Problem Statement

Convolutional neural network (CNN):-Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories.

Basic classification: Classify images of clothing

```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.15.0
```

```
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-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 [==============================] - 0s 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 [==============================] - 0s 0us/step
```

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

Explore the data

```
train_images.shape
```

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

```
len(train_labels)
```

```
60000
```

```
train_labels
```

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

```
test_images.shape
```

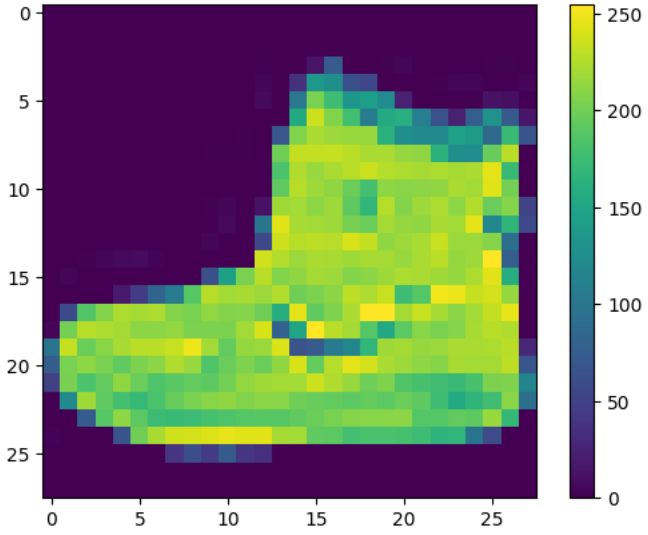
```
(10000, 28, 28)
```

```
len(test_labels)
```

```
10000
```

Preprocess the data

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




```
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()
```

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

Compile the model

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

```
model.fit(train_images, train_labels, epochs=15)
```

```
Epoch 1/15
1875/1875 [==============================] - 9s 4ms/step - loss: 0.4964 - accuracy: 0.8262
Epoch 2/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3759 - accuracy: 0.8634
Epoch 3/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3369 - accuracy: 0.8778
Epoch 4/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3128 - accuracy: 0.8853
Epoch 5/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2942 - accuracy: 0.8922
Epoch 6/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2805 - accuracy: 0.8966
Epoch 7/15
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2685 - accuracy: 0.8994
Epoch 8/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2584 - accuracy: 0.9041
Epoch 9/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2468 - accuracy: 0.9084
Epoch 10/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2372 - accuracy: 0.9116
Epoch 11/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2298 - accuracy: 0.9134
Epoch 12/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2241 - accuracy: 0.9160
Epoch 13/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2186 - accuracy: 0.9185
Epoch 14/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2089 - accuracy: 0.9211
Epoch 15/15
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2042 - accuracy: 0.9229
<keras.src.callbacks.History at 0x7d488e493c40>
```

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

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

```
313/313 - 1s - loss: 0.3456 - accuracy: 0.8851 - 985ms/epoch - 3ms/step

Test accuracy: 0.8851000070571899
```

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

```
predictions = probability_model.predict(test_images)
```

```
313/313 [==============================] - 1s 3ms/step
```

```
predictions[0]
```

```
array([1.0823580e-08, 6.5073597e-10, 1.8802895e-09, 2.6200671e-12,
       7.8096467e-11, 9.5620775e-04, 2.7143497e-07, 1.5706883e-03,
       2.3816295e-08, 9.9747282e-01], dtype=float32)
```

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

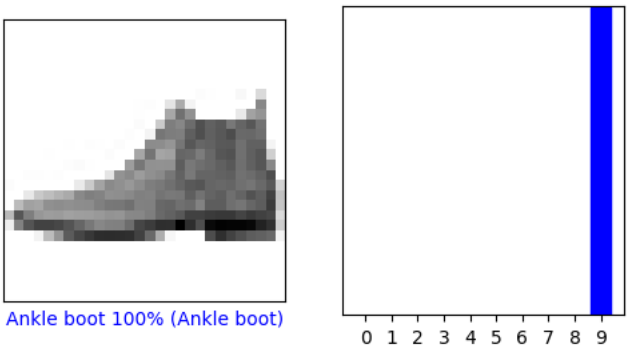
```
9
```

```
test_labels[0]
```

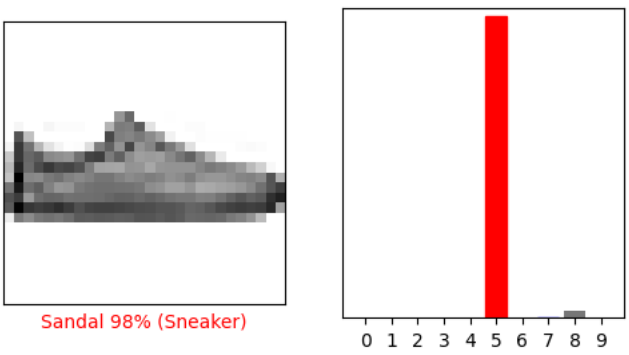
```
9
```

```
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')
```

