Richard Nadar
RA1911030010109
7-4-2022

ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 14: Applying Deep Learning methods to solve real world problems

AIM: Applying Deep Learning methods to solve real world problem.

LANGUAGE: Python

THEORY:

1. Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge.
2. Deep learning is an important element of data science, which includes statistics and predictive modelling. It is extremely beneficial to data scientists who are tasked with collecting, analysing and interpreting large amounts of data; deep learning makes this process faster and easier.
3. Deep Neural Networks (DNNs) are such types of networks where each layer can perform complex operations such as representation and abstraction that make sense of images, sound, and text.

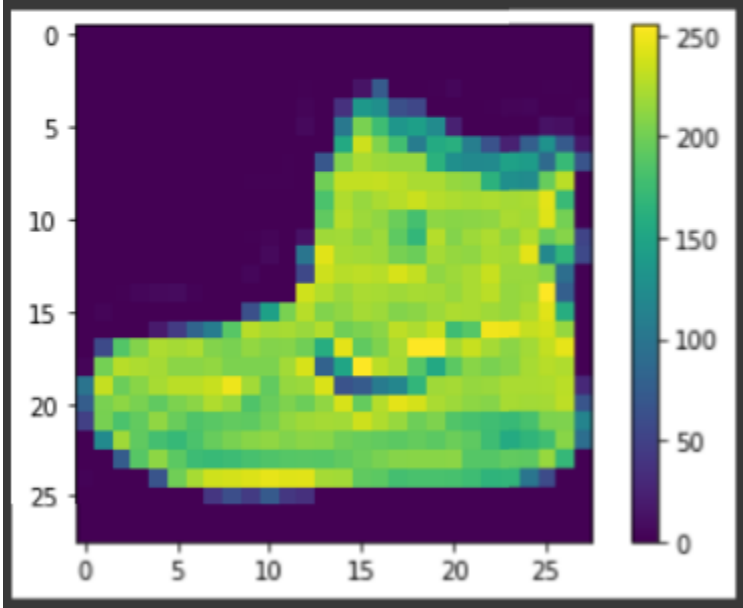
CODE and OUTPUTS:

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

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

```
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-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
40960/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

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




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

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

```
#Fitting the Model
model.fit(train_images, train_labels, epochs=10)
#Evaluating Accuracy
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.5008 - accuracy: 0.8237
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3749 - accuracy: 0.8634
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3353 - accuracy: 0.8767
Epoch 4/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3136 - accuracy: 0.8845
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2940 - accuracy: 0.8922
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2810 - accuracy: 0.8954
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2671 - accuracy: 0.9016
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2569 - accuracy: 0.9036
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2479 - accuracy: 0.9071
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2389 - accuracy: 0.9110
313/313 - 1s - loss: 0.3598 - accuracy: 0.8774 - 639ms/epoch - 2ms/step

Test accuracy: 0.8773999810218811
```

```
#Make Predictions
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
predictions[0]
```

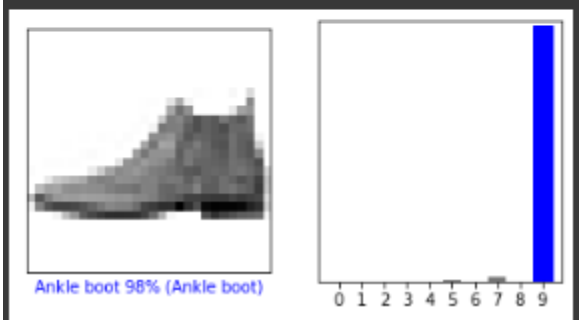
```
array([6.4676947e-06, 5.1236665e-10, 6.5977616e-08, 6.4088843e-12,
       1.8465265e-08, 3.7786926e-03, 2.5812517e-07, 1.6806135e-02,
       1.5682294e-07, 9.7940820e-01], dtype=float32)
```

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

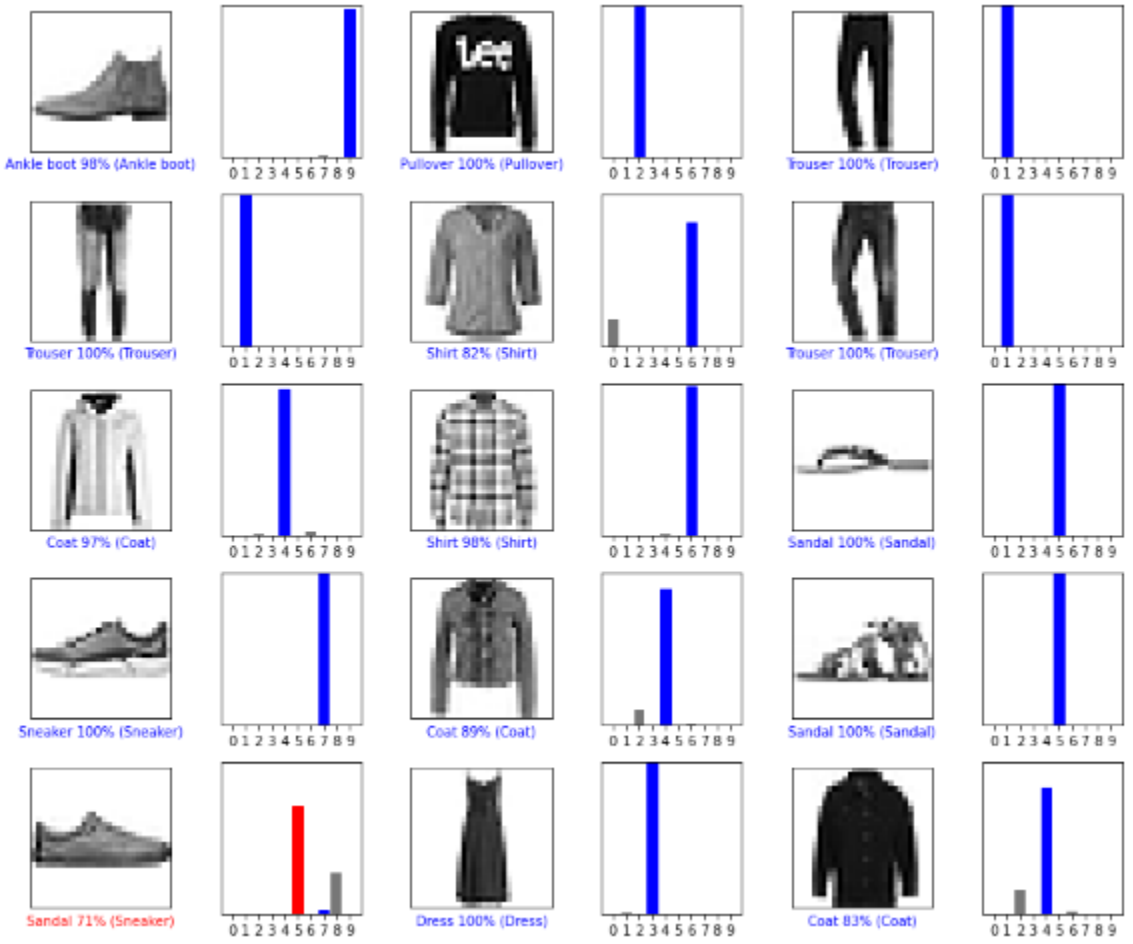
```
9
```

```
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()
```



```
# 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()
```



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

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

RESULT: Thus, we successfully solved one real world life problem using Deep Learning methods.