FLIP ROBO

IMAGE SCRAPING & CLASSIFICATION PROJECT

Submitted by:

Shama Tanweer

Internship-12

ACKNOWLEDGMENT

I would like to express my sincere gratitude to FlipRobo Technologies for supporting me throughout the internship and giving me the opportunity to explore the depth of Data Science by providing multiple projects like this.

Separately, I would like to thank:

- SME khushboo Garg
- Data Trained Team

I have taken help from following sites whenever stuck:
https://stackoverflow.com/questions/35050753/how-big-should-batch-size-and-number-of-epochs-be-when-fitting-a-model-in-keras

INTRODUCTION

➢ Business Problem Framing

Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal. This is done to make the model more and more robust

➢ Conceptual Background of the Domain Problem

Nowadays internet is filled with an abundance of images and videos, which is encouraging the development of search applications and algorithms that can examine the semantic analysis of image for presenting the user with better search content and their summarization.Manually checking and classifying images could be a tedious task especially when they are massive in number . Image classification refers to a process in computer vision that can classify an image according to its visual content. It involves the extraction of information from an image and then associating the extracted information to one or more class labels.

➢ Motivation for the Problem Undertaken

The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary objective. I would like to build an efficient model in order to classify images efficiently using best deep learning algorithms and data augmentation techniques

➢ Review of Literature

Recently, image classification is growing and becoming a trend among technology developers especially with the growth of data in different parts of industry such as e-commerce, automotive, healthcare, and gaming. For example,Whenever users upload a photo, Facebook is able to recognize objects and scenes in it before people enter a description

Photo recognition has also been embraced by other image-centric services online. Google Photos and Apple's Photos app cluster photos on the basis of events and places, plus offer face detection. The application of image recognition significantly enhances users' experience. It helps them organize their photos in meaningful series.

Analytical Problem Framing

Data Collection Phase

For this project we scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping are:

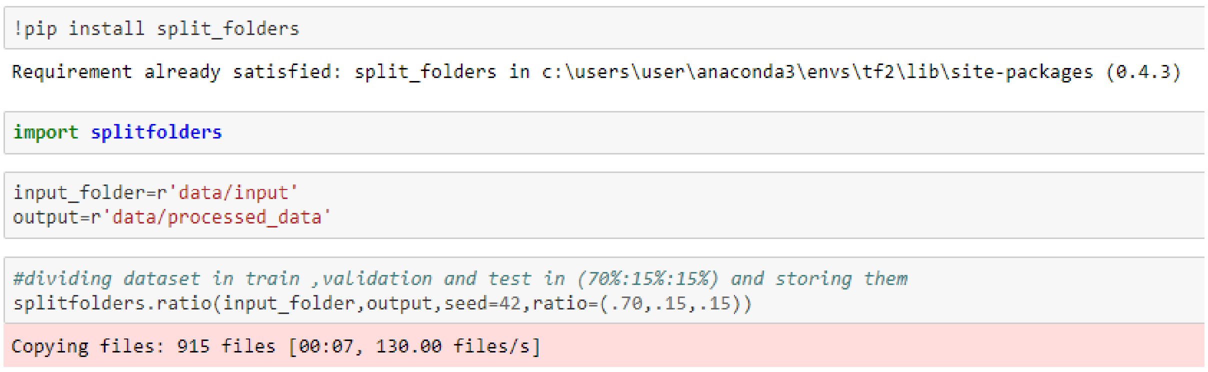
- Sarees (women)
- Trousers (men)
- Jeans (men)

We have scraped 305 images of each category and build data from it.

. That data will be provided as an input to the deep learning problem

• Mathematical/ Analytical Modeling of the Problem

The dataset is then divided into train,validation and test by using splitfolders in ratio (70:15:15)%



```
!pip install split_folders
```

```
Requirement already satisfied: split_folders in c:\users\user\anaconda3\envs\tf2\lib\site-packages (0.4.3)
```

```
import splitfolders
```

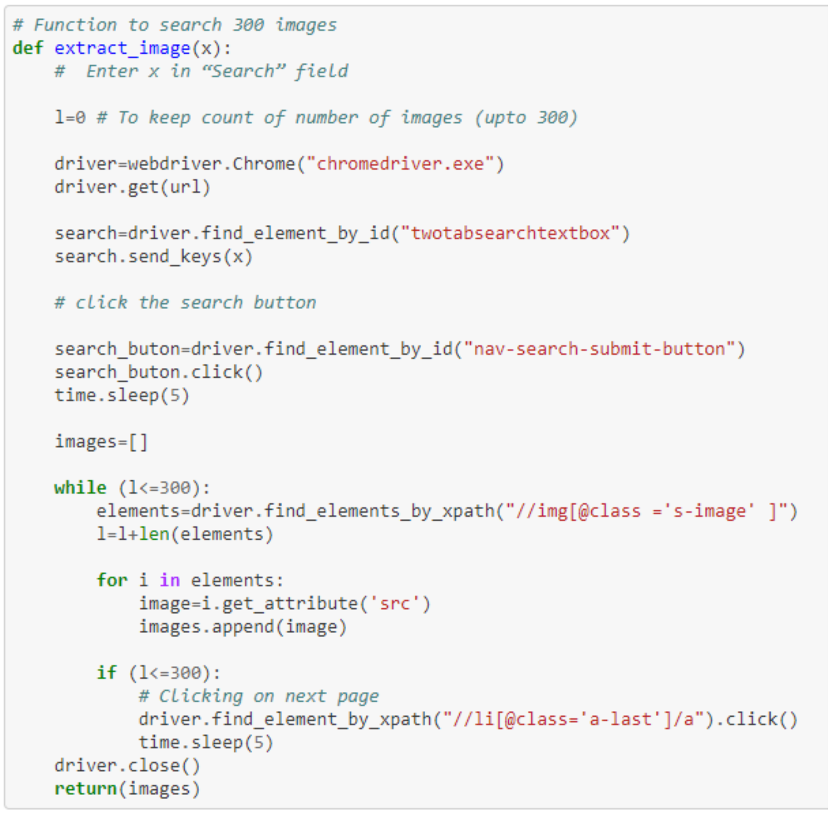
```
input_folder=r'data/input'
output=r'data/processed_data'
```

```
#dividing dataset in train ,validation and test in (70%:15%:15%) and storing them
splitfolders.ratio(input_folder,output,seed=42,ratio=(.70,.15,.15))
```

```
Copying files: 915 files [00:07, 130.00 files/s]
```

