# Problem 3.

## a) Install Tensorflow and Keras. Complete this tutorial:

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

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

print(tf.__version__)
```

```
fashion_mnist = 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-datase
32768/29515 [=================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
8192/5148 [==============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
4423680/4422102 [==============================] - 0s 0us/step
```

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

```
train_images.shape
```

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

```
len(train_labels)
```

```
60000
```

```
train_labels
```

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

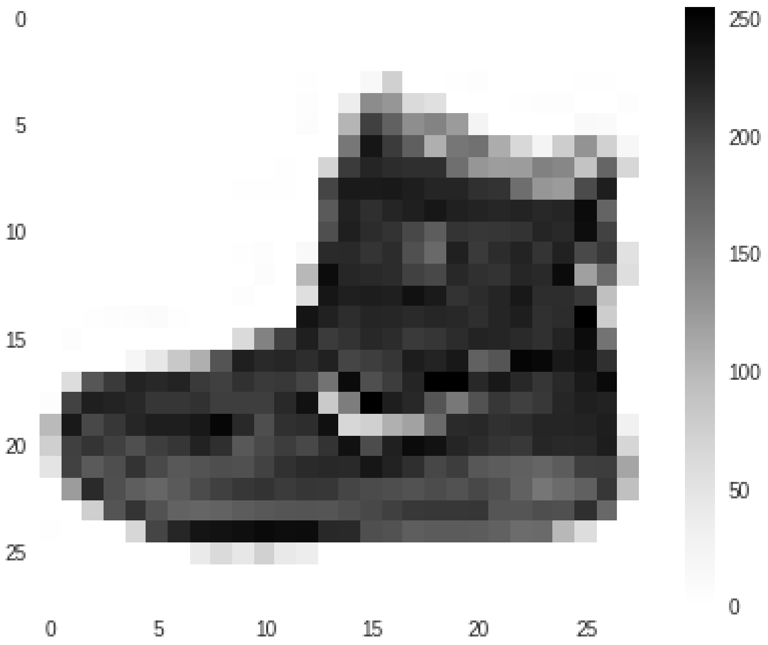
```
test_images.shape
```

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

```
len(test_labels)
```

```
10000
```

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



```
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]])
```

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

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.4936 - ac
Epoch 2/5
60000/60000 [==============================] - 5s 76us/step - loss: 0.3730 - ac
Epoch 3/5
60000/60000 [==============================] - 5s 75us/step - loss: 0.3342 - ac
Epoch 4/5
60000/60000 [==============================] - 4s 75us/step - loss: 0.3101 - ac
Epoch 5/5
60000/60000 [==============================] - 4s 75us/step - loss: 0.2957 - ac
<tensorflow.python.keras.callbacks.History at 0x7f0bc03ff5f8>
```

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

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

```
10000/10000 [==============================] - 0s 38us/step
Test accuracy: 0.875
```

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

```
predictions[0]
```

```
    array([2.7209355e-06, 2.3998481e-08, 2.1429447e-07, 1.2474297e-09,
```

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

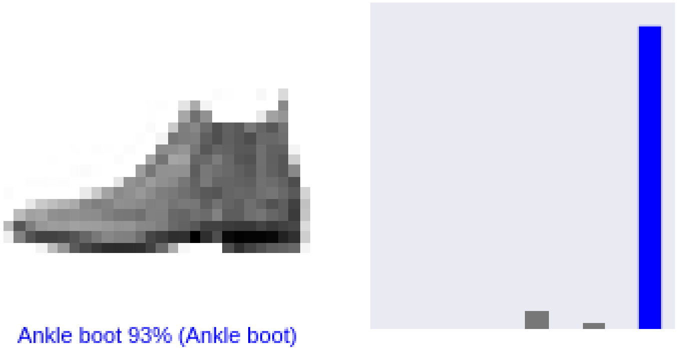
```
9
```

```
test_labels[0]
```

```
9
```

