Image Classification Project

Submitted by:

Bramee Venkatesan

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Sapna for her constant guidance and support.

Reference sources are: -

- Google
- AnalyticsVidhya.com
- Notes and repository from Data Trained
- Medium.com

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.

  We are trying to give an exposure of how an end-to-end project is developed in this field.

  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.

- Motivation for the Problem Undertaken

  In today world we are dealing with lots of data, which is present in various formats like numbers, categories, images etc.

  The data present in numbers or categorical form is easy to store and understand but image data is somehow different form these.

  In image data we must use some different techniques to deal it and these new techniques, algorithms to save and interpret and the way to deal image data motivates me to undertake this project and build a model for image classification.

Analytical Problem Framing

- Modelling of the problem

  In this project we are dealing with image classification model that will classify between these 3 categories.

  This project is divided into 2 main phases.
  a) Data Collection Phase
  b) Model Building Phase

  In data collection phase we must scrape and collect data and after collection data we must build convolution neural network model.

  Before the model building, we scraped the images of sarees, jeans, trousers from e-commerce website Amazon.

- Data Collection

  We gathered our data from online e-commerce website Amazon. As this is image classification project of 3 categories.

  So, we collected the images of only 3 categories i.e., Sarees, Jeans and Trousers.

  We save these images into our local system and then using Kera's and TensorFlow modules we build an image classification model.

  So, in these steps we collected the URLs links for each image into a separate list and then using these lists we save/downloaded the image into our system.

```
# Importing libraries
import selenium
import time
from bs4 import BeautifulSoup
import os
import requests
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException,StaleElementReferenceException
from selenium.webdriver.common.keys import Keys
```

```
# method to store all item's ursl

links=[]        # empty list to store urls
def urls(item, path):

    driver = webdriver.Chrome('chromedriver.exe')
    driver.get('https://www.amazon.in/')

    driver.find_element_by_id("twotabsearchtextbox").clear()
    time.sleep(2)
    driver.find_element_by_id("twotabsearchtextbox").send_keys(item,Keys.ENTER)
    time.sleep(2)

    start_page = 0
    end_page = 5
    for i in range(start_page,end_page+1):
        sar=driver.find_elements_by_xpath(path)
        for j in sar:
            image=j.get_attribute('src')
            links.append(image)

        nxt_button=driver.find_element_by_xpath("//li[@class='a-last']/a")
        driver.get(nxt_button.get_attribute('href'))
        time.sleep(5)
```

```
# Storing all links of Sarees into a list
urls("Sarees women", "//div[@class='a-section aok-relative s-image-tall-aspect']/img")
```

```
# Storing links into variable
sarees=links
print(len(sarees))
```

```
308
```

```
# using the list of links ,save the image into our local path

def saving_images(folder,list1,item):
    current = os.getcwd()
    new_path = os.path.join(current,folder)
    if not os.path.exists(new_path):
        os.makedirs(new_path)

    for i,link in enumerate(list1):
        try:
            response = requests.get(link)
            print("downloading {} of {}".format(i+1,len(list1)))
            with open(folder+"\\"+item+"_{}.jpg".format(i+1),"wb") as file:
                file.write(response.content)
        except:
            pass
            print("Error occured. Continuing...")
```

```
saving_images(r'C:\img\Saree women',sarees[:308],"saree")
```

```
downloading 7 of 308
downloading 8 of 308
downloading 9 of 308
downloading 10 of 308
downloading 11 of 308
downloading 12 of 308
downloading 13 of 308
downloading 14 of 308
downloading 15 of 308
downloading 16 of 308
downloading 17 of 308
downloading 18 of 308
downloading 19 of 308
downloading 20 of 308
downloading 21 of 308
downloading 22 of 308
downloading 23 of 308
downloading 24 of 308
downloading 25 of 308
```

```
saving_images(r'C:\img\Jeans men',Jeans[:308],"jean")
```

```
downloading 1 of 308
downloading 2 of 308
downloading 3 of 308
downloading 4 of 308
downloading 5 of 308
downloading 6 of 308
```

- Data Pre -Processing

  I have used Jupyter Notebook for data pre-processing and model building. Libraries used in this project are,

  1. Os
  2. Numpy
  3. Matplotlib
  4. Keras
  5. Tensorflow

```
#importing required libraries

import os
from os import listdir
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import numpy as np
from numpy import asarray
from numpy import save

import tensorflow as tf

from keras.preprocessing.image import  load_img
from keras.preprocessing.image import img_to_array

from keras.utils import np_utils

from tensorflow.random import set_seed
from tensorflow.keras import regularizers
import tensorflow as tf
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.random import set_seed
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
```

```
# Loading images and checking sample images
images = os.listdir(r'C:\Users\bramee\OneDrive\Desktop\trainn')
print("Sample images",images[::120])
```

```
Sample images ['jean_1.jpg', 'jean_207.jpg', 'jean_38.jpg', 'saree_146.jpg', 'saree_254.jpg', 'saree_85.jpg', 'trouser_193.jp
g', 'trouser_300.jpg']
```

```
print("Total number of images =",len(images))
```

```
Total number of images = 924
```

After imported the necessary libraries and images, we can see that there are 924 images as a total.

