Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

T.E.(Extc) Elective: NNFL Course Code: DJ19ECEL5014

EXTC I

NNFL Mini project

Title: Clothing Classifier

- ## Introduction

In the ever-expanding realm of e-commerce and fashion, the ability to automatically classify images of clothing has become paramount. A Clothing Classifier program serves as the technological backbone, enabling businesses to streamline inventory management, enhance user experience, and stay ahead in the competitive landscape.

Objective:

The primary objective of the Clothing Classifier program is to accurately categorize images of various clothing items into predefined classes such as shirts, pants, dresses, shoes, and more. This automated classification system aids in organizing vast datasets, facilitating efficient search functionality, and improving the overall navigation experience for users.

In conclusion, the Clothing Classifier program represents a pivotal tool in the fusion of technology and fashion, empowering businesses to stay agile and responsive in an ever-evolving market.

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

- **System Description**

Import all the necessary libraries to perform the program

```
# import Tensorflow
import tensorflow as tf

# import the other helper libraries required
import numpy as np
import keras
import cv2
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import KFold
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten, BatchNormalization
```

```
[10]
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Dataset to be used using fashion_mnist

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

Loading the dataset returns four NumPy arrays:

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

Labelling the datasets to make it functional. Here we used two columns “class_names” and “label_names” which gives them a name and index respectively.

```
class_names = ['T-shirt/top', 'Trouser','Pullover','Dress','Coat','Sandal','Shirt','Sneaker','Bag','Ankle boot']
label_names = range(10)
class_df = pd.DataFrame(
    {'Label': label_names,
     'Class': class_names,
    })
```

Here, we gave size to the images to be trained. 60000 images were trained with size 28*28

```
train_images.shape
```

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

Labelled all the 60000 images

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

```
train_labels.shape
```

```
(60000,)
```

Here, we gave size to the images to be tested. 10000 images were trained with size 28*28

```
test_images.shape
```

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

Labelled all the 10000 images

```
test_labels.shape
```

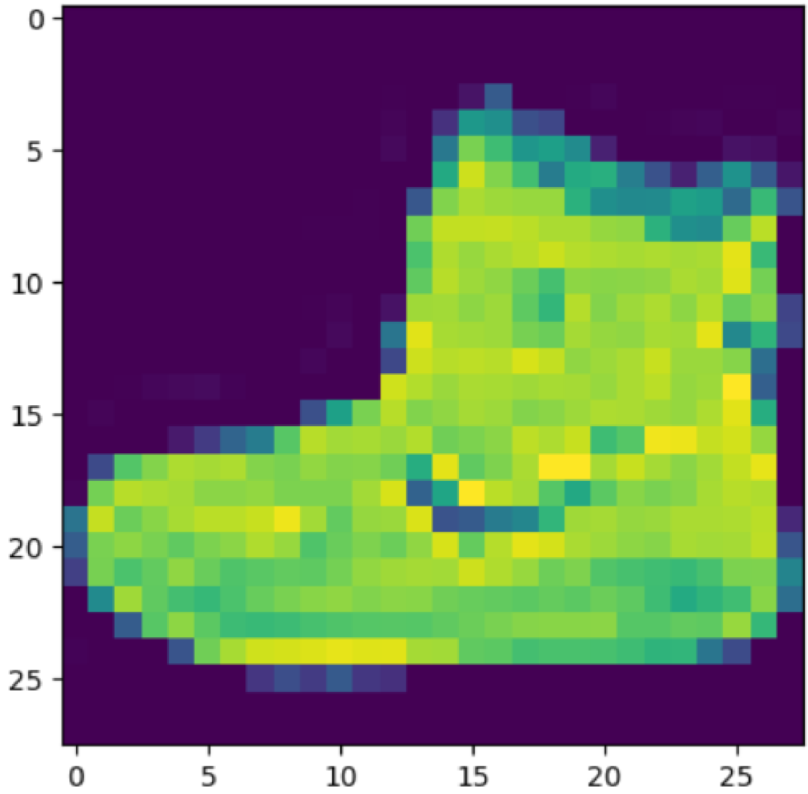
```
(10000,)
```

Here we pre-processed the image that is present at “0”

```
plt.imshow(train_images[0])
plt.show()
```

Output:

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052



Now, we need to grayscale all the images trained

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

To verify that the data is in the correct format and that you are ready to build and train the network, display the first 25 images the training set and display the class name below each image.

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