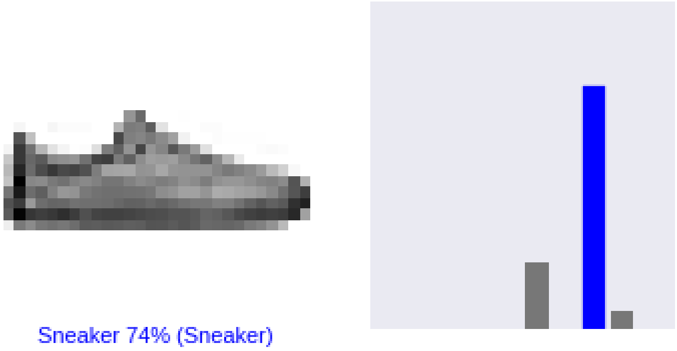
```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  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):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  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, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
```



⊞ CODE ⊞ TEXT

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
```



```
# Plot the first X test images, their predicted label, and the true label
# Color correct predictions in blue, 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, test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions, test_labels)
```

```
# Grab an image from the test dataset
img = test_images[0]

print(img.shape)
```

```
(28, 28)
```

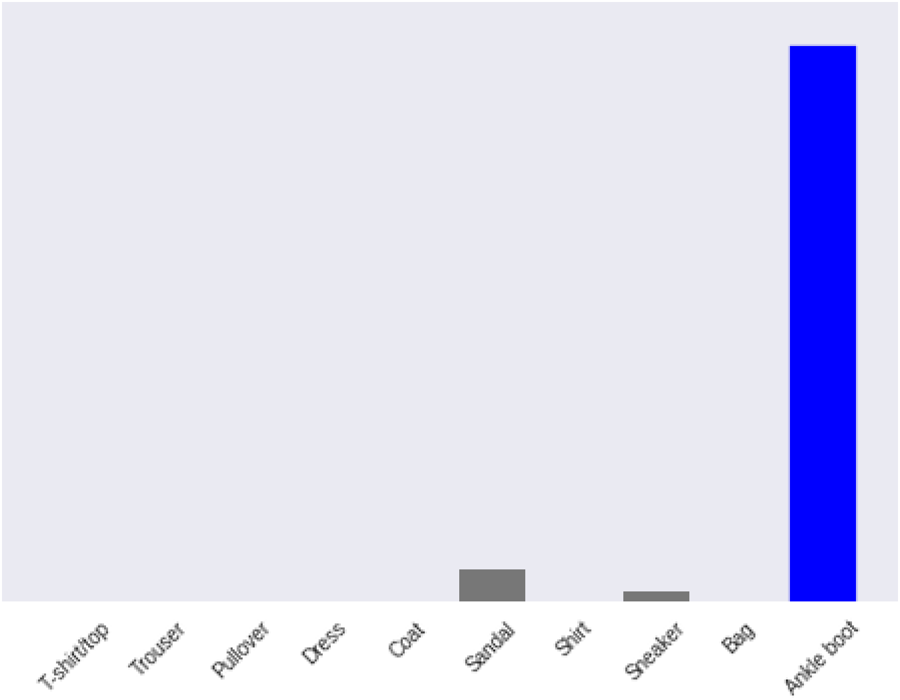
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

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

```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[2.7209351e-06 2.3998478e-08 2.1429403e-07 1.2474296e-09 1.6836999e-07
  5.4479115e-02 1.6102676e-06 1.7795969e-02 4.6110285e-06 9.2771554e-01]]
```

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

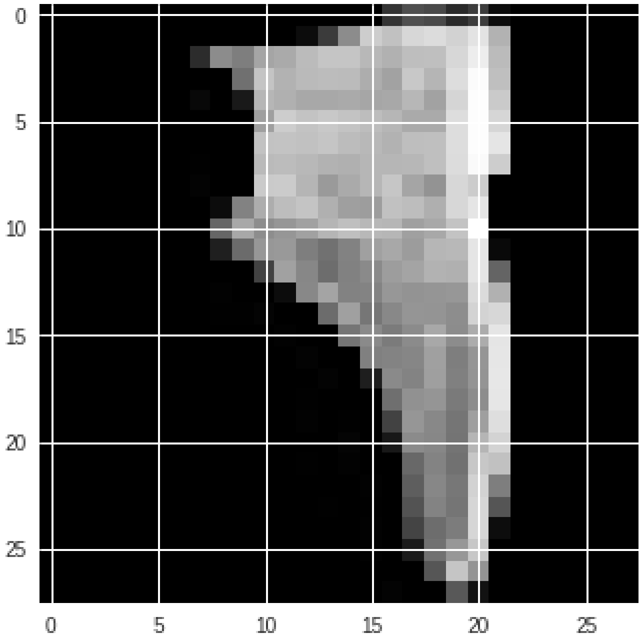


```
np.argmax(predictions_single[0])
```

### b) Create 5 new images by modifying current ones (e.g. by rotation, translation or stretching). Try your best model on them. How accurate is it? Should you be modifying images from the training set or test set?

