Image Scraping and Classification

Submitted by:

NARA LOHITH

ACKNOWLEDGMENT

The project entitled “IMAGE SCRAPING AND CLASSIFICATION” is done by me during my internship with Flip Robo Technologies. I am grateful to Data Trained and Flip Robo Technologies for their guidance during this project.
Other reference websites used to complete this project are:

1. Data scraped from amazon.
2. Towardsdatascience.com
3. Stackoverflow.com
4. Datacamp.com

INTRODUCTION

- Conceptual Background of the Domain Problem

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 you 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.

This task is divided into two phases: Data Collection and Mode Building.

Data Collection Phase: In this section, you need to scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:

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

Model Building Phase: After the data collection and preparation is done, you need to build an image classification model that will classify between these 3 categories mentioned above. You can play around with optimizers and learning rates for improving your model's performance.

Problem Statement

- You need to scrape images of these 3 categories and build your data from it. That data will be provided as an input to your deep learning problem. You need to scrape minimum 200 images of each categories. There is no maximum limit to the data collection. You are free to apply image augmentation techniques to increase the size of your data but make sure the quality of data is not compromised.
- Remember, in case of deep learning models, the data needs to be big for building a good performing model. More the data, better the results.

Analytical Problem Framing

- ## Data Collection

We have collected/scraped data from amazon for the three categories: Sarees, Jeans and Trousers.

In total there are 343 rows for each of the items as shown below:

Data Collection Phase

```
n [1]: #importing all the required libraries
       import selenium
       import pandas as pd
       from selenium import webdriver # Importing selenium webdriver
       import time

       # Importing required Exceptions which needs to handled
       from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException

n [2]: #creating empty lists
       Sarees_Image=[]
       Trousers_Image=[]
       Jeans_Image=[]

n [3]: #search_list=['sarees','trousers','jeans']

n [4]: url1='https://www.amazon.com/'

n [5]: ver1 = webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
       ver_1.get(url1)

       earching required fields
       rch_bar = driver_1.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
       rch_bar.send_keys('sarees')

       rch_button = driver_1.find_element_by_xpath('//span[@id="nav-search-submit-text"]/input')       # Locating search_bu
       rch_button.click()


       rt_page = 0
       _page = 5
       i in range(start_page,end_page+1):
        sar=driver_1.find_elements_by_xpath("//div[@class='a-section aok-relative s-image-square-aspect']/img")
        for j in sar:
            image=j.get_attribute('src')
            Sarees_Image.append(image)
```

```
In [8]: driver1 = webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
        driver_1.get(url1)

        # searching required fields
        search_bar = driver_1.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
        search_bar.send_keys('trousers')

        search_button = driver_1.find_element_by_xpath('//span[@id="nav-search-submit-text"]/input')       # Locating sear
        search_button.click()


        start_page = 0
        end_page = 5
        for i in range(start_page,end_page+1):
            sar=driver_1.find_elements_by_xpath("//div[@class='a-section aok-relative s-image-square-aspect']/img")
            for j in sar:
                image=j.get_attribute('src')
                Trousers_Image.append(image)

            nxt_button=driver_1.find_element_by_xpath("//li[@class='a-last']/a").click()
            #driver_1.get(nxt_button[i].get_attribute('href'))
            time.sleep(5)
```

```
10]: len(Trousers_Image)

10]: 358

11]: driver1 = webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
     driver_1.get(url1)

     # searching required fields
     search_bar = driver_1.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
     search_bar.send_keys('jeans')

     search_button = driver_1.find_element_by_xpath('//span[@id="nav-search-submit-text"]/input')       # Locating s
     search_button.click()


     start_page = 0
     end_page = 5
     for i in range(start_page,end_page+1):
         sar=driver_1.find_elements_by_xpath("//div[@class='a-section aok-relative s-image-square-aspect']/img")
         for j in sar:
             image=j.get_attribute('src')
             Jeans_Image.append(image)

         nxt_button=driver_1.find_element_by_xpath("//li[@class='a-last']/a").click()
         #driver_1.get(nxt_button[i].get_attribute('href'))
         time.sleep(5)
```

```
In [14]: Sarees=[]
         Sarees=Sarees_Image[0:343]
         len(Sarees)

Out[14]: 343

In [15]: Trousers=[]
         Trousers=Trousers_Image[0:343]
         len(Trousers)

Out[15]: 343

In [16]: Jeans=[]
         Jeans=Jeans_Image[0:343]
         len(Jeans)

Out[16]: 343
```

