IMAGE SCRAPING AND CLASSIFICATION PROJECT

Submitted by: Neeraj Kumar S

ACKNOWLEDGMENT

I, Neeraj Kumar S- the intern Data Science of FlipRobo is overwhelmed in all humbleness and gratefulness to acknowledge my deep gratitude to all those who have helped me put my ideas to perfection and have assigned tasks, well above the level of simplicity and into something concrete and unique. I, wholeheartedly thank my SME Miss.Swati Mahaseth for having faith in me, whose active interest in the project & insight helped us to formulate, redefine, implement our approach to the project. Separately, I am also thankful to our institute -Data Trained & others seen unseen hands which have given us direct& indirect help in completion of this project. With help of their brilliant guidance and encouragement, I was able to complete my tasks properly and were up to the mark in all the tasks assigned. During the process, I got a chance to see the stronger side of my technical and non-technical aspects and also strengthen my concepts.

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.

Multiclass image classification is a common task in computer vision, where we categorize an image into three or more classes.

Web scraping is the process of using bots to extract content and data from a **website**. Unlike screen **scraping**, which only copies pixels displayed on screen, **web scraping** extracts underlying HTML code and, with it, data stored in a database. The **scraper** can then replicate entire **website** content elsewhere.

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
- Model Building.

The data in the form of images are scraped from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:

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

ANALYTICAL PROBLEM FRAMING

Beginning of the Code

The code in this project is written in Python 3.6.6-Anaconda3. To begin the project, importing essential libraries are extremely initial and the very base step as it provides fast, expressive, and flexible data structures to easily work with.

The libraries includes :

For scraping the images

```
# Importing necessary lbiraries

import pandas as pd
import numpy as np
import selenium
from selenium import webdriver
import shutil
import requests
from selenium.webdriver.chrome.options import Options
import os
import time
from selenium.common.exceptions import NoSuchElementException
```

For Deep learning

```
# Importing necessary libraries

import cv2
import numpy as np
import pandas as pd
import os
from PIL import Image
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers


%matplotlib inline
```

Data Collection

For scraping the data **Selenium** library is used.

The data is scraped from **Amazon.com**, categories being :

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

A directory is created inorder to save the scraped images.

The images are first scraped, downloaded and then saved in the directory formed in the .jpg format.

The labels were also assigned to the respective categories.

As a next step, the list of labels is converted into a dataframe using *to_DataFrame()*. And the same has been then saved in a csv file.

For Directory

```
# 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
```

For scraping, downloading and saving the images.

```
# Function for scraping, downloading and storing the required images/data.

def images(url, folder, labels, label):

    # Opening drive
    chrome_options = Options()
    chrome_options.add_argument("--incognito")
    driver = webdriver.Chrome(r"D:\sanss_data\FlipRobo\chromedriver.exe", options=chrome_options)
    amazon_url = driver.get(url)

    time.sleep(3)

    # Locating the search bar
    search_bar = driver.find_element_by_id("twotabsearchtextbox")
    search_bar.clear()
    category = input('Search Category : ')
    search_bar.send_keys(category)

    #Locating the button and clicking it to search the category
    button = driver.find_element_by_id('nav-search-submit-button')
    button.click()

    time.sleep(3)

    # Creating empty list fto sotre data
    image_urls = []

    # Loop for iterating the pages and thus scraping the images
    for page in range(1, 6):
        print('\nPage', page)

        time.sleep(3)

        # Giving path of the images to be scrapped
        images = driver.find_elements_by_xpath("//img[@class = 's-image']")

        # Getting the source/link of the images.
        for img in images:
            source = img.get_attribute('src')
            image_urls.append(source)

        # Downloading the images and saving the same in the respective directory.
        for i, image_link in enumerate(image_urls):
            response = requests.get(image_link)

            # Downloading the images
            image_name = folder + '/' + category + '_' + str(i+1) + '.jpg'
            with open(image_name, 'wb') as file:
                file.write(response.content)

        time.sleep(3)

        print("Downloads of images from Page", page, "is completed! \n")

        time.sleep(3)

        # Moving to the next page.
        next_page = driver.find_element_by_xpath("//li[@class = 'a-last']/a")
        next_page.click()

    print("Total images downlaoded of", str(category), ": ", len(image_urls))
    for i in image_urls:
        labels.append(label)
```

Splitting the data

In a new notebook, the images and the csv file are loaded and then the same has been splitted into t% and raining and testing data as ‘X’ and ‘y’ as an array. The splitting ratio is of % respectively between train dataset and test dataset.

```
# Checking the shape of the splitted data.

print ("number of training examples = " + str(X_train.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
print ("y_train shape: " + str(y_train.shape))
print ("X_test shape: " + str(X_test.shape))
print ("y_test shape: " + str(y_test.shape))
```

```
number of training examples = 548
number of test examples = 137
X_train shape: (548, 224, 224, 3)
y_train shape: (548, 1)
X_test shape: (137, 224, 224, 3)
y test shape: (137, 1)
```

MODEL/S DEVELOPMENT AND EVALUATION

In multi-class classification, the neural network has the same number of output nodes as the number of classes. Each output node belongs to some class and outputs a score for that class.