```
image = test_images[0]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bced4f518>
```

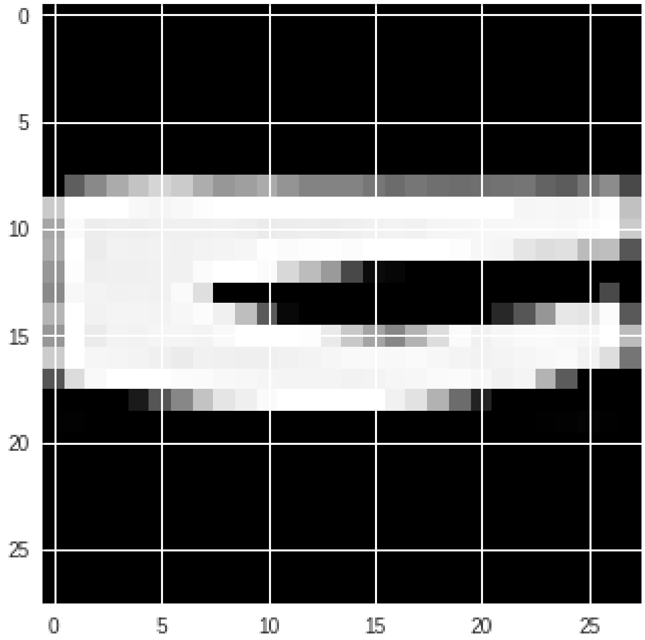


```
image = test_images[1]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bc03c53c8>
```

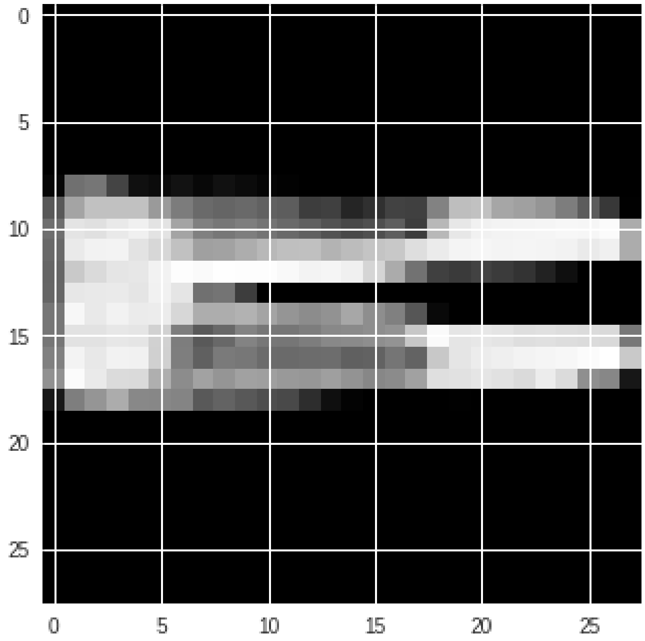
```
image = test_images[2]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bbd52cd68>
```



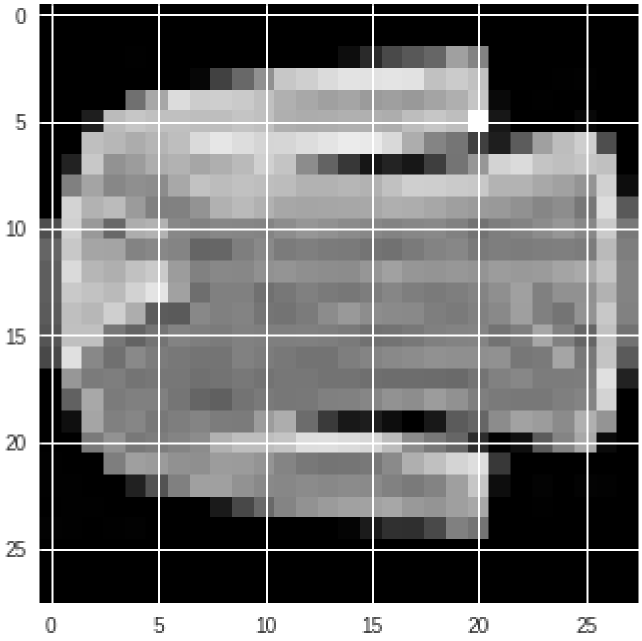
```
image = test_images[3]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bb9506b00>
```