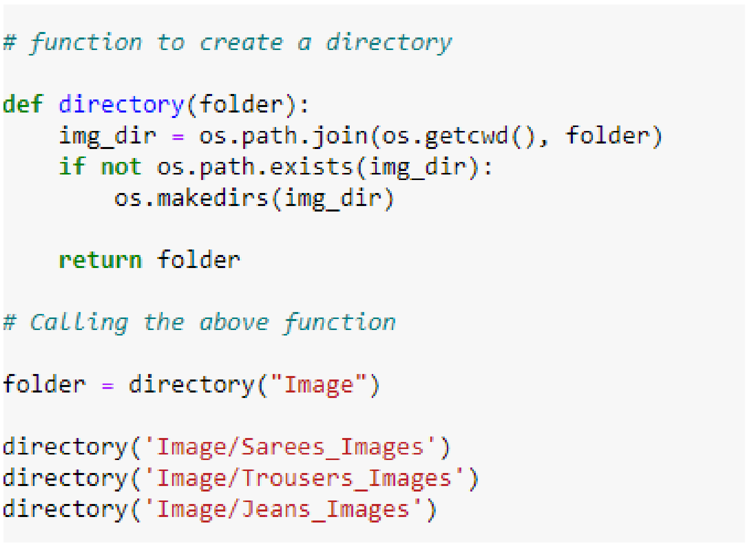
➢ Data Sources and their formats

The data is obtained by scrapping data from e-commerce website like Amazon.com. The scrapper was written in Python programming language using the Selenium package.



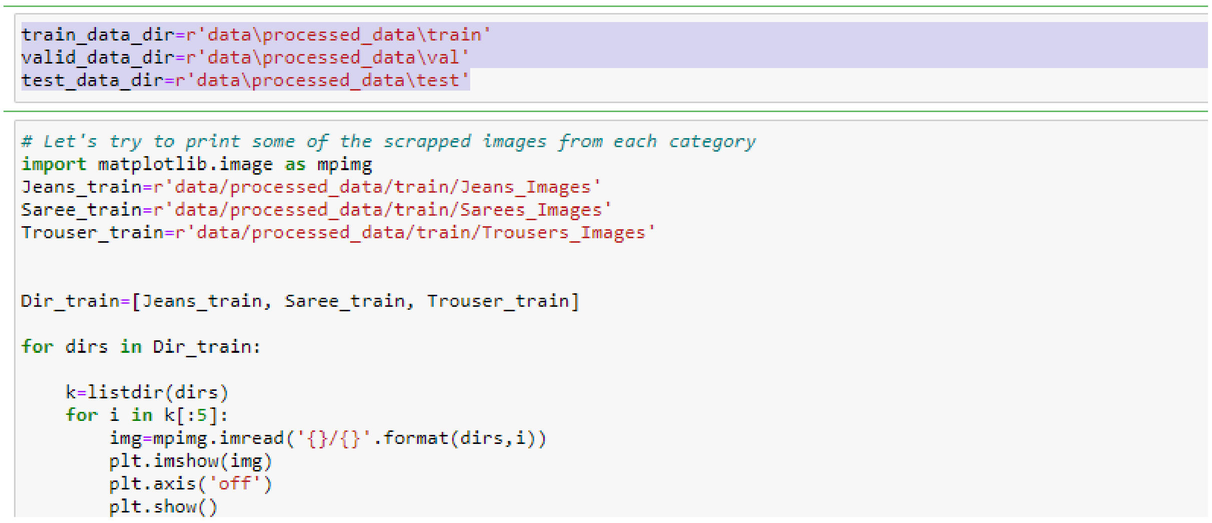
```
# Function to search 300 images
def extract_image(x):
    #  Enter x in “Search” field

    l=0 # To keep count of number of images (upto 300)

    driver=webdriver.Chrome("chromedriver.exe")
    driver.get(url)

    search=driver.find_element_by_id("twotabsearchtextbox")
    search.send_keys(x)

    # click the search button

    search_buton=driver.find_element_by_id("nav-search-submit-button")
    search_buton.click()
    time.sleep(5)

    images=[]

    while (l<=300):
        elements=driver.find_elements_by_xpath("//img[@class ='s-image' ]")
        l=l+len(elements)

        for i in elements:
            image=i.get_attribute('src')
            images.append(image)

        if (l<=300):
            # Clicking on next page
            driver.find_element_by_xpath("//li[@class='a-last']/a").click()
            time.sleep(5)
    driver.close()
    return(images)
```