- 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.

```
In [24]: # Creating Directories
         import os

         def directory(dir):
             current_path=os.getcwd()
             new=os.path.join(current_path,dir)
             if not os.path.exists(new):
                 os.makedirs(new)

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

```
# Downloading images

import shutil
import requests
```

In [21]:
```
for index, link in enumerate(Sarees):
    print('Downloading {0} of 343 saree images'.format(index+1))
    response=requests.get(link)
    with open('Sarees_Images/img{0}.jpeg'.format(index+1),"wb") as file:
        file.write(response.content)
```

```
Downloading 1 of 343 saree images
Downloading 2 of 343 saree images
Downloading 3 of 343 saree images
Downloading 4 of 343 saree images
Downloading 5 of 343 saree images
Downloading 6 of 343 saree images
Downloading 7 of 343 saree images
Downloading 8 of 343 saree images
Downloading 9 of 343 saree images
Downloading 10 of 343 saree images
Downloading 11 of 343 saree images
Downloading 12 of 343 saree images
Downloading 13 of 343 saree images
Downloading 14 of 343 saree images
```

In [18]:
```
for index, link in enumerate(Trousers):
    print('Downloading {0} of 343 trouser images'.format(index+1))
    response=requests.get(link)
    with open('Trousers_Images/img{0}.jpeg'.format(index+1),"wb") as file:
        file.write(response.content)
```

```
Downloading 1 of 343 trouser images
Downloading 2 of 343 trouser images
Downloading 3 of 343 trouser images
Downloading 4 of 343 trouser images
Downloading 5 of 343 trouser images
Downloading 6 of 343 trouser images
Downloading 7 of 343 trouser images
Downloading 8 of 343 trouser images
Downloading 9 of 343 trouser images
Downloading 10 of 343 trouser images
Downloading 11 of 343 trouser images
Downloading 12 of 343 trouser images
Downloading 13 of 343 trouser images
Downloading 14 of 343 trouser images
Downloading 15 of 343 trouser images
Downloading 16 of 343 trouser images
Downloading 17 of 343 trouser images
Downloading 18 of 343 trouser images
Downloading 19 of 343 trouser images
Downloading 20 of 343 trouser images
```

In [25]:
```
for index, link in enumerate(Jeans):
    print('Downloading {0} of 343 jeans images'.format(index+1))
    response=requests.get(link)
    with open('Jeans_Images/img{0}.jpeg'.format(index+1),"wb") as file:
        file.write(response.content)
```

```
Downloading 1 of 343 jeans images
Downloading 2 of 343 jeans images
Downloading 3 of 343 jeans images
Downloading 4 of 343 jeans images
Downloading 5 of 343 jeans images
Downloading 6 of 343 jeans images
Downloading 7 of 343 jeans images
Downloading 8 of 343 jeans images
Downloading 9 of 343 jeans images
```

Model/s Development and Evaluation

After collecting the data we next do training of the data. For that firstly, we created a main folder called “Clothes” in current working directory inside which I further created two folders called “Train” and “Test”.

In Train folder we have kept 300 images from each clothing category and remaining 43 images we have kept in Test folder for each category. Hence, we got 900 images for training and 129 for testing.

In [3]:
```
import os
from os import listdir

#train=r'Sarees_Images'
```

In [4]:
```
train_data=r'Clothes/Train'
test_data=r'Clothes/Test'
```

In [5]:
```
# Let's try to print some of the scrapped images from each category
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

train_jeans=r'Clothes/Train/Jeans_Images'
train_saree=r'Clothes/Train/Sarees_Images'
train_trouser=r'Clothes/Train/Trousers_Images'


Cloth_train=[train_jeans, train_saree, train_trouser]
for dirs in Cloth_train:
    k=listdir(dirs)
    for i in k[:3]:
        img=mpimg.imread('{}/{}'.format(dirs,i))
        plt.imshow(img)
        plt.axis('off')
        plt.show()
```

In [8]:
```
print("Count of Training Images")
print("No.of Images of Sarees in train dataset -> ",len(os.listdir(r'Clothes/Train/Sarees_Images')))
print("No.of Images of Jeans in train dataset -> ",len(os.listdir(r'Clothes/Train/Jeans_Images')))
print("No.of Images of Trousers in train dataset ->",len(os.listdir(r'Clothes/Train/Trousers_Images')))
"\n"

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

```
Count of Training Images
No.of Images of Sarees in train dataset ->  300
No.of Images of Jeans in train dataset ->  300
No.of Images of Trousers in train dataset -> 300
Count of Test Images
No.of Images of Sarees in test dataset->  43
No.of Images of Jeans in test dataset -> 43
No.of Images of Trousers in test dataset->  43
```

Next we defined dimensions of images and other parametrs also. Then, for data augmentation we defined training and testing set.