```
image = test_images[4]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bbff165f8>
```



```
# Add the image to a batch where it's the only member.
new_img = (np.expand_dims(image_2,0))

print(new_img.shape)
```

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

```
new_predictions_single = model.predict(new_img)

print(new_predictions_single)
```

```
[[6.4677835e-02 2.1164182e-04 1.0499120e-02 3.7047594e-05 1.5458831e-03
  3.3092888e-06 7.2778083e-02 8.2773018e-05 8.5013485e-01 2.9419767e-05]]
```

```
new_predictions_single[0]
```

```
array([6.4677835e-02, 2.1164182e-04, 1.0499120e-02, 3.7047594e-05,
       1.5458831e-03, 3.3092888e-06, 7.2778083e-02, 8.2773018e-05,
       8.5013485e-01, 2.9419767e-05], dtype=float32)
```

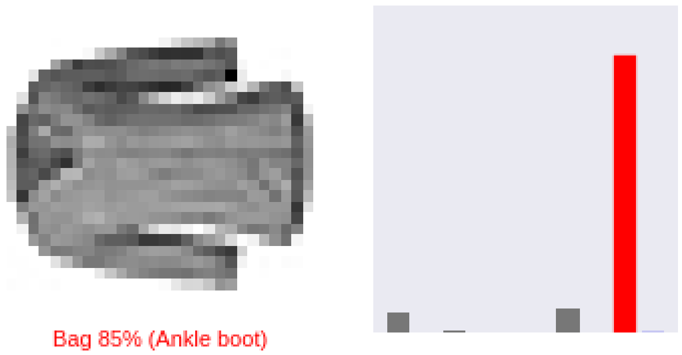
```
np.argmax(new_predictions_single[0])
```

```
8
```

```
test_labels[0]
```

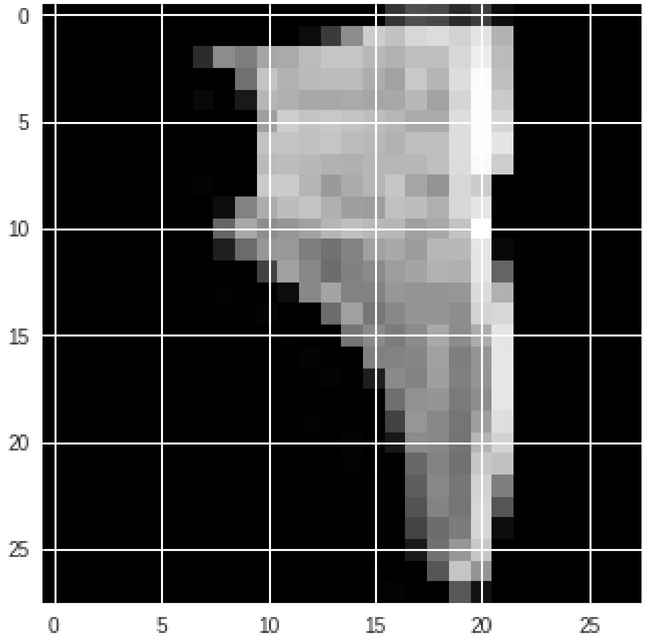
```
9
```

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_predictions_single, test_labels, new_img)
plt.subplot(1,2,2)
plot_value_array(i, new_predictions_single,  test_labels)
```



```
boot_image = test_images[0]
boot_image_2 = np.rot90(boot_image)
plt.imshow(boot_image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bbd515208>
```



```
# Add the image to a batch where it's the only member.
new_boot_img = (np.expand_dims(boot_image_2,0))

print(new_boot_img.shape)
```