Next we created directories to store images of each clothing item scraped above. Further we will be downloading images to required folders/directories along with download status message.



```python
# function to create a directory

def directory(folder):
    img_dir = os.path.join(os.getcwd(), folder)
    if not os.path.exists(img_dir):
        os.makedirs(img_dir)

    return folder

# Calling the above function

folder = directory("Image")

directory('Image/Sarees_Images')
directory('Image/Trousers_Images')
directory('Image/Jeans_Images')
```

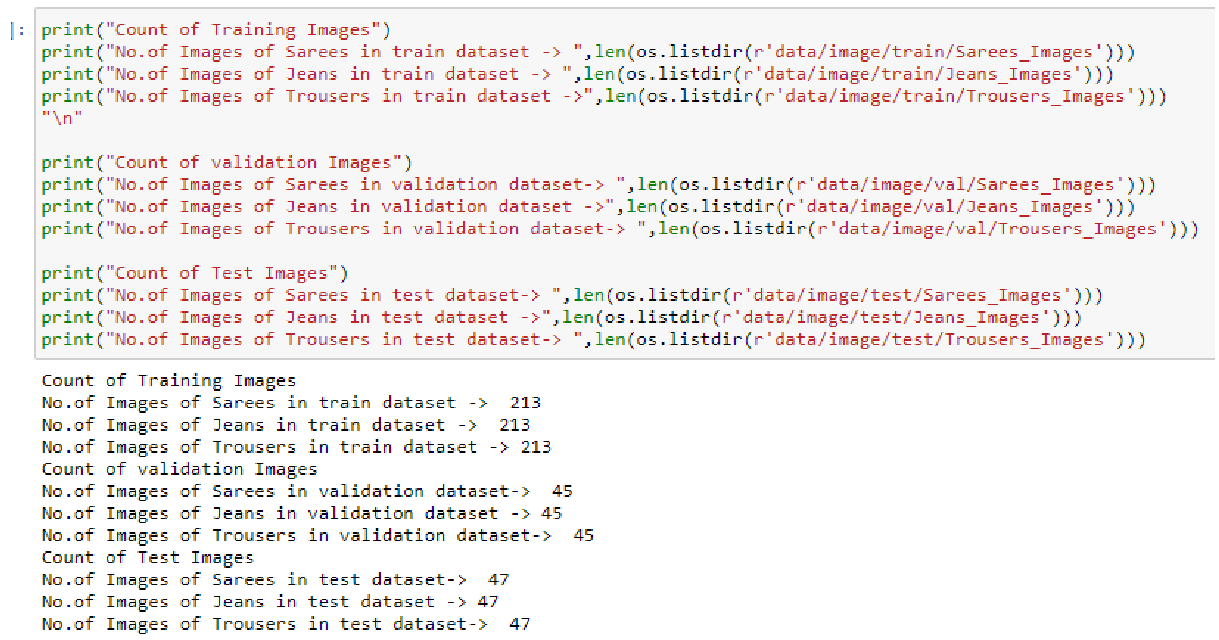
After collecting the data we next do training of the data. For that firstly, we created a main folder called “data” in current working directory inside which I further created 1 folder called input .Input folders contains 3 folders for sarees ,jeans and trousers images each of which contains 305 images.In jupyter notebook using split folders we have dived the data in train ,validation and test for each class and stores them in new folder called processed_data



```python
train_data_dir=r'data\processed_data\train'
valid_data_dir=r'data\processed_data\val'
test_data_dir=r'data\processed_data\test'
```

```python
# Let's try to print some of the scrapped images from each category
import matplotlib.image as mpimg
Jeans_train=r'data/processed_data/train/Jeans_Images'
Saree_train=r'data/processed_data/train/Sarees_Images'
Trouser_train=r'data/processed_data/train/Trousers_Images'


Dir_train=[Jeans_train, Saree_train, Trouser_train]

for dirs in Dir_train:

    k=listdir(dirs)
    for i in k[:5]:
        img=mpimg.imread('{}/{}'.format(dirs,i))
        plt.imshow(img)
        plt.axis('off')
        plt.show()
```

Data Pre-processing



```
]: print("Count of Training Images")
   print("No.of Images of Sarees in train dataset -> ",len(os.listdir(r'data/image/train/Sarees_Images')))
   print("No.of Images of Jeans in train dataset -> ",len(os.listdir(r'data/image/train/Jeans_Images')))
   print("No.of Images of Trousers in train dataset ->",len(os.listdir(r'data/image/train/Trousers_Images')))
   "\n"

   print("Count of validation Images")
   print("No.of Images of Sarees in validation dataset-> ",len(os.listdir(r'data/image/val/Sarees_Images')))
   print("No.of Images of Jeans in validation dataset ->",len(os.listdir(r'data/image/val/Jeans_Images')))
   print("No.of Images of Trousers in validation dataset-> ",len(os.listdir(r'data/image/val/Trousers_Images')))

   print("Count of Test Images")
   print("No.of Images of Sarees in test dataset-> ",len(os.listdir(r'data/image/test/Sarees_Images')))
   print("No.of Images of Jeans in test dataset ->",len(os.listdir(r'data/image/test/Jeans_Images')))
   print("No.of Images of Trousers in test dataset-> ",len(os.listdir(r'data/image/test/Trousers_Images')))
```

```
Count of Training Images
No.of Images of Sarees in train dataset ->  213
No.of Images of Jeans in train dataset ->  213
No.of Images of Trousers in train dataset -> 213
Count of validation Images
No.of Images of Sarees in validation dataset->  45
No.of Images of Jeans in validation dataset -> 45
No.of Images of Trousers in validation dataset->  45
Count of Test Images
No.of Images of Sarees in test dataset->  47
No.of Images of Jeans in test dataset -> 47
No.of Images of Trousers in test dataset->  47
```

