

IMAGE SCRAPING AND CLASSIFICATION PROJECT

Submitted by:

ssudhakar0108@gmail.com

**ACKNOWLEDGMENT**

<https://www.selenium.dev/documentation/en/>

<https://www.tensorflow.org/>

**INTRODUCTION**

* Business Problem Framing

**Problem Statement:**

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)
* 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.
* **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.
* Submission Details:
* In form of submission, you need to share the following:
* Zipped file of your data directory containing the scraped images of each category.
* Web Scraping script used for scraping the images.
* Jupyter Notebook that contains the mode building section.
* A detailed report of the project and its implementation.
* A power point presentation for the project.
* Motivation for the Problem Undertaken

this assignment was provided by FLIPROBO technologies.

**Analytical Problem Framing**

* **webscraping**
* **import libraries:**
* **# Importing Libraries**
* **imp# Importing Libraries**
* **import pandas as pd**
* **from bs4 import BeautifulSoup**
* **import requests**
* **import selenium**
* **from selenium import webdriver**
* **import time**
* **from selenium.webdriver.support.ui import WebDriverWait**
* **from selenium.webdriver.common.by import By**
* **from selenium.webdriver.support.ui import WebDriverWait**
* **from selenium.webdriver.support import expected\_conditions as ECort pandas as pd**
* **from bs4 import BeautifulSoup**
* **import requests**

**driver=webdriver.Chrome(r"/home/sudhakar/Downloads/chromedriver")**

**time.sleep(5)**

**driver.get("https://www.amazon.in" )**

**time.sleep(5)**

**search = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[type='text']")))**

**search.click()**

**search.send\_keys(input())**

**button1 = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[id='nav-search-submit-button']"))).click()**

**sraping model:**

**images = driver.find\_elements\_by\_xpath('//img[@class="s-image"]')**

**img\_urls=[]**

**for i in images:**

**img\_urls.append(i.get\_attribute("src"))**

**for i in range(len(img\_urls)):**

**if i >= 100:**

**break**

**print("Downloading {0} of {1} images" .format(i, 100))**

**response= requests.get(img\_urls[i])**

**file = open("/home/sudhakar/Downloads/amazonsaree/img"+str(i)+".jpg", "wb")**

**file.write(response.content)**

**nex = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "li[class='a-last']"))).click()**

**downloding images:**

**Downloading 0 of 100 images**

Downloading 1 of 100 images

Downloading 2 of 100 images

Downloading 3 of 100 images

Downloading 4 of 100 images

Downloading 5 of 100 images

Downloading 6 of 100 images

Downloading 7 of 100 images

Downloading 8 of 100 images

Downloading 9 of 100 images

Downloading 10 of 100 images

Downloading 11 of 100 images

Downloading 12 of 100 images

Downloading 13 of 100 images

Downloading 14 of 100 images

Downloading 15 of 100 images

Downloading 16 of 100 images

Downloading 17 of 100 images

Downloading 18 of 100 images

Downloading 19 of 100 images

Downloading 20 of 100 images

Downloading 21 of 100 images

Downloading 22 of 100 images

Downloading 23 of 100 images

Downloading 24 of 100 images

Downloading 25 of 100 images

Downloading 26 of 100 images

Downloading 27 of 100 images

Downloading 28 of 100 images

Downloading 29 of 100 images

Downloading 30 of 100 images

Downloading 31 of 100 images

Downloading 32 of 100 images

Downloading 33 of 100 images

Downloading 34 of 100 images

Downloading 35 of 100 images

Downloading 36 of 100 images

Downloading 37 of 100 images

Downloading 38 of 100 images

Downloading 39 of 100 images

Downloading 40 of 100 images

Downloading 41 of 100 images

Downloading 42 of 100 images

Downloading 43 of 100 images

Downloading 44 of 100 images

Downloading 45 of 100 images

Downloading 46 of 100 images

Downloading 47 of 100 images

Downloading 48 of 100 images

Downloading 49 of 100 images

Downloading 50 of 100 images

Downloading 51 of 100 images

Downloading 52 of 100 images

Downloading 53 of 100 images

Downloading 54 of 100 images

Downloading 55 of 100 images

Downloading 56 of 100 images

Downloading 57 of 100 images

Downloading 58 of 100 images

Downloading 59 of 100 images

Downloading 60 of 100 images

Downloading 61 of 100 images

Downloading 62 of 100 images

Downloading 63 of 100 images

 [9]:

**Model/s Development and Evaluation**

* Identification of possible problem-solving approaches (methods)

using google colab:

from google.colab import drive

drive.mount('/content/drive')

import cv2,os

data\_path='/content/drive/MyDrive/image\_classification\_dataset/IMAGE\_CLASSIFICATION/'

categories=os.listdir(data\_path)

labels=[i for i in range(len(categories))]

label\_dict=dict(zip(categories,labels)) #empty dictionary

print(label\_dict)

print(categories)

print(labels)

img\_size=100

data=[]

target=[]

for category in categories:

folder\_path=os.path.join(data\_path,category)

img\_names=os.listdir(folder\_path)

for img\_name in img\_names:

img\_path=os.path.join(folder\_path,img\_name)

img=cv2.imread(img\_path)

try:

resized=cv2.resize(img,(img\_size,img\_size))

#resizing the image into 100x100, since we need a fixed common size for all the images in the dataset

data.append(resized)

target.append(label\_dict[category])

#appending the image and the label(categorized) into the list (dataset)

except Exception as e:

print('Exception:',e)

#if any exception rasied, the exception will be printed here. And pass to the next image

* Testing of Identified Approaches (Algorithms)

Limport numpy as np

data=np.array(data)/255.0

data=np.reshape(data,(data.shape[0],img\_size,img\_size,3))

target=np.array(target)

from keras.utils import np\_utils

new\_target=np\_utils.to\_categorical(target)

* Run and Evaluate selected models

from keras.models import Sequential

from keras.layers import Dense,Activation,Flatten,Dropout

from keras.layers import Conv2D,MaxPooling2D

from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(200,(3,3),input\_shape=data.shape[1:]))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(100,(3,3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#The second convolution layer followed by Relu and MaxPooling layers

model.add(Flatten())

model.add(Dropout(0.5))

#Flatten layer to stack the output convolutions from second convolution layer

model.add(Dense(50,activation='relu'))

#Dense layer of 64 neurons

model.add(Dense(3,activation='softmax'))

#The Final layer with two outputs for three categories

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

from sklearn.model\_selection import train\_test\_split

train\_data,test\_data,train\_target,test\_target=train\_test\_split(data,new\_target,test\_size=0.1)

#checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val\_loss',verbose=0,save\_best\_only=True,mode='auto')

history=model.fit(train\_data,train\_target,epochs=200,validation\_split=0.2)

* Key Metrics for success in solving problem under consideration

ACCURACY,LOSS,VALUE LOSS are key

* Visualizations

# plot the training loss and accuracy

N = 200

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), history.history["loss"], label="train\_loss")

plt.plot(np.arange(0, N), history.history["val\_loss"], label="val\_loss")

plt.plot(np.arange(0, N), history.history["accuracy"], label="train\_acc")

plt.plot(np.arange(0, N), history.history["val\_accuracy"], label="val\_acc")

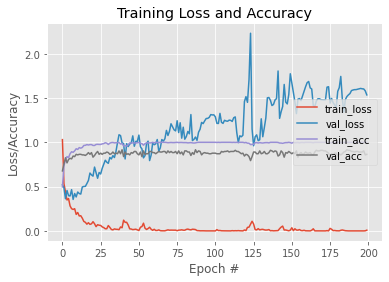
plt.title("Training Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="center right")

plt.savefig("CNN\_Model")



Interpretation of the Results

testing accuracy was good with 90 to 100 percent

**CONCLUSION**

we have developed a deeplearning model using google colab with accuracy more than 90 percent.