Practical – 10

Name: Shivam Tawari

Roll no: A-58

Aim: Building a basic deep network using tensorflow

Theory:

Tensorflow:

TensorFlow is an open-sourced library that's available on GitHub. It is one of the more famous libraries when it comes to dealing with Deep Neural Networks. The primary reason behind the popularity of TensorFlow is the sheer ease of building and deploying applications using TensorFlow.

TensorFlow Usage:

TensorFlow excels at numerical computing, which is critical for deep learning. It provides APIs in most major languages and environments needed for deep learning projects: Python, C, C++, Rust, Haskell, Go, Java, Android, IoS, Mac OS, Windows, Linux, and Raspberry Pi.

Moreover, TensorFlow was created keeping the processing power limitations in mind. Implying, we can run this library on all kinds of computers, irrespective of their processing powers. It can even be run on a smartphone.

Fashion Data:

Your Neural Network needs something to learn from. In Machine Learning that something is called datasets. The dataset for today is called Fashion MNIST.

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

In other words, we have 70,000 images of 28 pixels width and 28 pixels height in greyscale. Each image is showing one of 10 possible clothing types.

Code:

Practical 10

```
[2]  import tensorflow as tf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     %matplotlib inline
     from sklearn.datasets import make_classification
     from sklearn.metrics import plot_confusion_matrix
     from sklearn.model_selection import train_test_split
     from tensorflow.keras.datasets import fashion_mnist
```

```
[3]  (train_data, train_labels), (test_data, test_labels)=fashion_mnist.load_data()
```

```
[4]  # Checking  shape,
     print(f"Training sample:\n{train_data[0]}\n")
     print(f"Training label: {train_labels[0]}")
     train_data[0].shape, train_labels[0].shape
```

```
Training sample:
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0  13  73   0
    0   1   4   0   0   0   0   1   1   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3   0  36 136 127  62
   54   0   0   0   1   3   4   0   0   3]
 [  0   0   0   0   0   0   0   0   0   0   0   0   6   0 102 204 176 134
  144 123  23   0   0   0   0  12  10   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 155 236 207 178
  107 156 161 109  64  23  77 130  72  15]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0  69 207 223 218 216
  216 163 127 121 122 146 141  88 172  66]
 [  0   0   0   0   0   0   0   0   0   1   1   1   0 200 232 232 233 229
  223 223 215 213 164 127 123 196 229   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 183 225 216 223 228
  235 227 224 222 224 221 223 245 173   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 193 228 218 213 198
  180 212 210 211 213 223 220 243 202   0]
```

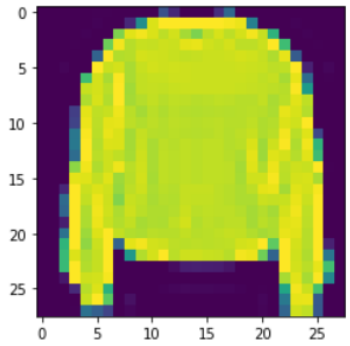
```
 [ 48 203 183 194 213 197 185 190 194 192 202 214 219 221 220 236 225 216
  199 206 186 181 177 172 181 205 206 115]
 [  0 122 219 193 179 171 183 196 204 210 213 207 211 210 200 196 194 191
  195 191 198 192 176 156 167 177 210  92]
 [  0   0  74 189 212 191 175 172 175 181 185 188 189 188 193 198 204 209
  210 210 211 188 188 194 192 216 170   0]
 [  2   0   0   0  66 200 222 237 239 242 246 243 244 221 220 193 191 179
  182 182 181 176 166 168  99  58   0   0]
 [  0   0   0   0   0   0   0  40  61  44  72  41  35   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]

Training label: 9
((28, 28), ())
```

```
[5]  print(train_data.shape)
     print(train_labels.shape)
     print(train_data.dtype)
     print(train_labels.dtype)
```

```
     (60000, 28, 28)
     (60000,)
     uint8
     uint8
```

```
[6]  plt.imshow(train_data[7])
```

```
     <matplotlib.image.AxesImage at 0x7fe3b04b1310>
```



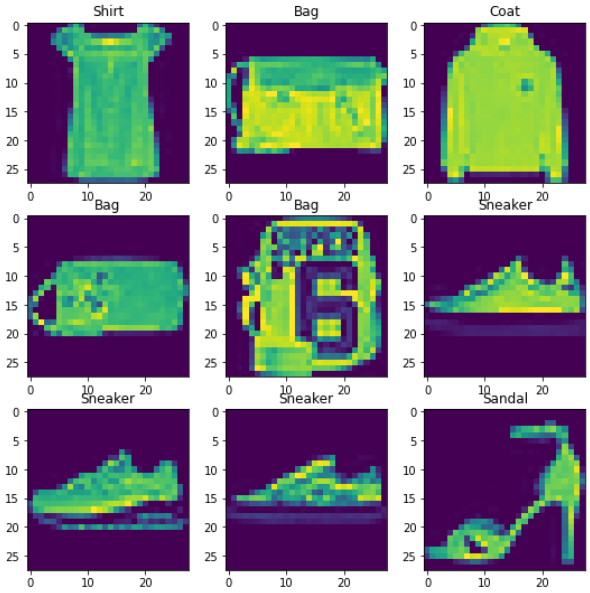

```
[7]  # Check on the lables
     train_labels[7]
```

```
     2
```

```
[8]  # Creating lable list
     class_names=["T-shirt/top", "Trouser","Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker", "Bag"," Ankle boot"]
     len(class_names)
```

```
     10
```

```
[9]  # Check on  sample data and its lable
     import random
     plt.figure(figsize=(9,9))
     for i in range(9):
       ax=plt.subplot(3,3, i+1)
       image_index= random.choice(range(len(train_data)))
       plt.imshow(train_data[image_index])
       plt.title (class_names[train_labels[image_index]])
```