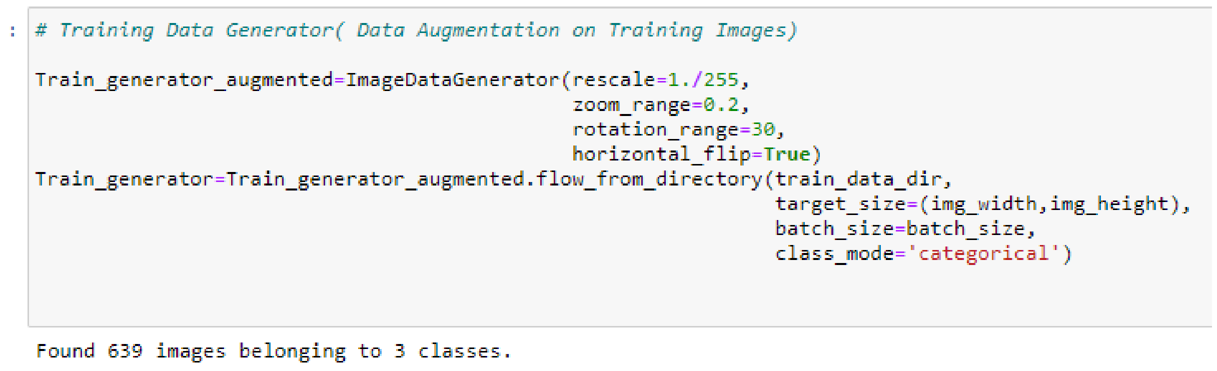
In keras, we get two predefined methods for data loading. One is flow from directory and other is flow from dataframe.

Here we use flow from directory.

- Defining image dimension

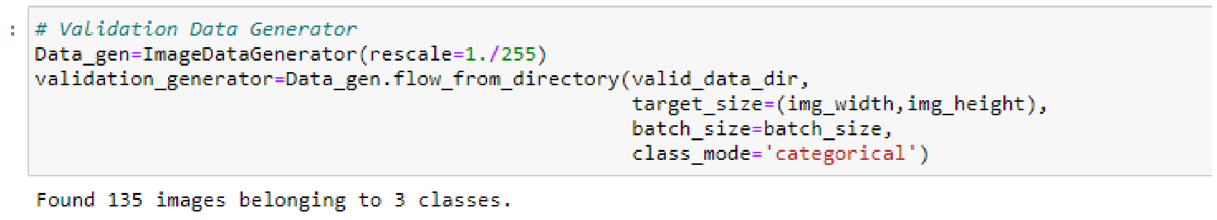
```
input_shape=(224,224,3)
img_width=224
img_height=224
batch_size=32
epoch=100
train_samples=213
val_samples=45
test_samples=47
```

- Rescaling the images - Image data was rescaled by dividing it by 255. This is done using image data generator object



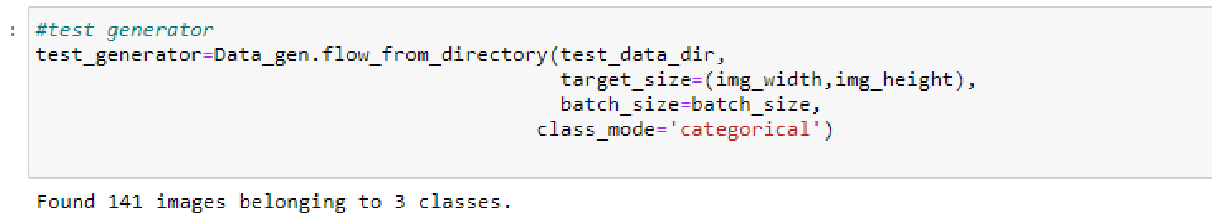
```
: # Training Data Generator( Data Augmentation on Training Images)

  Train_generator_augmented=ImageDataGenerator(rescale=1./255,
                                               zoom_range=0.2,
                                               rotation_range=30,
                                               horizontal_flip=True)
  Train_generator=Train_generator_augmented.flow_from_directory(train_data_dir,
                                                                target_size=(img_width,img_height),
                                                                batch_size=batch_size,
                                                                class_mode='categorical')
```

```
Found 639 images belonging to 3 classes.
```



```
: # Validation Data Generator
  Data_gen=ImageDataGenerator(rescale=1./255)
  validation_generator=Data_gen.flow_from_directory(valid_data_dir,
                                                    target_size=(img_width,img_height),
                                                    batch_size=batch_size,
                                                    class_mode='categorical')
```

```
Found 135 images belonging to 3 classes.
```



```
: #test generator
  test_generator=Data_gen.flow_from_directory(test_data_dir,
                                              target_size=(img_width,img_height),
                                              batch_size=batch_size,
                                            class_mode='categorical')
```

```
Found 141 images belonging to 3 classes.
```

➢ Data Inputs- Logic- Output Relationships

Input data is in the form of images. Images were then transformed to be utilized for our purpose (example preparing a general model)

Hardware and Software Requirements and Tools Used:

- Framework-Annconda
- IDE-Jupyter –Notebook
- Coding Language-Python
- Hardware used: system memory 8GB,
- Processor: core i3

Libraries used: Below are the python library used in this project. •

- ➢ Pandas: To read the Data file in form of data.
- ➢ Matplotlib: This library is typically used to plot the figures for better visualisation of data.
- ➢ Seaborn: A advanced version of Matplotlib
- ➢ Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this it contains very useful joblib library for serialization purpose using which final model have been saved in this project.
- ➢ **Keras: :** A wrapper above tensorflow . This library is used to develop deep learning models using ANN ,CNN and RNN.

