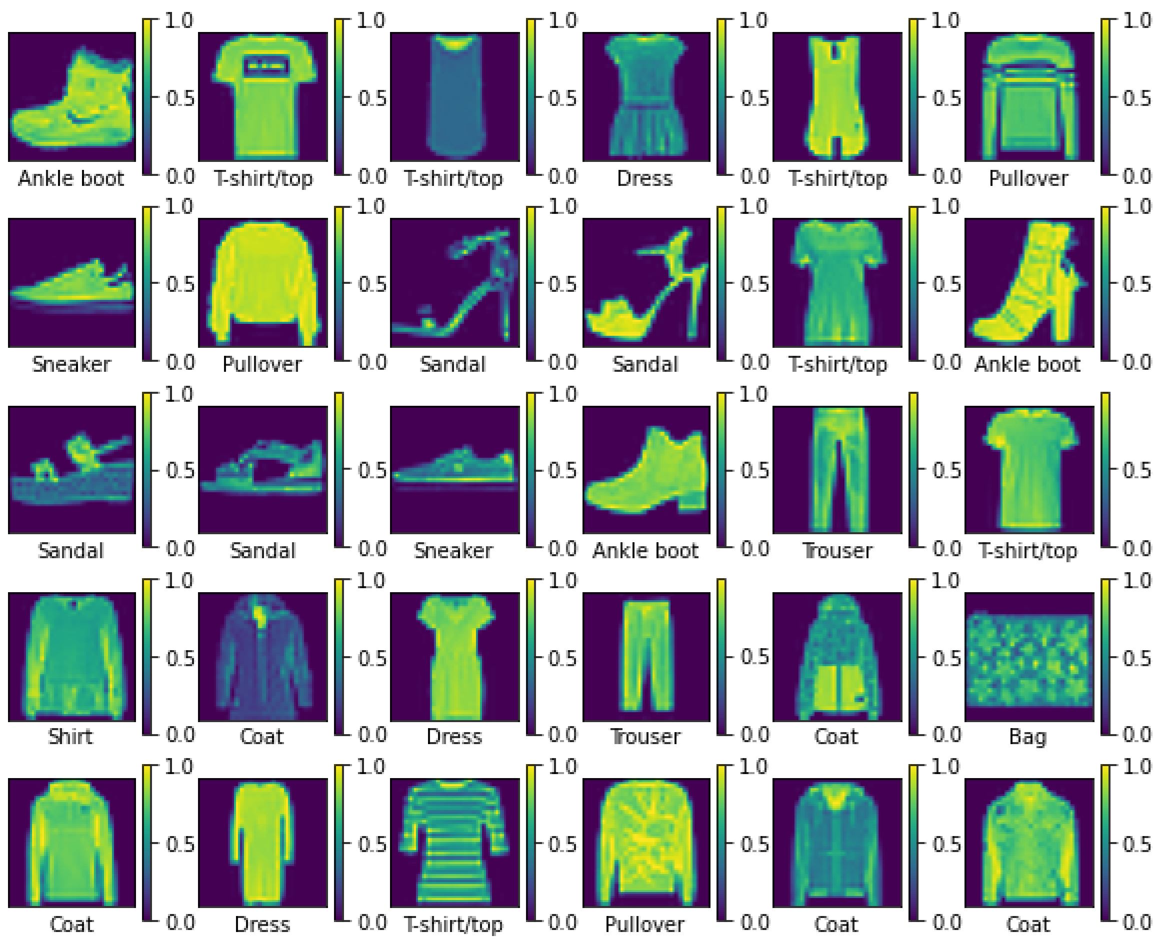
```
plt.figure()
plt.imshow(train_images[100])
plt.colorbar()
plt.grid(False)
plt.show()
```



In [21]:

```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

In [31]:
```
plt.figure(figsize=(10,10))
for i in range(30):
    plt.subplot(6,6,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
#     plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.imshow(train_images[i])
    plt.colorbar()
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```



Build the model

In [33]:
```
#using keras we are now building model

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

In [34]:
```
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossen
                metrics=['accuracy'])
```

In [36]:

```
model.fit(train_images,train_labels,epochs=20,use_multiprocessing=True) ## w
```