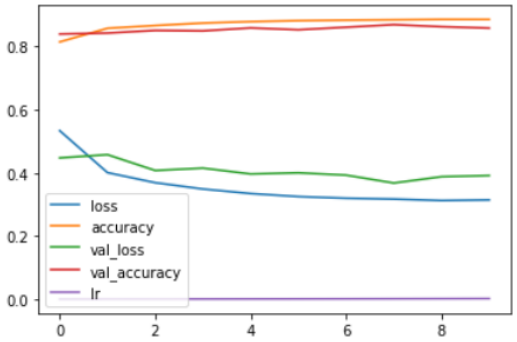


[10]
```
train_data_norm= train_data/225
test_data_norm= test_data/225
```

[11]
```
# Model building.
tf.random.set_seed(42)
#model_01= tf.keras.Sequential( [tf.keras.layers.Flatten(input_shape=(28, 28))])
#model_01.add(tf.keras.layers.Dense(4, activation="relu"))
#model_01.add(tf.keras.layers.Dense(10, activation=tf.keras.activations.softmax))

model_01 = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)), # input layer
  tf.keras.layers.Dense(50, activation="relu"),
  tf.keras.layers.Dense(20, activation="relu"),
  tf.keras.layers.Dense(10, activation="softmax") # output shape is 10, activation is softmax
])


# Model Compile
model_01.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                 optimizer=tf.keras.optimizers.Adam(),
                 metrics=["accuracy"])

# Create Learning rate callback
lr_scheduler= tf.keras.callbacks.LearningRateScheduler(lambda epochs: 1e-3*10**(epochs/20))


# Model fit & Evaluation.
history= model_01.fit(train_data_norm,train_labels,
                      epochs=10, verbose=0,
                      validation_data=(test_data_norm, test_labels), callbacks=[lr_scheduler] )
```

```
[12] pd.DataFrame(history.history).plot()
```

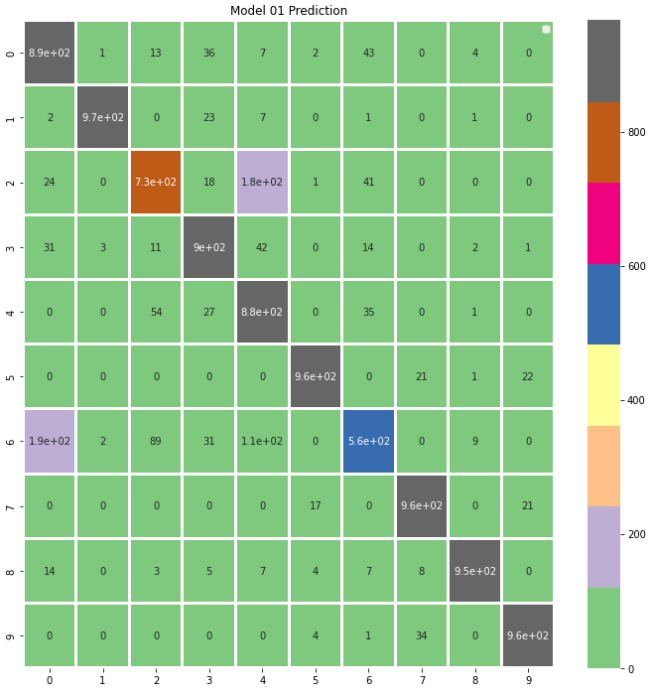
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe3a0608b10>
```



Confusion Metrics

```
[15] from sklearn.metrics import confusion_matrix
     test_pred= model_01.predict(test_data_norm).argmax(axis=1)
     x=confusion_matrix(y_true= test_labels, y_pred=test_pred)
```

```
[16] import seaborn as sn
     plt.figure(figsize=(12,12),facecolor='w')
     plt.title("Model 01 Prediction")
     plt.xticks(ticks= [0,1,2,3,4,5,6,7,8,9], labels=class_names)
     plt.legend( [0,1,2,3,4,5,6,7,8,9], labels=class_names, loc='best')
     plt.ylabel("True Label")
     sn.heatmap(x,annot=True, linecolor='white', linewidth=2, cmap="Accent")
```



Conclusion: Hence, successfully performed Understanding the basics of TensorFlow and Keras.