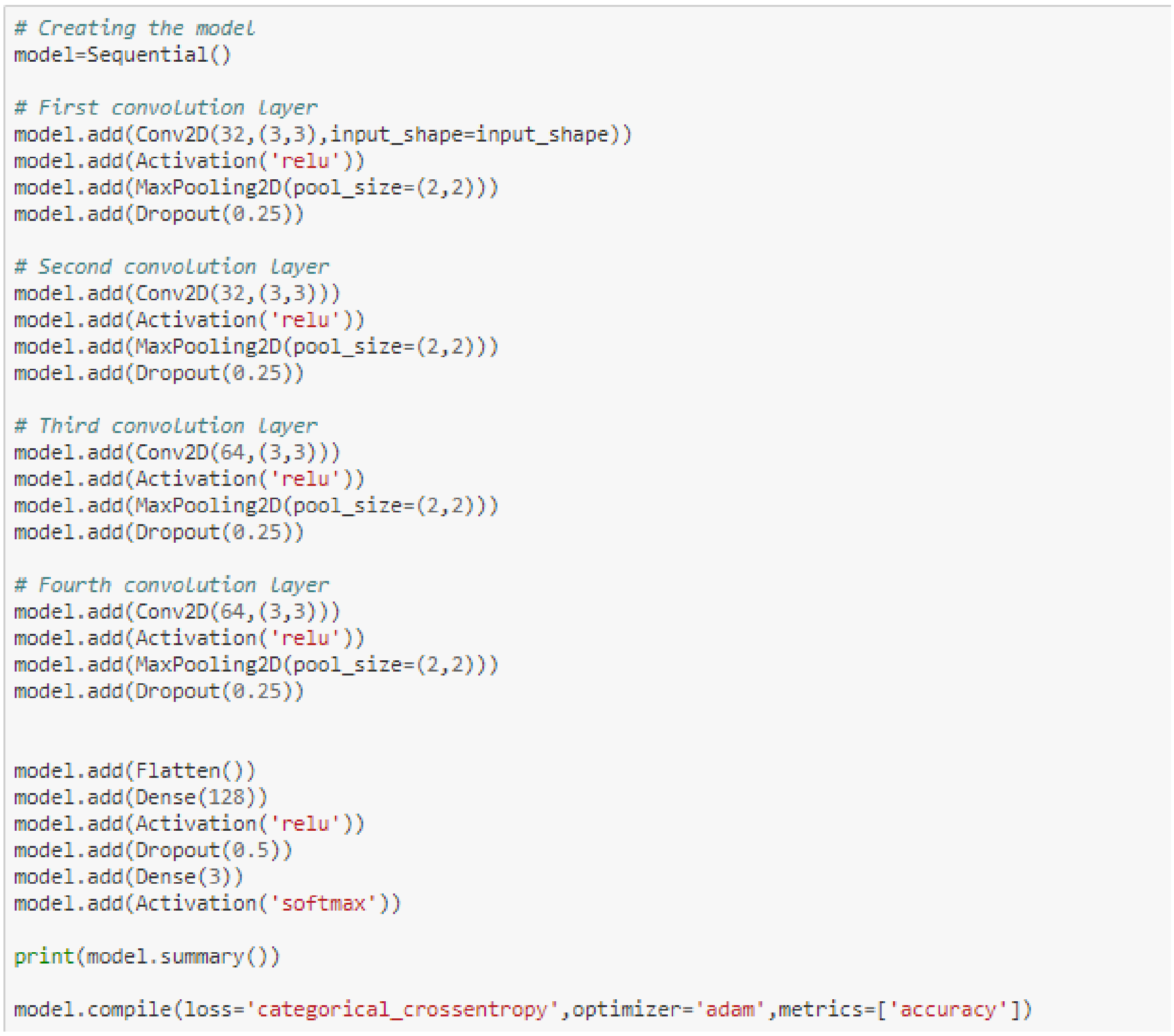
Model/s Development and Evaluation

➢ Testing of Identified Approaches (Algorithms)

- Deep learning network – My randomly prepared deep learning network/model consisted of 4 Conv2D layers and 2 Maxpooling layers in between followed by last Dense layer.
  Since target labels are dependent I have used activation softmax in my last Dense layer