In [9]:
```
import pandas as pd
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout, Activation, Conv2D, MaxPooling2D,BatchNormalization
from tensorflow.keras import optimizers
from keras.models import load_model
from tensorflow.keras.preprocessing import image
import random
import scipy
import pylab as pl
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import warnings
warnings.filterwarnings("ignore")
```

In [10]:
```
input_shape=(128,128,3)
img_width=128
img_height=128
batch_size=12
epoch=100
train_samples=300
test_samples=43
```

In [11]:
```
# Data Augmentation on Training Images

Train_datagen=ImageDataGenerator(rescale=1./255,
                                 zoom_range=0.2,
                                 rotation_range=30,
                                 horizontal_flip=True)
Training_set=Train_datagen.flow_from_directory(train_data,
                                               target_size=(img_width,img_height),
                                               batch_size=batch_size,
                                               class_mode='categorical')

# Validation Data Generator
Test_datagen=ImageDataGenerator(rescale=1./255)
Test_set=Test_datagen.flow_from_directory(test_data,
                                          target_size=(img_width,img_height),
                                          batch_size=batch_size,
                                          class_mode='categorical')
```

```
Found 900 images belonging to 3 classes.
Found 129 images belonging to 3 classes.
```

In [12]:
```
# 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"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      896
_________________________________________________________________
activation (Activation)      (None, 126, 126, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
activation_1 (Activation)    (None, 61, 61, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 128)               295040
_________________________________________________________________
activation_4 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_5 (Activation)    (None, 3)                 0
=================================================================
Total params: 360,995
Trainable params: 360,995
Non-trainable params: 0
```

- Then we defined Early Stop criteria and saved the model as ‘best.h5’ for the best results.

In [13]:
```
# Defining Early stopping and Model check point
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

ES = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=30)
MC = ModelCheckpoint('best.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
```

In [14]:
```
# Fitting the Training Data
history = model.fit(
    Training_set,
    epochs=epoch,
    validation_data=Test_set,
    validation_steps=test_samples//batch_size,
    steps_per_epoch=train_samples//batch_size,
    callbacks=[ES,MC])
```

```
Epoch 00048: val_accuracy did not improve from 0.91667
Epoch 49/100
25/25 [==============================] - 61s 2s/step - loss: 0.3374 - accuracy: 0.8467 - val_loss: 0.2337 - val_accuracy: 0.972
2

Epoch 00049: val_accuracy improved from 0.91667 to 0.97222, saving model to best.h5
Epoch 50/100
```

In [15]:
```
model.save('best_model.h5')
```

In [16]:
```
losses = pd.DataFrame(model.history.history)
losses
```

Out[16]:

	loss	accuracy	val_loss	val_accuracy
0	1.094139	0.463333	1.051300	0.416667
1	1.027087	0.500000	1.013963	0.583333
2	0.817383	0.606667	0.783595	0.555556
3	0.735359	0.586667	0.744196	0.555556
4	0.591579	0.716667	0.732184	0.722222
...	...	...	...	...
74	0.323508	0.880000	1.154527	0.611111
75	0.443401	0.833333	0.352910	0.861111
76	0.363662	0.830000	0.283024	0.944444
77	0.367861	0.856667	0.309966	0.888889
78	0.392140	0.846667	0.243337	0.944444

79 rows × 4 columns

Prediction

- Now, we load our saved model and do prediction of our test images.
- For that we first load all the test images from test folder created in main folder Clothes as follows:

In [17]:
```
saved_model = load_model('best_model.h5')
```

In [18]:
```
test_jeans=r'Clothes/Test/Jeans_Images'
test_saree=r'Clothes/Test/Sarees_Images'
test_trouser=r'Clothes/Test/Trousers_Images'
```

- Then we predicted the images and displayed it as shown:

In [19]:
```
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=(128, 128))
        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 = saved_model.predict(test_image)
        print("Predicted Label is:",np.argmax(result, axis=1),"\n")
```

Input Image is: img301.jpeg

Predicted Label is: [2]

Input Image is: img302.jpeg

Predicted Label is: [0]

Input Image is: img301.jpeg

Predicted Label is: [1]

Input Image is: img302.jpeg

Predicted Label is: [1]

Input Image is: img303.jpeg

CONCLUSION

- Our model is working well and gave accuracy of 97%.
- It was able to classify the three clothing items distinctly.
- Since in all three categories there were some extra/unnecessary items other than the main items hence, it could have been removed and we could have got better result. Moreover, training data could have been increased.