PROJECT REPORT

Fashion-MNIST classification

In this project we are going to build deep learning neural network model which can classify the different images

Project flow

1.Import Libraries

2.Load data

3.Show image from numbers

4.Feature scaling

5.Build Neural Network model

6. Train model

7. Test model

8. Evaluate the model

9. Confusion matrix

10. Classification report

11. Save model

Data-set link

https://www.tensorflow.org/api_docs/python/tf/keras/datasets/fashion_mnist/load_data

Data

The Fashion-MNIST dataset is proposed as a more challenging replacement dataset for the MNIST dataset.

It is a dataset comprised of 60,000 small square 28×28-pixel grayscale images of items of 10 types of clothing, such as shoes, t-shirts, dresses, and more. The mapping of all 0-9 integers to class labels is listed below.

- 0: T-shirt/top
- 1: Trouser
- 2: Pullover
- 3: Dress
- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

Importing required Libraries

Importing Libaries

```
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
import tensorflow.keras as tk
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Data

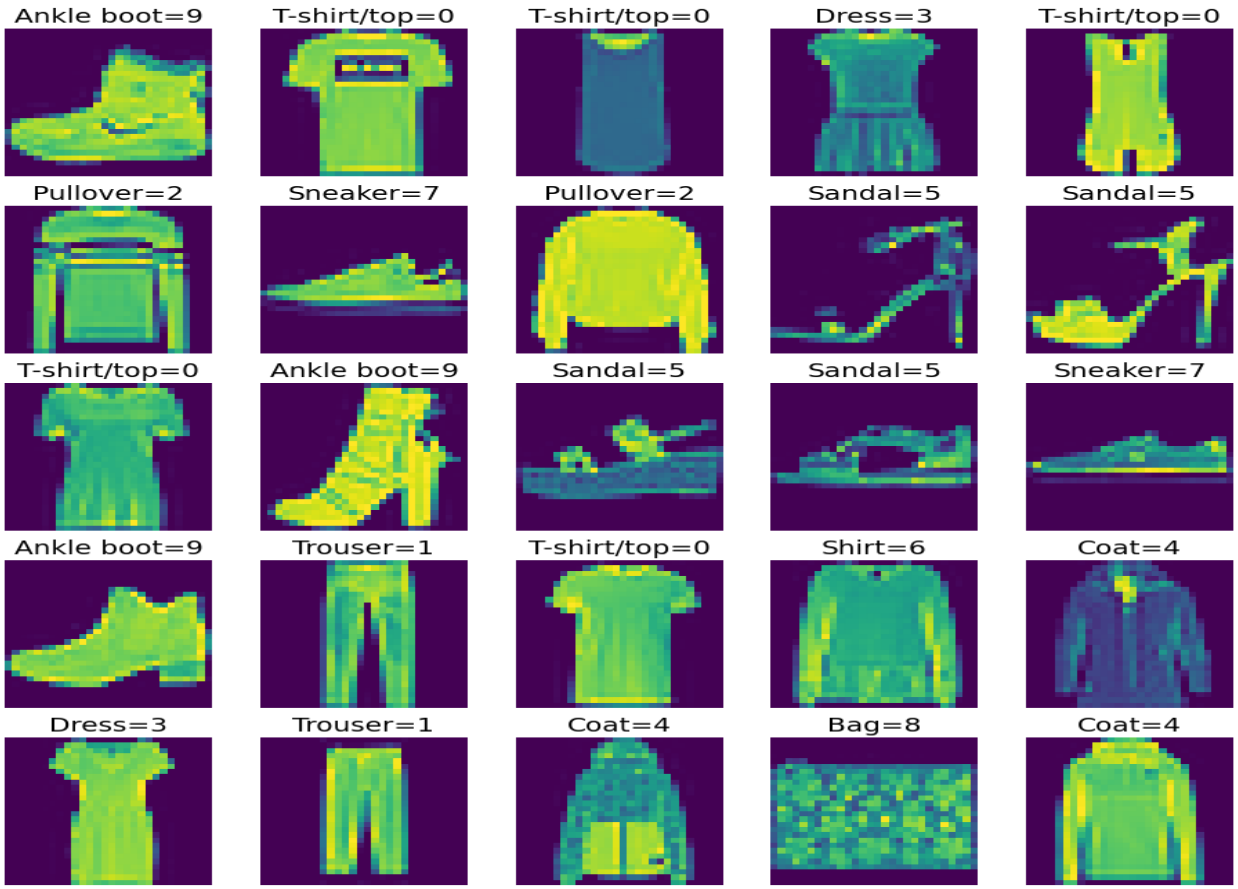
DF=KERAS.DATASETS.FASHION_MNIST.LOAD_DATA()
DF

The data is always in need to be in numeric form in most of the cases data is available in unstructured (images, audio file, etc.) we need to convert it into numeric form but it our case it is already present in structured form numeric form

The shape of the data is

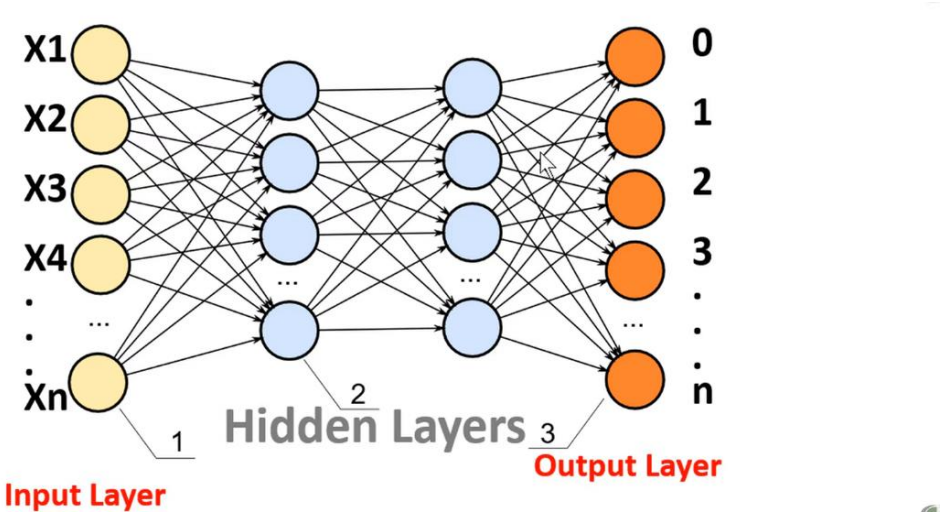
```
[56] (X_train,y_train),(X_test,y_test)=df