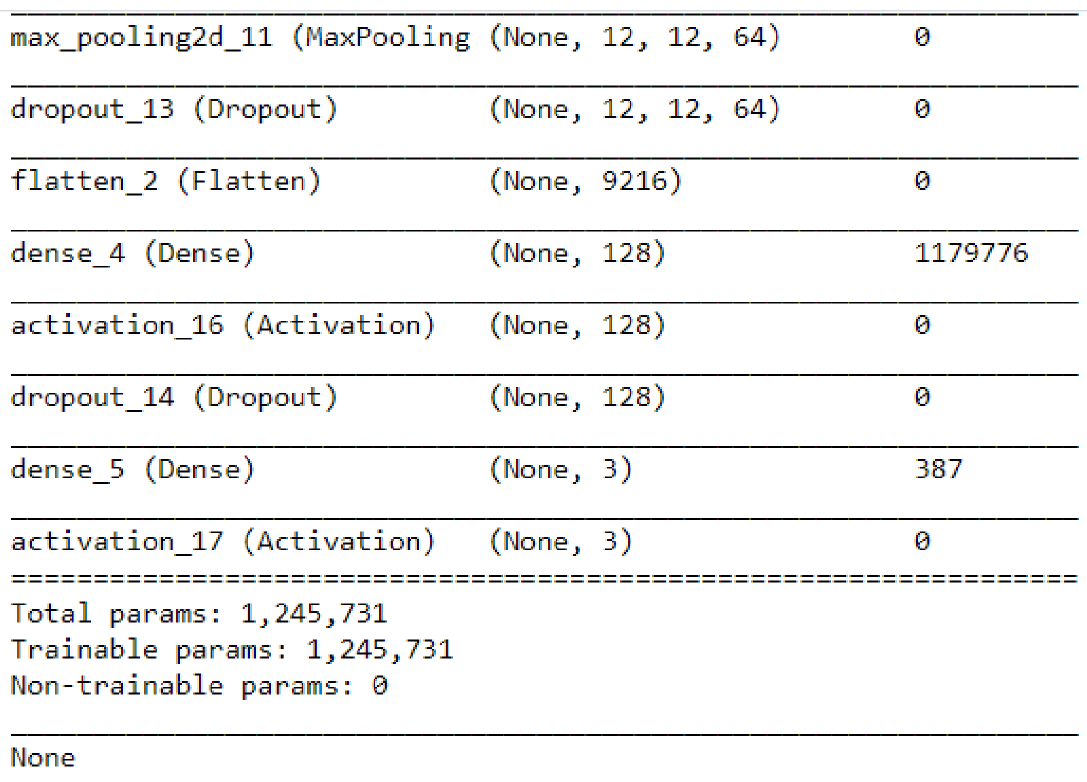
➢ Run and Evaluate selected models

Below is the model prepared



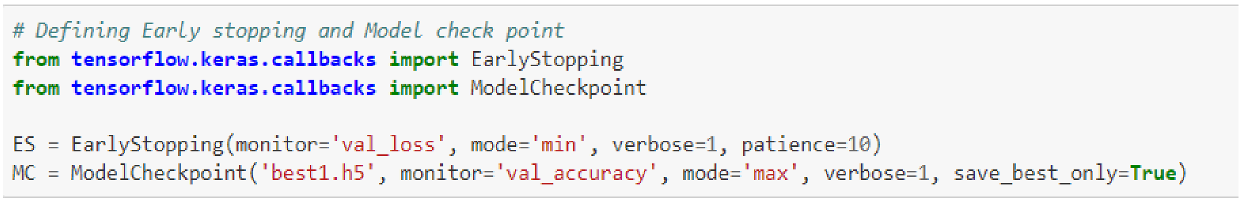
```
# Creating the model
model=Sequential()

# First convolution layer
model.add(Conv2D(32,(3,3),input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Second convolution layer
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Third convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Fourth convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))

print(model.summary())

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

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_8 (Conv2D)            (None, 222, 222, 32)      896       
_________________________________________________________________
activation_12 (Activation)   (None, 222, 222, 32)      0         
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 111, 111, 32)      0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 111, 111, 32)      0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 109, 109, 32)      9248      
_________________________________________________________________
activation_13 (Activation)   (None, 109, 109, 32)      0         
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 54, 54, 32)        0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 54, 54, 32)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 52, 52, 64)        18496     
_________________________________________________________________
activation_14 (Activation)   (None, 52, 52, 64)        0         
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 26, 26, 64)        0         
_________________________________________________________________
dropout_12 (Dropout)         (None, 26, 26, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 64)        36928     
_________________________________________________________________
activation_15 (Activation)   (None, 24, 24, 64)        0         
```

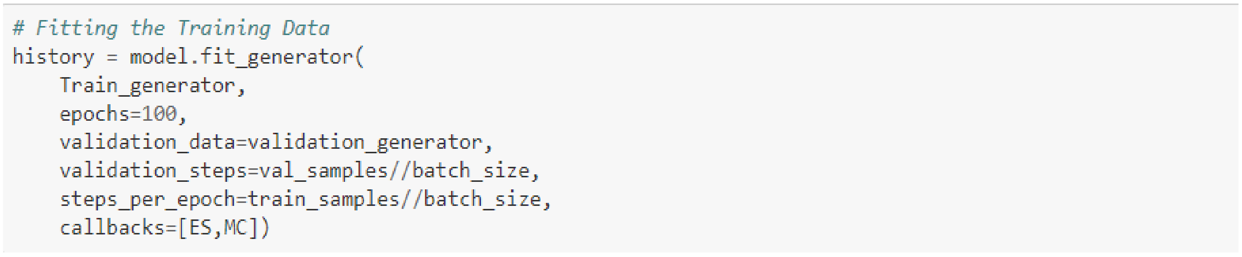


```
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               1179776
_________________________________________________________________
activation_16 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_14 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_17 (Activation)   (None, 3)                 0
=================================================================
Total params: 1,245,731
Trainable params: 1,245,731
Non-trainable params: 0
_________________________________________________________________
None
```

Defined Early Stop criteria and saved the model as 'best.h5' for the best results