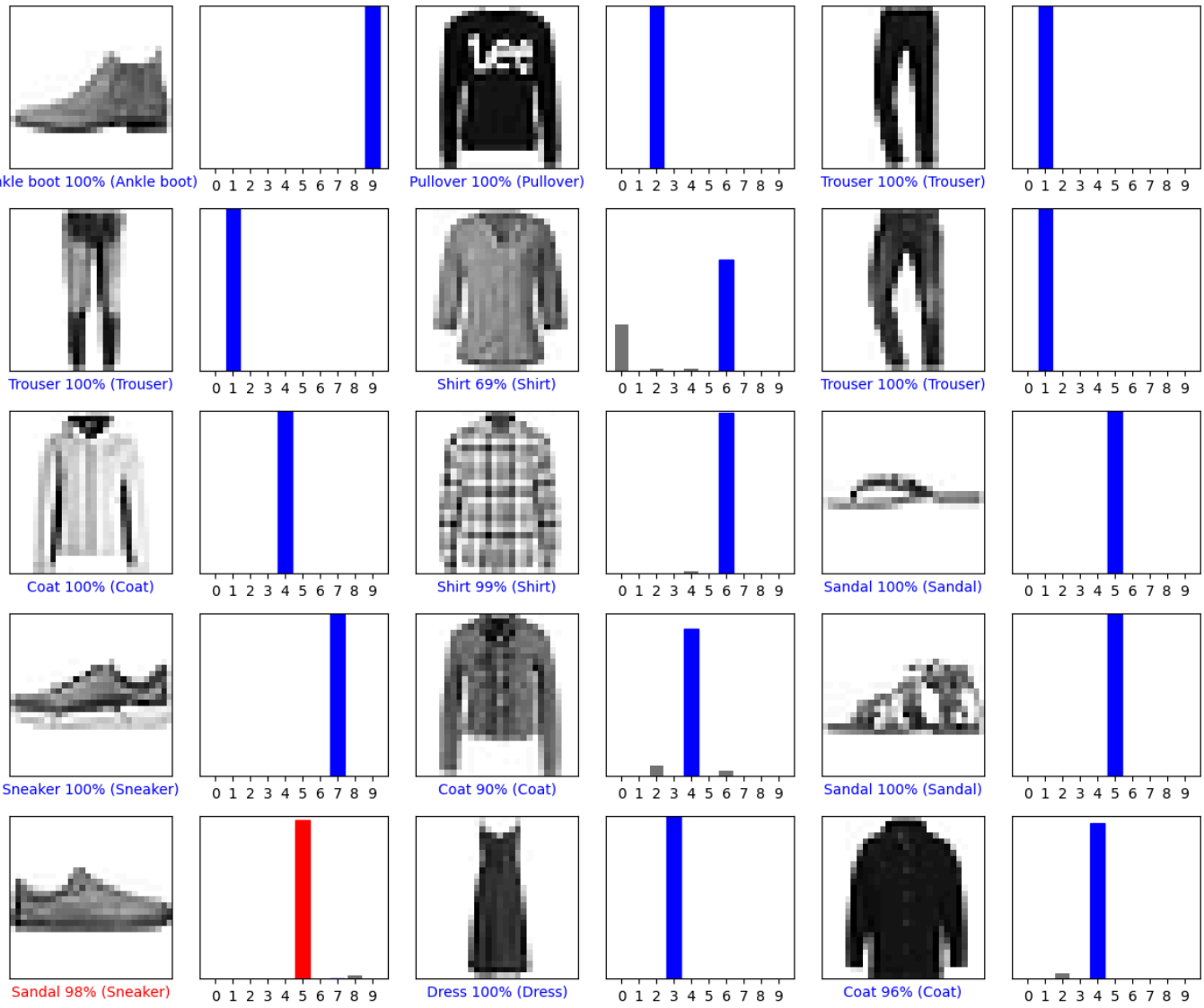
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```




```
i = 12
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()
```



```
# 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()
```




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

print(img.shape)
```

```
(28, 28)
```

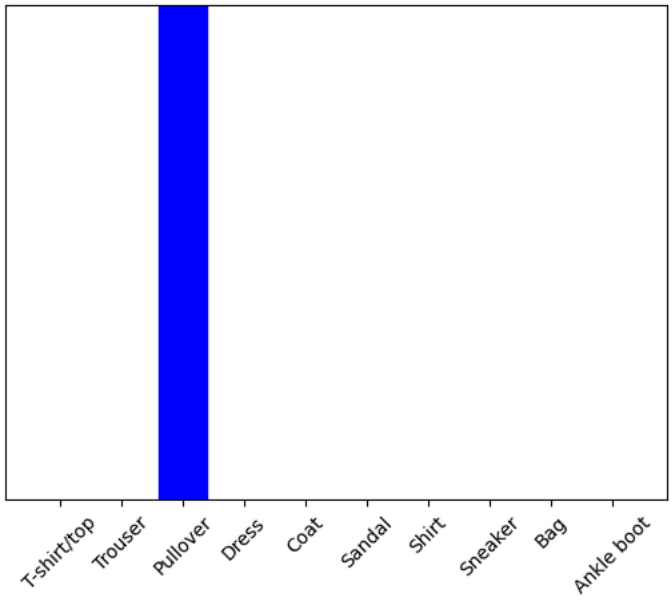
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

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

```
predictions_single = probability_model.predict(img)

print(predictions_single)
```

```
1/1 [==============================] - 0s 30ms/step
[[9.0286740e-06 2.3807123e-13 9.9955529e-01 8.0495425e-14 3.9120100e-04
  1.1454096e-16 4.4491706e-05 5.2756730e-23 4.3317456e-15 2.0521505e-16]]
```

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




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

```
2
```

```
Start coding or generate with AI.
```