Let's see the sample images and Re-sizing the image to the size of 200*200 and saving the re-sized image to the desired folder.

```
# Let's see the samples images

def sample_images(path,item):
  nrows = 4
  ncols = 4

  fig = plt.gcf()
  fig.set_size_inches(nrows*4, ncols*4)

  nxt_image = [os.path.join(path,figr)
  for figr in item]

  # Generating plot
  for i ,img_path in enumerate(nxt_image):
    plt.subplot(nrows, ncols, i+1)

    img = mpimg.imread(img_path)
    plt.axis('off')
    plt.imshow(img)
```

```
sample_images(r'C:\Users\bramee\OneDrive\Desktop\trainn',images[::120])
```

```
# Let's now save to a new file

photos ,label = [],[]

def combine_all(folder):
 for file in listdir(folder):
    # determine class
    output = 0.0
    if file.startswith('j'):        # jeans
        output = 1.0
    elif file.startswith('s'):      # sarees
        output = 2.0
    else: # file.startswith('t'):   #trousers
        output = 3.0
    # load image

    photo = load_img(folder +'/'+ file ,target_size=(200,200))
    # convert to numpy array
    photo = img_to_array(photo)
    # storing
    photos.append(photo)
    label.append(output)
```

```
combine_all(r'C:\Users\bramee\OneDrive\Desktop\trainn')
```

```
# convert to numpy array
photos = asarray(photos)
label = asarray(label)

print(photos.shape, label.shape)
```

```
(924, 200, 200, 3) (924,)
```

```
# save the reshaped photos
save("amazon_photos.npy",photos)
save('amazon_labels.npy',label)
```

Now we assign independent and target variable i.e., x and y respectively. The y variable defines the category of image, which is Saree, Jeans or Trouser.

After this the most important step is to rescale our independent data. We are rescaling the data into 255 scale.

```python
# defining independent and target variables
x = photos
y = label
```

```python
print(x.shape)
print(y.shape)
```

```
(924, 200, 200, 3)
(924,)
```

```python
y.astype(int)
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

```python
y = np_utils.to_categorical(y-1, 3)
y.shape
```

```
(924, 3)
```

```python
# Rescaling the image
x = x/255
x
```

```
array([[[[1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         ...,
         [0.5058824 , 0.4       , 0.33333334],
         [0.7490196 , 0.6784314 , 0.6392157 ],
         [1.        , 0.99215686, 0.972549  ]],
```

- Model Building

In model building phase first we splitted our dataset into train and test data using the train_test_split model selection method.

```
# splitting dataset

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(693, 200, 200, 3) (693, 3)
(231, 200, 200, 3) (231, 3)
```

After splitting are using ImageDataGenerator technique. This technique is used for data argumentation which means to create more data from the present data.

This technique will produce more data using the zoom, rotation etc methods.

```
from keras.preprocessing.image import ImageDataGenerator

# Image data augmentation
augs_gen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.2,
        height_shift_range=0.2,
        horizontal_flip=True,
        vertical_flip=False)

augs_gen.fit(x_train)
```

So, this is CNN (Convolution Neural Network) project, we are using the Sequential method to build our model which is present in TensorFlow module. In this Sequential model we are using 6 step Convolution2d and MaxPooling layers.

In 1st layer of Convolution2d layer we are using 16 filters with a kernel size of (3,3) and the activation method used is “Relu activation”. After this Maxpooling layers follows up. In 2nd layer of Convolution2d we increase the filter size by 2 and so on in other layers.

After convolutional and Maxpooling another layer present is flattening layer Flattening a tensor means to remove all the dimensions except for one. This is exactly what the Flatten layer does.

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),
    # 3 output neurons for 3 classes with the softmax activation
    tf.keras.layers.Dense(3, activation='softmax')
])
```