```
fig = plt.figure(figsize=(15, 15))
columns = 5
rows = 5
a=0
for i in range(1, columns*rows+1):
    img=train_images[a]
    name=class_df.at[train_labels[a], "Class"]
    fig.add_subplot(rows, columns, i)
    plt.xlabel(name)
    plt.imshow(img, cmap=plt.cm.binary)
    a+=1

plt.show()
```

Output:

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

Now we need to create a model having the layers defined

Here Flatten: converts the input to 1- dimensional matrix

Dense: each neuron takes input from previous neurons

Two activations layers

1.) Relu (before normalization)

2.) Softmax (after normalization)

```
model = Sequential()
model.add(Flatten())
model.add(Dense(128, input_dim=784, activation="relu"))
model.add(BatchNormalization())
model.add(Dense(10, activation="softmax"))
```

Now we compile the model using “model. compile” using “adam” optimiser;

“sparsecategorialcrossentropy” loss function and metrics.

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

Now, feeding of model is done. To train the model use “model.fit” with 20 epochs.

```
model.fit(train_images, train_labels, epochs=20)
```

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

```
Epoch 1/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.4879 - accuracy: 0.8286
Epoch 2/20
1875/1875 [==============================] - 5s 2ms/step - loss: 0.3989 - accuracy: 0.8572
Epoch 3/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3718 - accuracy: 0.8660
Epoch 4/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3488 - accuracy: 0.8740
Epoch 5/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3368 - accuracy: 0.8774
Epoch 6/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3283 - accuracy: 0.8828
Epoch 7/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3161 - accuracy: 0.8867
Epoch 8/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3021 - accuracy: 0.8899
Epoch 9/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2941 - accuracy: 0.8927
Epoch 10/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2888 - accuracy: 0.8951
Epoch 11/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2849 - accuracy: 0.8952
Epoch 12/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2770 - accuracy: 0.8989
Epoch 13/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2718 - accuracy: 0.9007
Epoch 14/20
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2666 - accuracy: 0.9010
Epoch 15/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2597 - accuracy: 0.9032
Epoch 16/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2567 - accuracy: 0.9046
Epoch 17/20
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2538 - accuracy: 0.9057
Epoch 18/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2518 - accuracy: 0.9069
Epoch 19/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2489 - accuracy: 0.9079
Epoch 20/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2431 - accuracy: 0.9089
<keras.src.callbacks.History at 0x7e6bc04b5f90>
```

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

Now after 20 epochs, we have to test the model

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

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

Output:

```
313/313 - 1s - loss: 0.3508 - accuracy: 0.8778 - 673ms/epoch - 2ms/step

Test accuracy: 0.8777999877929688
```

Now to add "prediction" feature to the model

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

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

```
313/313 [==============================] - 1s 2ms/step
```

Here, the model has predicted the label for each image in the testing set.

Let's perform 1st prediction:

```
predictions[0]
```

```
array([0.08565874, 0.0856587 , 0.08565874, 0.08565882, 0.0856588 ,
       0.08659141, 0.08565872, 0.08699168, 0.08565888, 0.22680555],
      dtype=float32)
```

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052


```
arg = np.argmax(predictions, axis=1)
np.argmax(predictions[0])
```

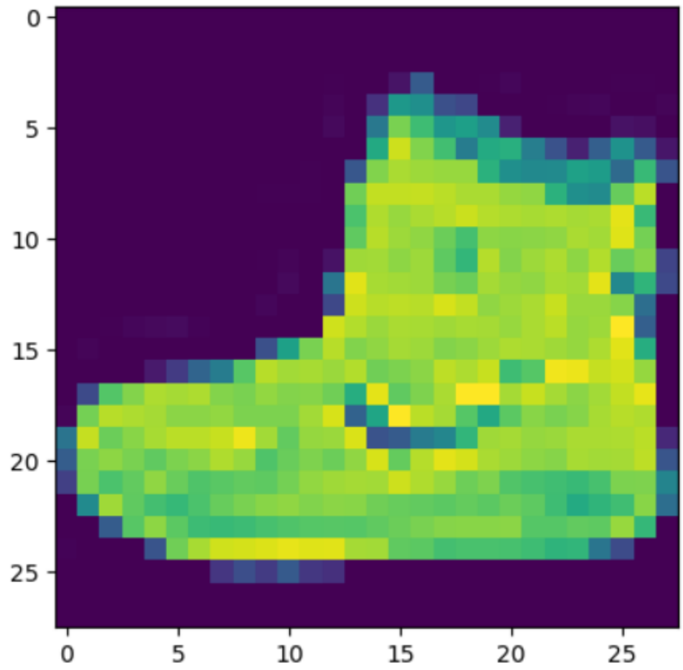
```
9
```

```
test_labels[0]
```

```
9
```

The prediction for the value ‘0’ gave is 9 i.e., according the model the clothing that is present at 0^{th} place in the dataset is present in the 9^{th} place class of clothing we created (ankle boots).