Thus, a CNN model is built.

Activation Functions

In the model,

- The **'relu'** activation function is used in the hidden layers.
- **'softmax'** activation function is used in the final or output layer.

Metrics

In order to be able to evaluate the performance of each algorithm, several metrics are defined accordingly, and are discussed briefly in this section.

- **Accuracy**: This metric measures how many of the comments are labeled correctly. However, in our data set, where most of the comments are not toxic, regardless of performance of the model, a high accuracy was achieved.

$$\text{Precision} := TP + TN/ N' + P'$$

Optimizer

The optimizer used in the model is **'Adam'.**

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best

properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

Loss

In order to compute the loss, **'sparse_categorical_crossentropy'** is used.

This loss is computed when there are two or more label classes. And, the labels are expected to be provided as integers.

Fig. 1

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

# First convolution layer
model.add(Conv2D(32,(3,3),input_shape=X.shape[1:]))
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())
```

Fig. 2

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 222, 222, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 32)      9248
_________________________________________________________________
activation_2 (Activation)    (None, 109, 109, 32)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 52, 52, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 26, 26, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
activation_4 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 128)               1179776
_________________________________________________________________
activation_5 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_6 (Activation)    (None, 3)                 0
=================================================================
Total params: 1,245,731
Trainable params: 1,245,731
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. 3

```
# Fitting the model

model.fit(X, y, batch_size=32, epochs=30, validation_split=0.1)
```

Results

By fitting the model, the good accuracy is achieved being **90.32% at 30th epoch.**

```
Epoch 20/30
20/20 [==============================] - 29s 1s/step - loss: 0.2604 - accuracy: 0.8933 - val_loss: 1.1799 - val_accuracy: 0.463
8
Epoch 21/30
20/20 [==============================] - 29s 1s/step - loss: 0.3133 - accuracy: 0.8721 - val_loss: 1.3113 - val_accuracy: 0.173
9
Epoch 22/30
20/20 [==============================] - 29s 1s/step - loss: 0.3237 - accuracy: 0.8688 - val_loss: 1.0591 - val_accuracy: 0.434
8
Epoch 23/30
20/20 [==============================] - 29s 1s/step - loss: 0.3504 - accuracy: 0.8685 - val_loss: 1.0657 - val_accuracy: 0.376
8
Epoch 24/30
20/20 [==============================] - 29s 1s/step - loss: 0.2618 - accuracy: 0.8740 - val_loss: 1.2246 - val_accuracy: 0.391
3
Epoch 25/30
20/20 [==============================] - 30s 1s/step - loss: 0.2868 - accuracy: 0.8870 - val_loss: 1.2766 - val_accuracy: 0.347
8
Epoch 26/30
20/20 [==============================] - 30s 1s/step - loss: 0.2468 - accuracy: 0.8940 - val_loss: 1.3023 - val_accuracy: 0.434
8
Epoch 27/30
20/20 [==============================] - 29s 1s/step - loss: 0.2305 - accuracy: 0.8919 - val_loss: 1.2789 - val_accuracy: 0.463
8
Epoch 28/30
20/20 [==============================] - 29s 1s/step - loss: 0.2835 - accuracy: 0.8757 - val_loss: 0.9833 - val_accuracy: 0.623
2
Epoch 29/30
20/20 [==============================] - 29s 1s/step - loss: 0.2421 - accuracy: 0.8939 - val_loss: 0.8430 - val_accuracy: 0.724
6
Epoch 30/30
20/20 [==============================] - 29s 1s/step - loss: 0.2645 - accuracy: 0.9032 - val_loss: 1.1613 - val_accuracy: 0.521
7
```

An observation is made that with the increase in epoch the accuracy of the model is also increasing.

Evaluating the test dataset

On evaluating the test dataset or model, the test dataset accuracy achieved is also satisfactory as per the training accuracy results. On evaluating the model, the accuracy achieved is **89.78%**.

```
# Evaluating the model

preds = model.evaluate(x = X_test, y = y_test, verbose=1)

print()
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
5/5 [==============================] - 1s 259ms/step - loss: 0.2544 - accuracy: 0.8978

Loss = 0.25440308451652527
Test Accuracy = 0.8978102207183838
```

CONCLUSION

The task of classifying images requires in-depth knowledge of the domain and expertise to identify anomalies in the image. In this project, a multi-class classification problem is solved. As the motive is to build the deep learning model in order to classify among the categories of images.

Additionally, the data for classification is created by scraping the same from an e-commerce portal.

Thus, the classification of images is discussed in this project where the data is obtained from Amazon into saree, jeans and trousers, using Deep learning.

Using deep learning convolutional neural networks. While the dataset is effectively solved, it can be used as the basis for learning and practicing how to develop, evaluate, and use convolutional deep learning neural networks for image classification from scratch.