- Model Summary

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_17 (Conv2D)          (None, 198, 198, 16)      448

 max_pooling2d_17 (MaxPoolin  (None, 99, 99, 16)       0
 g2D)

 conv2d_18 (Conv2D)          (None, 97, 97, 32)        4640

 max_pooling2d_18 (MaxPoolin  (None, 48, 48, 32)       0
 g2D)

 conv2d_19 (Conv2D)          (None, 46, 46, 64)        18496

 max_pooling2d_19 (MaxPoolin  (None, 23, 23, 64)       0
 g2D)

 conv2d_20 (Conv2D)          (None, 21, 21, 64)        36928

 max_pooling2d_20 (MaxPoolin  (None, 10, 10, 64)       0
 g2D)

 conv2d_21 (Conv2D)          (None, 8, 8, 64)          36928

 max_pooling2d_21 (MaxPoolin  (None, 4, 4, 64)         0
 g2D)

 flatten_5 (Flatten)         (None, 1024)              0

 dense_10 (Dense)            (None, 128)               131200

 dense_11 (Dense)            (None, 3)                 387

=================================================================
Total params: 229,027
Trainable params: 229,027
Non-trainable params: 0
_________________________________________________________________
```

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

```
# Fit the model
history = model.fit(x_train,y_train, epochs=10, validation_split=0.2, # taking 20 percent of training set for validation
                    callbacks = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy', patience=3))
```

```
Epoch 1/10
18/18 [==============================] - 11s 638ms/step - loss: 0.2899 - accuracy: 0.8736 - val_loss: 0.4257 - val_accuracy: 0.
8129
Epoch 2/10
18/18 [==============================] - 12s 641ms/step - loss: 0.2403 - accuracy: 0.8899 - val_loss: 0.4022 - val_accuracy: 0.
8273
Epoch 3/10
18/18 [==============================] - 11s 637ms/step - loss: 0.2083 - accuracy: 0.9116 - val_loss: 0.4964 - val_accuracy: 0.
8058
Epoch 4/10
18/18 [==============================] - 12s 645ms/step - loss: 0.1939 - accuracy: 0.8989 - val_loss: 0.4045 - val_accuracy: 0.
8129
Epoch 5/10
18/18 [==============================] - 11s 631ms/step - loss: 0.1910 - accuracy: 0.9224 - val_loss: 0.4461 - val_accuracy: 0.
7914
```

- Model Evaluations

```
print("Accuracy : ", model.evaluate(x_test, y_test))
```

```
8/8 [==============================] - 1s 125ms/step - loss: 0.3668 - accuracy: 0.8571
Accuracy :  [0.36684566736221313, 0.8571428656578064]
```

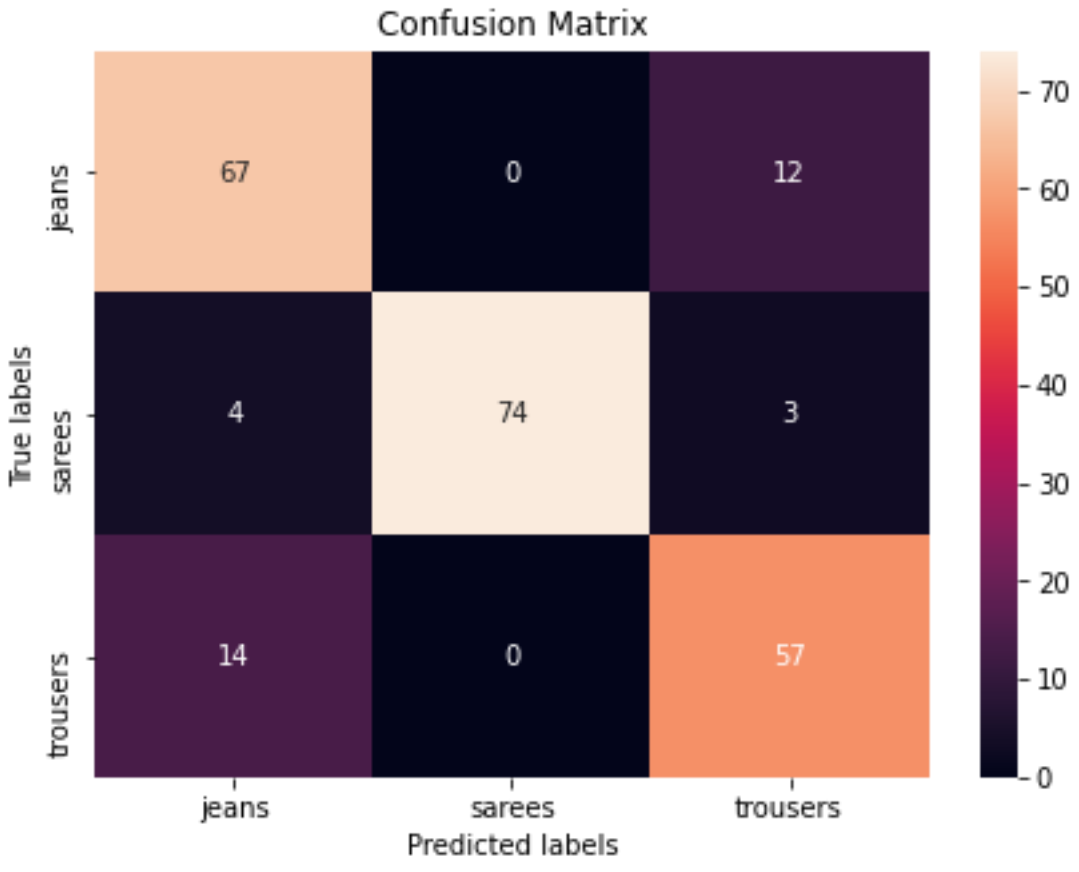
```
# Prediction on the test image
cnn_pred = model.predict(x_test, verbose=1)
cnn_pred = np.argmax(cnn_pred, axis=1) # this will pick the value in an array having the maximum score
```