X_train.shape , X_test.shape

((60000, 28, 28), (10000, 28, 28))

y_train.shape , y_test.shape

((60000,), (10000,))
```

The actual data look like



Building Neural Network

We will be using convolutional neural network model for the problem. The problem is a multi-class classification, we know that we will require an output layer with 10 nodes in order to predict the probability distribution of an image belonging to each of the 10 classes.



For compile the model

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

- We This will also require the use of a ReLU activation and Softmax function activation function
- We will be using Adam optimizer
  (Adam: A Method for Stochastic Optimization)
- We will be using sparse categorical crossentropy
  (tf.keras.losses.sparse_categorical_crossentropy,)
- We will be using Accuracy matrix

Training the model

We are training the model i.e. X_train and y_train with learning rate epochs (10)

```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3690 - accuracy: 0.8673
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3501 - accuracy: 0.8743
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3373 - accuracy: 0.8776
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3283 - accuracy: 0.8803
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3188 - accuracy: 0.8831
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3097 - accuracy: 0.8881
Epoch 7/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3048 - accuracy: 0.8892
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2990 - accuracy: 0.8896
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2937 - accuracy: 0.8926
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2885 - accuracy: 0.8939
<keras.callbacks.History at 0x7fbf31f97100>
```

Which gives us the accuracy of 0.8939 with loss 0.2885

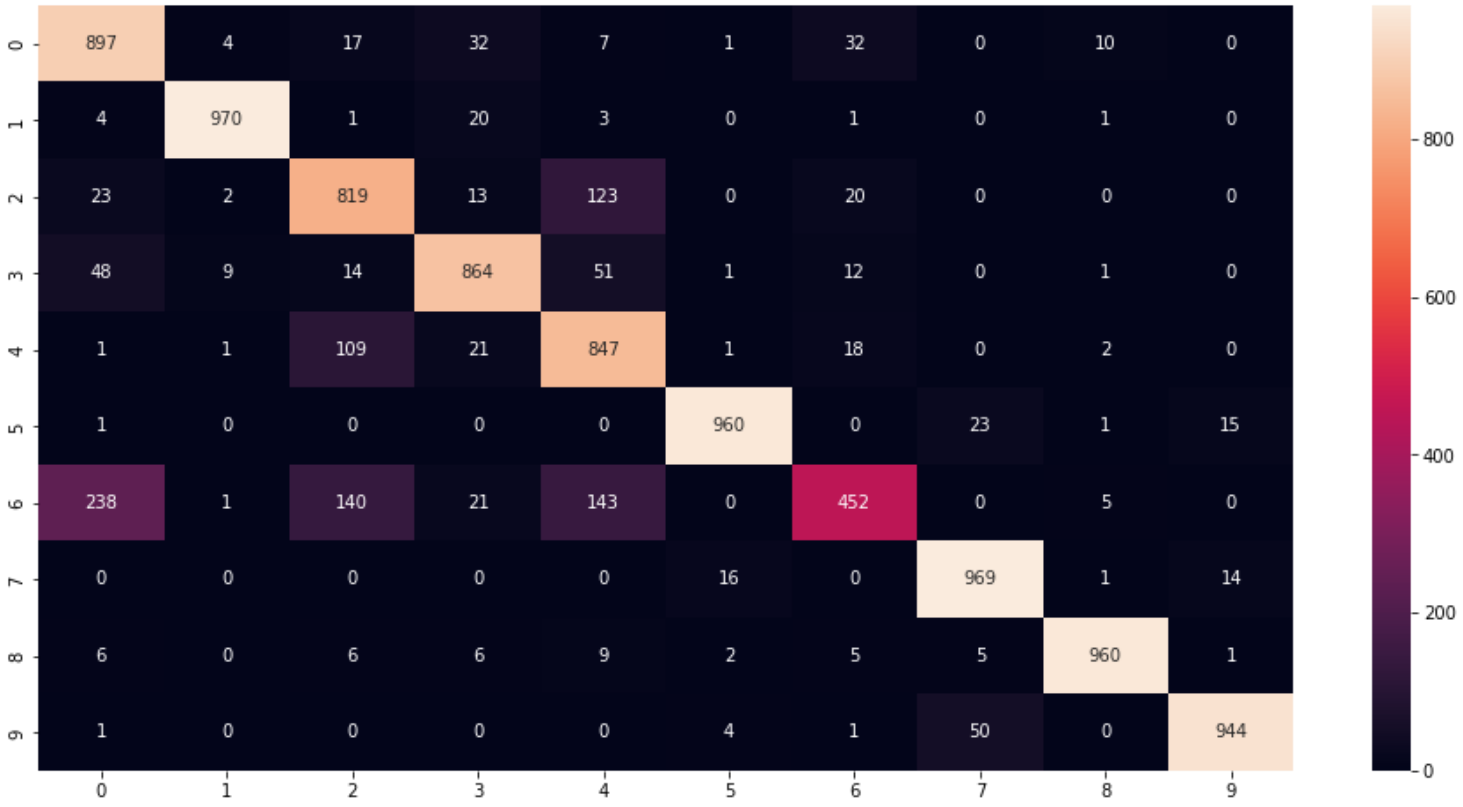