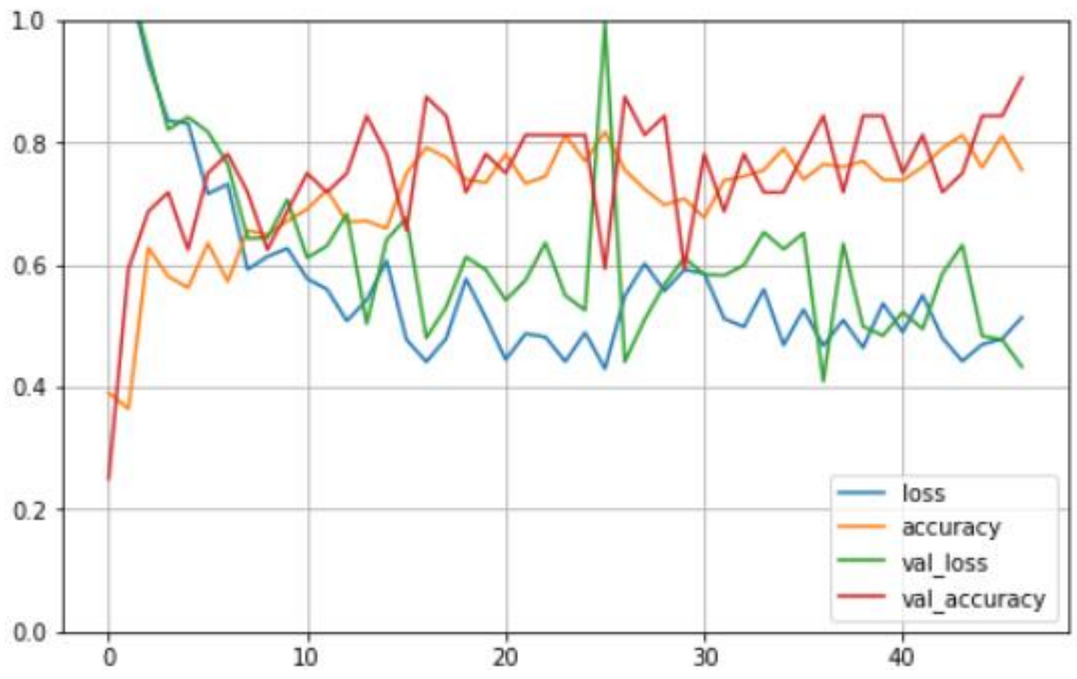
```python
# Defining Early stopping and Model check point
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint

ES = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
MC = ModelCheckpoint('best1.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
```



```python
# Fitting the Training Data
history = model.fit_generator(
    Train_generator,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=val_samples//batch_size,
    steps_per_epoch=train_samples//batch_size,
    callbacks=[ES,MC])
```

Here I have used **early stopping** concept to save the time which proved much helpful .

```
#plotting loss ,accuracy,val_loss,val_accuracy
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```

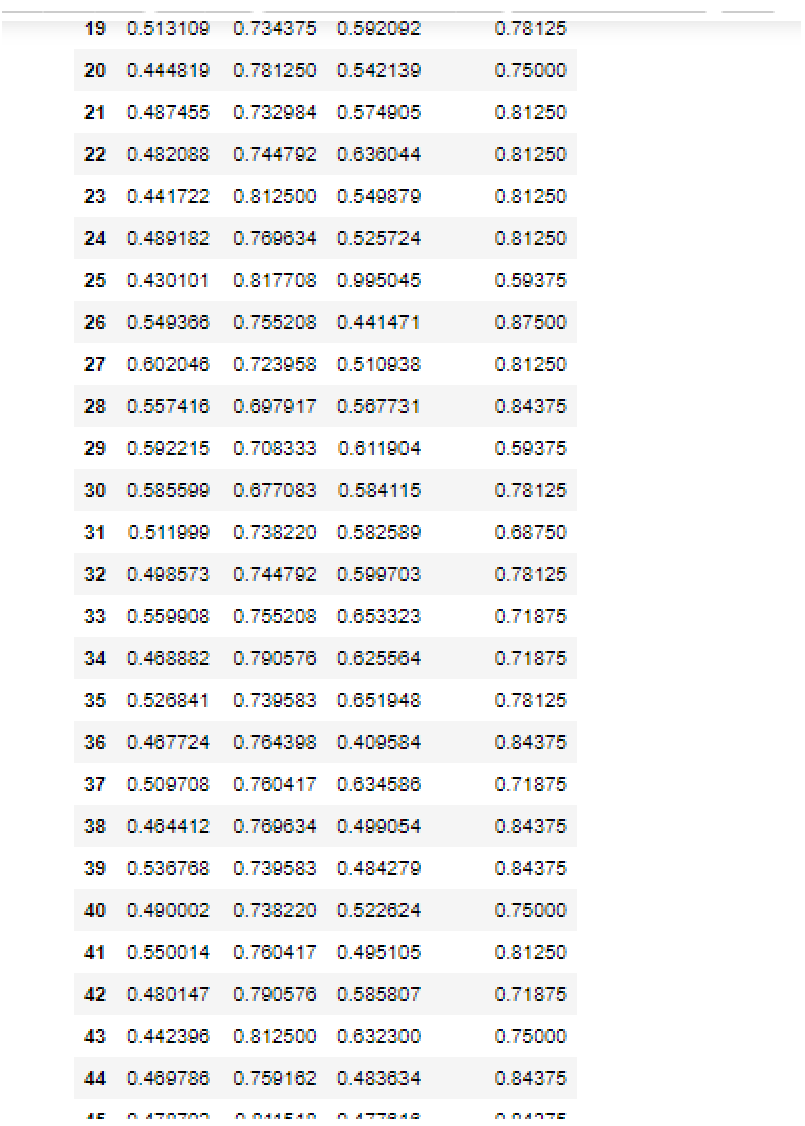


Saving the best model