```
Epoch 1/20
1875/1875 [==============================] - 1s 734us/step - loss: 0.2245 - acc
uracy: 0.9148
Epoch 2/20
1875/1875 [==============================] - 1s 734us/step - loss: 0.2142 - acc
uracy: 0.9194
Epoch 3/20
1875/1875 [==============================] - 1s 725us/step - loss: 0.2081 - acc
uracy: 0.9208
Epoch 4/20
1875/1875 [==============================] - 1s 731us/step - loss: 0.2022 - acc
uracy: 0.9230
Epoch 5/20
1875/1875 [==============================] - 1s 735us/step - loss: 0.1941 - acc
uracy: 0.9247
Epoch 6/20
1875/1875 [==============================] - 1s 730us/step - loss: 0.1896 - acc
uracy: 0.9268
Epoch 7/20
1875/1875 [==============================] - 1s 740us/step - loss: 0.1828 - acc
uracy: 0.9298
Epoch 8/20
1875/1875 [==============================] - 1s 729us/step - loss: 0.1791 - acc
uracy: 0.9315
Epoch 9/20
1875/1875 [==============================] - 1s 736us/step - loss: 0.1754 - acc
uracy: 0.9341
Epoch 10/20
1875/1875 [==============================] - 1s 721us/step - loss: 0.1670 - acc
uracy: 0.9361
Epoch 11/20
1875/1875 [==============================] - 1s 729us/step - loss: 0.1645 - acc
uracy: 0.9371
Epoch 12/20
1875/1875 [==============================] - 1s 727us/step - loss: 0.1593 - acc
uracy: 0.9394
Epoch 13/20
1875/1875 [==============================] - 1s 730us/step - loss: 0.1574 - acc
uracy: 0.9390
Epoch 14/20
1875/1875 [==============================] - 1s 711us/step - loss: 0.1526 - acc
uracy: 0.9413
Epoch 15/20
1875/1875 [==============================] - 1s 715us/step - loss: 0.1476 - acc
uracy: 0.9431
Epoch 16/20
1875/1875 [==============================] - 1s 734us/step - loss: 0.1469 - acc
uracy: 0.9427
Epoch 17/20
1875/1875 [==============================] - 1s 726us/step - loss: 0.1402 - acc
uracy: 0.9458
Epoch 18/20
1875/1875 [==============================] - 1s 742us/step - loss: 0.1389 - acc
uracy: 0.9461
Epoch 19/20
```

```
1875/1875 [==============================] - 1s 719us/step - loss: 0.1336 - acc
uracy: 0.9484
Epoch 20/20
1875/1875 [==============================] - 1s 733us/step - loss: 0.1328 - acc
uracy: 0.9489
```

Out[36]:
```
<tensorflow.python.keras.callbacks.History at 0x263f05fd130>
```

In [38]:
```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
313/313 - 0s - loss: 0.4475 - accuracy: 0.8906

Test accuracy: 0.8906000256538391
```

In [39]:
```
model.save("fashion_mnist.h5")
```

Make Prediction

In [40]:
```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
```

In [41]:
```
predictions = probability_model.predict(test_images)
```

In [42]:
```
predictions[0]
```

Out[42]:
```
array([1.2450453e-10, 8.7283697e-10, 2.5722284e-11, 2.7918209e-13,
       6.0904712e-09, 7.6097108e-06, 3.3953999e-11, 4.3500785e-04,
       1.2557068e-12, 9.9955732e-01], dtype=float32)
```

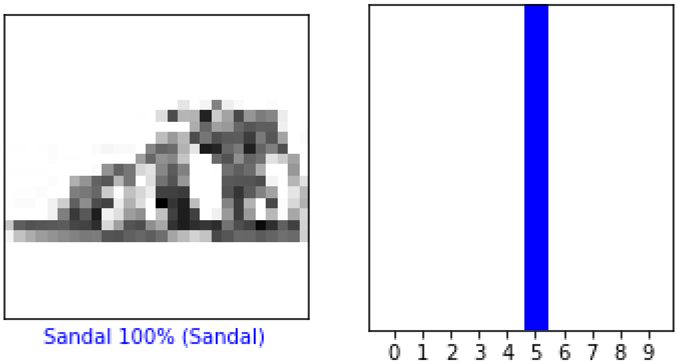
In [59]:
```
for i in range(20):
    print("Actual label for image ",i ," : ",test_labels[i],end=" and ")
    print("Predicted label for image ",i,": ",np.argmax(predictions[i]))
```