Testing and evaluating the model

1. Testing

```
plt.figure(figsize=(13,13))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.imshow(X_test[i],cmap="Greys")
  plt.axis('off')
  plt.title("Actual= {} \n Predicted = {}".format(class_labels[y_test[i]],
 class_labels[np.argmax(y_pred[i])]))
```

2 Evaluating the model with the help of confusion matrix

```
from sklearn.metrics._plot.confusion_matrix import confusion_matrix
cm=confusion_matrix(y_test,[np.argmax(i)for i in y_pred])

plt.figure(figsize=(16,8))
sns.heatmap(cm,annot=True,fmt='d')
```



Because of similarity between shirts and T-shirts model is not able to per

Classification report

	precision	recall	f1-score	support
T-shirt/top	0.74	0.90	0.81	1000
Trouser	0.98	0.97	0.98	1000
Pullover	0.74	0.82	0.78	1000
Dress	0.88	0.86	0.87	1000
Coat	0.72	0.85	0.78	1000
Sandal	0.97	0.96	0.97	1000
Shirt	0.84	0.45	0.59	1000
Sneaker	0.93	0.97	0.95	1000
Bag	0.98	0.96	0.97	1000
Ankle boot	0.97	0.94	0.96	1000
accuracy			0.87	10000
macro avg	0.87	0.87	0.86	10000
weighted avg	0.87	0.87	0.86	10000

Tp- True positive
Fp – False positive

- Precision score -The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

- Recall score - The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

  The best value is 1 and the worst value is 0.

* F1-Score - represents the model score as a function of precision and recall score

Saving model

A final model is typically fit on all available data, such as the combination of all train and test dataset.

```
model.save('Fashion_mnist.h5')
```

Make Predictions –

We can use our saved model to make a prediction on new images and the size of the image is need to be square with the size 28×28 pixels.

```
model= keras.models.load_model('Fashion_mnist.h5')

model.predict(X_test)
```

Project link - https://github.com/bhushanbkt/Fashion_MNIST.git