```
#Saving the best model
model.save('best1.h5')
```

```
losses = pd.DataFrame(model.history.history)
losses
```

	loss	accuracy	val_loss	val_accuracy
0	1.387443	0.390625	1.097744	0.25000
1	1.068188	0.364583	1.069859	0.59375
2	0.929812	0.628272	0.945519	0.68750
3	0.835976	0.581152	0.821086	0.71875
4	0.831470	0.562500	0.841689	0.62500
5	0.716112	0.635417	0.817752	0.75000
6	0.731904	0.572917	0.764775	0.78125
7	0.592611	0.656250	0.643429	0.71875
8	0.613526	0.649215	0.644333	0.62500
9	0.626902	0.671875	0.706565	0.68750
10	0.577405	0.691099	0.611464	0.75000
11	0.560014	0.722513	0.631120	0.71875
12	0.508271	0.670157	0.683231	0.75000
13	0.542536	0.671875	0.503807	0.84375
14	0.606741	0.659686	0.641125	0.78125
15	0.477862	0.750000	0.676564	0.65625
16	0.441411	0.791667	0.480434	0.87500
17	0.479378	0.776042	0.530976	0.84375
18	0.576856	0.739583	0.612837	0.71875



19	0.513109	0.734375	0.592092	0.78125
20	0.444819	0.781250	0.542139	0.75000
21	0.487455	0.732984	0.574905	0.81250
22	0.482088	0.744792	0.636044	0.81250
23	0.441722	0.812500	0.549879	0.81250
24	0.489182	0.769634	0.525724	0.81250
25	0.430101	0.817708	0.995045	0.59375
26	0.549366	0.755208	0.441471	0.87500
27	0.602046	0.723958	0.510938	0.81250
28	0.557416	0.697917	0.567731	0.84375
29	0.592215	0.708333	0.611904	0.59375
30	0.585599	0.677083	0.584115	0.78125
31	0.511999	0.738220	0.582589	0.68750
32	0.498573	0.744792	0.599703	0.78125
33	0.559908	0.755208	0.653323	0.71875
34	0.468882	0.790576	0.625564	0.71875
35	0.526841	0.739583	0.651948	0.78125
36	0.467724	0.764398	0.409584	0.84375
37	0.509708	0.760417	0.634586	0.71875
38	0.464412	0.769634	0.499054	0.84375
39	0.536768	0.739583	0.484279	0.84375
40	0.490002	0.738220	0.522624	0.75000
41	0.550014	0.760417	0.495105	0.81250
42	0.480147	0.790576	0.585807	0.71875
43	0.442396	0.812500	0.632300	0.75000
44	0.469786	0.759162	0.483634	0.84375
45	[illegible]	[illegible]	[illegible]	[illegible]

Prediction

```
# Model Evaluation
evl=model.evaluate(test_generator)
print("Test Loss",evl[0])
print("Test Accuracy",evl[1])
```

```
5/5 [==============================] - 9s 666ms/step - loss: 0.4841 - accuracy: 0.8511
Test Loss 0.48408788442611694
Test Accuracy 0.8510638475418091
```

```
#Loading the saved model
model = load_model('best1.h5')
```

```
#creating instances where elements from test directory will be called
test_jeans=r'data/processed_data/test/Jeans_Images'
test_saree=r'data/processed_data/test/Sarees_Images'
test_trouser=r'data/processed_data/test/Trousers_Images'
```

```
print(test_generator.class_indices)
```

```
{'Jeans_Images': 0, 'Sarees_Images': 1, 'Trousers_Images': 2}
```

Displaying predicted image

```
test_dire=[test_jeans,test_saree,test_trouser]

for test_dir in test_dire:
    for i in listdir(test_dir):
        print("Input Image is:",i)
        img= image.load_img('{}/{}'.format(test_dir,i))
        test_image = image.load_img('{}/{}'.format(test_dir,i),target_size=(224,224))
        test_image = image.img_to_array(test_image)
        plt.imshow(img)
        plt.axis('off')
        plt.show()
        test_image = np.expand_dims(test_image, axis=0)
        result = model.predict(test_image)
        print("Predicted Label is:",np.argmax(result, axis=1),"\n")
```

```
Input Image is: img139.jpeg
```

```
Predicted Label is: [2]
```

```
Input Image is: img141.jpeg
```

```
Predicted Label is: [2]
```

```
Input Image is: img135.jpeg
```

```
Predicted Label is: [1]
```

```
Input Image is: img139.jpeg
```

```
Predicted Label is: [1]
```

```
Input Image is: img143.jpeg
```

```
Predicted Label is: [1]

Input Image is: img146.jpeg
```

```
Predicted Label is: [1]

Input Image is: img150.jpeg
```

```
Input Image is: img206.jpeg
```

```
Predicted Label is: [2]

Input Image is: img211.jpeg
```

```
Predicted Label is: [2]
```

- **Key Metrics for success in solving problem under consideration**

  Since my dataset was balanced evaluation metric was 'accuracy'

CONCLUSION

- **Key Findings and Conclusions of the Study**
- Images pulled from data directory had different shapes which need to be handled to prepare our model.
- Model gives 85% test accuracy.
- The accuracy can be enhanced by using more training images. Here, I have used 305 images per category which can be increased to 1000 per category to give more training data
- **Learning Outcomes of the Study in respect of Data Science**
- Image augmentation was very helpful technique to generate the real augmented data which can be used for various purpose.
- Concept of **early stopping** was excellent since it could save a lot of time in case of huge datasets

Thankyou