```
Actual label for image  0  :  9 and Predicted label for image  0 :  9
Actual label for image  1  :  2 and Predicted label for image  1 :  2
Actual label for image  2  :  1 and Predicted label for image  2 :  1
Actual label for image  3  :  1 and Predicted label for image  3 :  1
Actual label for image  4  :  6 and Predicted label for image  4 :  6
Actual label for image  5  :  1 and Predicted label for image  5 :  1
Actual label for image  6  :  4 and Predicted label for image  6 :  4
Actual label for image  7  :  6 and Predicted label for image  7 :  6
Actual label for image  8  :  5 and Predicted label for image  8 :  5
Actual label for image  9  :  7 and Predicted label for image  9 :  7
Actual label for image  10  :  4 and Predicted label for image  10 :  4
Actual label for image  11  :  5 and Predicted label for image  11 :  5
Actual label for image  12  :  7 and Predicted label for image  12 :  5
Actual label for image  13  :  3 and Predicted label for image  13 :  3
Actual label for image  14  :  4 and Predicted label for image  14 :  4
Actual label for image  15  :  1 and Predicted label for image  15 :  1
Actual label for image  16  :  2 and Predicted label for image  16 :  2
Actual label for image  17  :  4 and Predicted label for image  17 :  2
Actual label for image  18  :  8 and Predicted label for image  18 :  8
Actual label for image  19  :  0 and Predicted label for image  19 :  0
```

In [60]:
```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

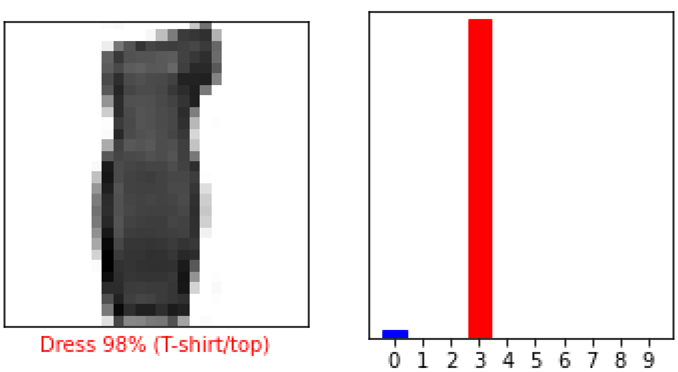
In [66]:

```
i = 11
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



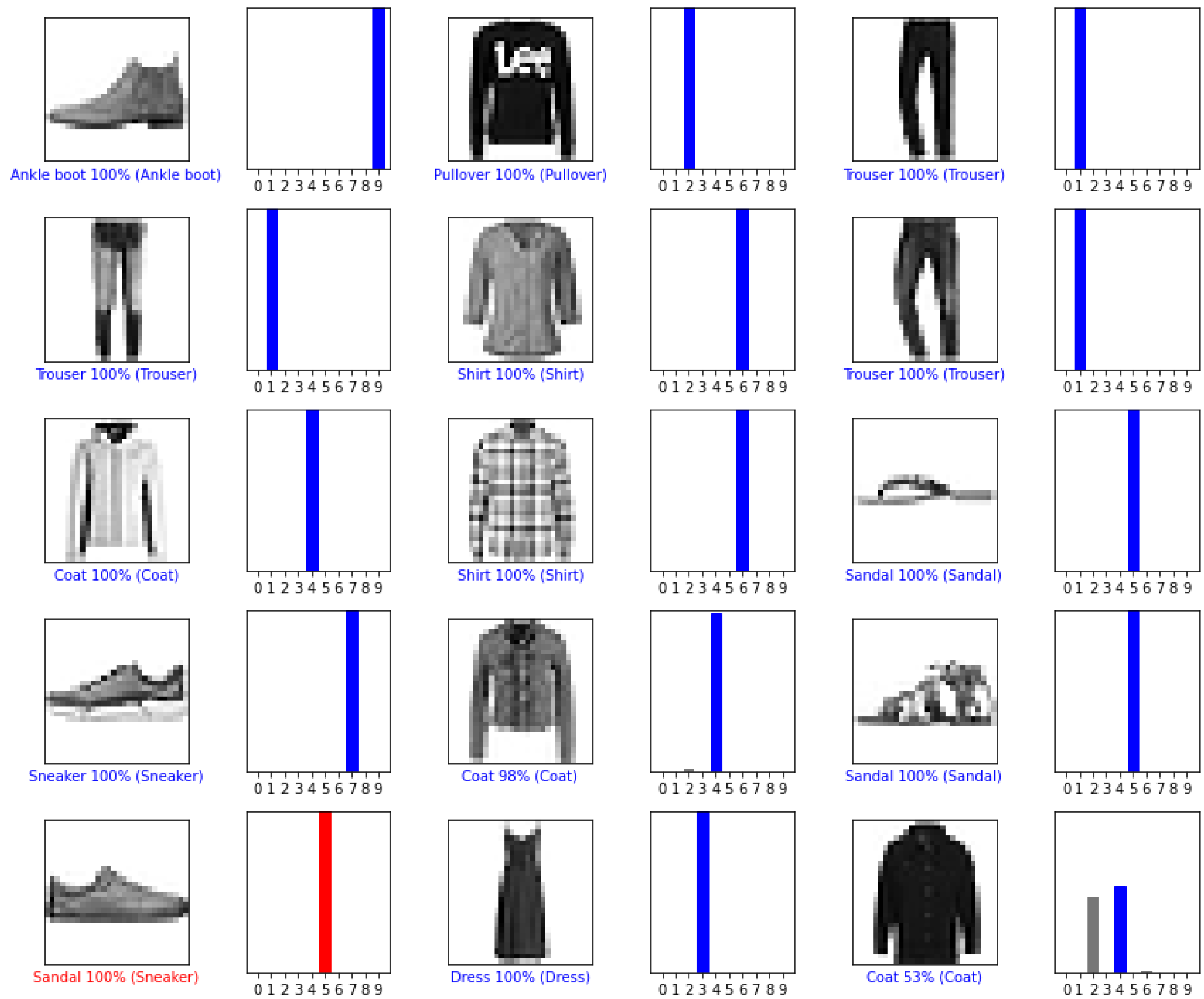
In [69]:

```
i = 1111
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



In [70]:

```python
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```



Testing on trained model

In [86]:
```
# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)
```

```
(28, 28)
```

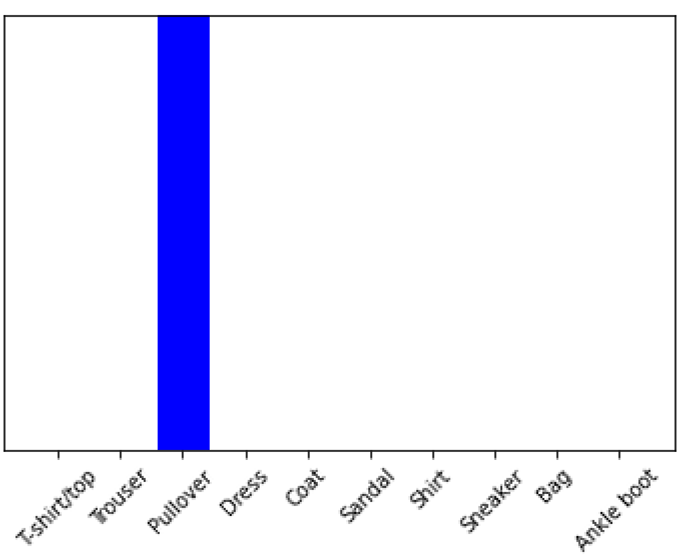
In [87]:
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

```
(1, 28, 28)
```

In [88]:
```
predictions_single = probability_model.predict(img)

print(predictions_single)
```

```
[[6.6959117e-08 1.5248392e-26 9.9999666e-01 1.4043638e-16 2.9703324e-06
  1.8136180e-27 2.9681675e-07 3.4467158e-18 3.6841552e-19 3.8420584e-31]]
```

In [89]:
```
plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



In [90]:
```
## thus we successfully trained an ANN for classification of fashion images
```