```
plt.imshow(train_images[0])
plt.show()
```



Let’s make another prediction for 1234^{th} position:

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052


```
predictions[1234]
```

```
array([0.08851852, 0.08839637, 0.10867459, 0.08837378, 0.1744287 ,
       0.08831084, 0.09836431, 0.08831106, 0.08831085, 0.0883109 ],
      dtype=float32)
```

```
arg = np.argmax(predictions, axis=1)
np.argmax(predictions[1234])
```

```
4
```

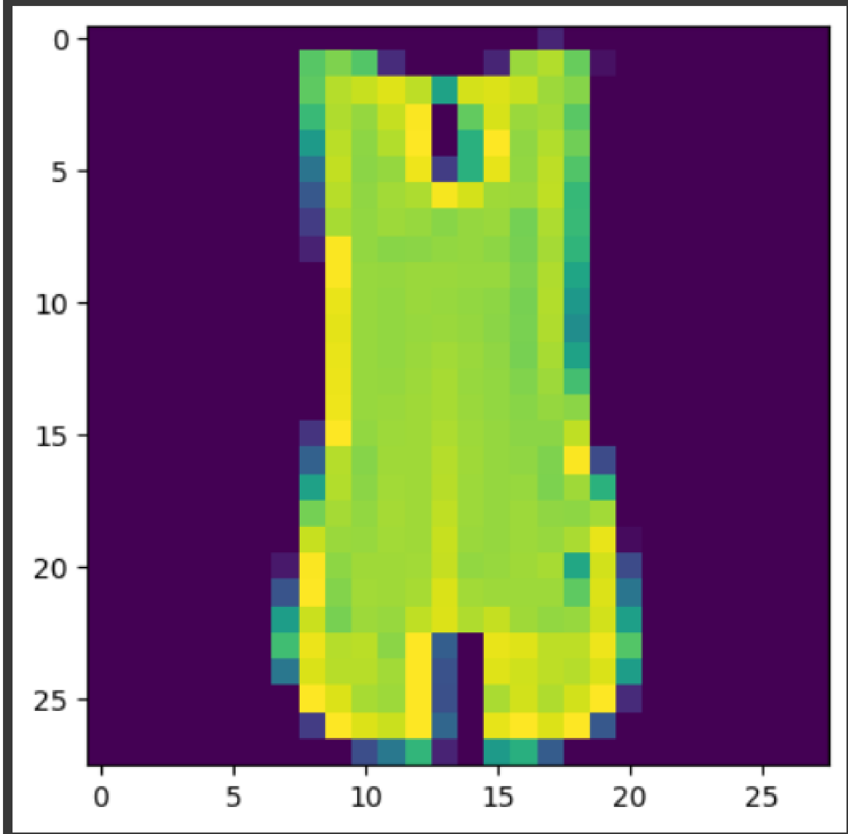
```
test_labels[1234]
```

```
4
```

Model gave prediction: 4 i.e.; Coat

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

```
plt.imshow(train_images[4])
plt.show()
```



Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

- **Results and Discussion**

  The model we created is working well alongside fashion_mnist dataset and OpenCV. The working of the model is moderate and could be better. Results it provided were accurate.

- **Conclusion**

In conclusion, the Clothing Classifier project redefines the fashion landscape by seamlessly integrating advanced image recognition. Its precision in categorization, user-friendly interface, and scalability empower businesses for efficient inventory management and enhanced customer experiences. Beyond mere classification, it serves as a data-rich tool, providing insights critical for strategic decision-making. This project is a pivotal step towards a more connected, efficient, and data-driven future for the fashion industry.

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052

Sahat Bhan 60002210040
Yash Makwana 60002210053
Kris Shah 60002210066
Manish Parmar 60002210052