```
8/8 [==============================] - 1s 93ms/step
```

As accuracy increases, loss will decrease.

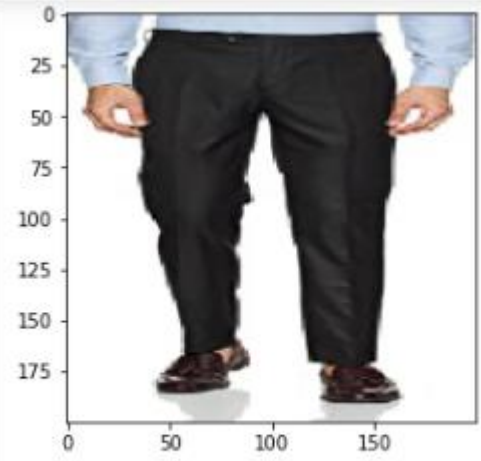
Confusion matrix-

In Confusion matrix, here we are getting good result in classification of sarees, but our model is somehow lacking in accurate classification between Jeans and Trousers.



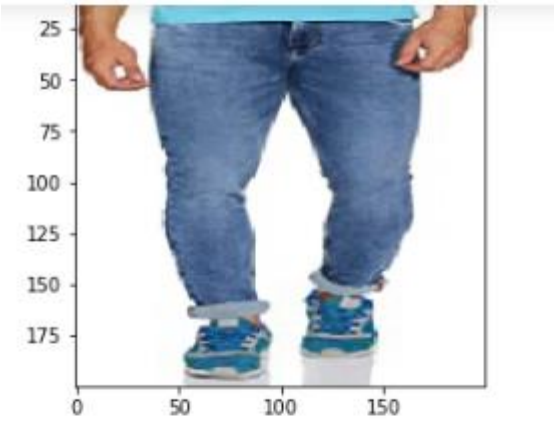
Predictions:

```
test_labels=rounded_labels.tolist() # converting the test_labels into a list

# Creating a function which picks random images and identifies the class to which the image belongs
def get_image_and_class(size):
  idx = np.random.randint(len(x_test), size=size) # generating a random image from the test data
  for i in range(len(idx)):
    plt.imshow(x_test[idx,:][i])
    plt.show()

  # Print the class of the random image picked above
    if test_labels[idx[i]] == 1:
      print('This is a sarees!')
    elif test_labels[idx[i]] == 0:
      print('This is a jeans!')
    elif test_labels[idx[i]] == 2:
      print('This is a trousers!')
```

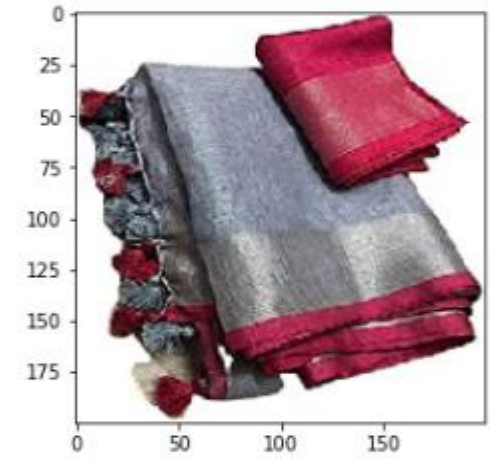



```
This is a trousers!
```


```
This is a trousers!
```




```
This is a jeans!
```




```
This is a sarees!
```


```
This is a jeans!
```

CONCLUSION:

The key finding and the conclusion of the study: -

1. The image data was collected using Web scrapping from Amazon for Jeans, Sarees and Trousers.

2. We have used Convolutional Neural Network for the project and giving us the accuracy of 85% with 0.3 log loss.

Thank You!!!