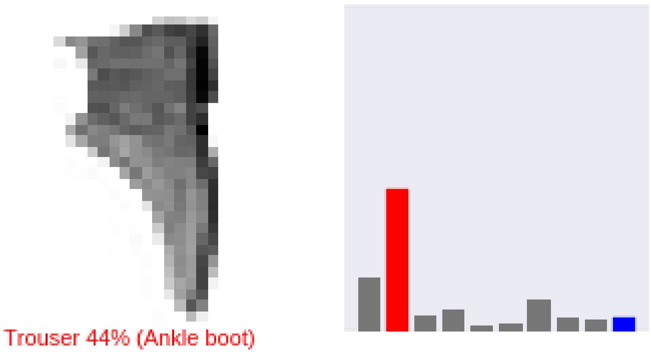
```
(1, 28, 28)
```

```
new_boot_predictions_single = model.predict(new_boot_img)

print(new_boot_predictions_single)
```

```
[[0.1673985  0.4369683  0.05170421 0.0705625  0.01976415 0.0258841
  0.10230368 0.04151233 0.04052156 0.04338068]]
```

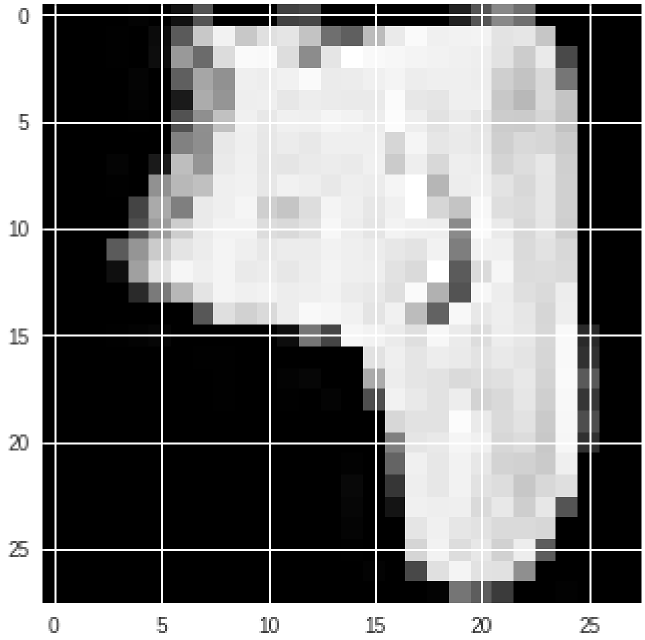
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_boot_predictions_single, test_labels, new_boot_img)
```

```
plt.subplot(1,2,2)
plot_value_array(i, new_boot_predictions_single,  test_labels)
```



```
train_images[0]

boot_train_image = train_images[0]
boot_train_image_2 = np.rot90(boot_image)
plt.imshow(boot_train_image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bb9467588>
```



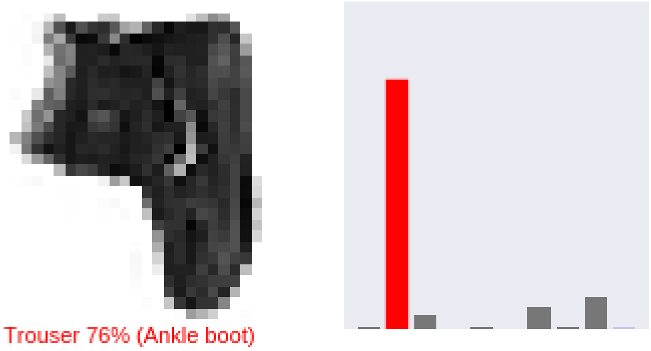
```
# Add the image to a batch where it's the only member.
new_train_boot_img = (np.expand_dims(boot_train_image_2,0))

print(new_train_boot_img.shape)
```

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

```
new_train_boot_predictions_single = model.predict(new_train_boot_img)

print(new_train_boot_predictions_single)
```

```
[[9.6188355e-03 7.5917405e-01 4.5817479e-02 1.3074232e-03 9.3391798e-03
  1.9230654e-05 7.1154073e-02 5.7603340e-03 9.7748585e-02 6.0950922e-05]]
```

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_train_boot_predictions_single, test_labels, new_train_boot_img)
plt.subplot(1,2,2)
plot_value_array(i, new_train_boot_predictions_single,  test_labels)
```



After rotating the images in both test data and train data the images are not predicted in right manner. The prediction is completely wrong.Therefore the accuracy is also zero.