



Image Scraping and Classification Project

Submitted By:-

Pragati Maheshwari

Table of Contents

Acknowledgment	3
Introduction	4
Business Problem Framing	4
Conceptual Background of the Domain Problem.....	4
Review of Literature	5
Motivation for the Problem Undertaken	5
Analytical Problem Framing	5
Model/s Development and Evaluation	9
Identification of possible problem-solving approaches (methods)	Error! Bookmark not defined.
Testing of Identified Approaches (Algorithms)	Error! Bookmark not defined.
Run and Evaluate selected models	9
Key Metrics for success in solving problem under consideration	10
Visualizations	11
Interpretation of the Results.....	Error! Bookmark not defined.
Conclusion.....	15
Key Findings and Conclusions of the Study.....	15
Learning Outcomes of the Study in respect of Data Science	15

Acknowledgment

Following are the external references which I used:

- www.w3school.com
- www.stackoverflow.com
- www.google.com
- www.geeksforgeeks.org
- <https://www.youtube.com/watch?v=j-3vuBynnOE>
- <https://pythonprogramming.net/tensorboard-analysis-deep-learning-python-tensorflow-keras/?completed=/convolutional-neural-network-deep-learning-python-tensorflow-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. 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
- Model Building.

Conceptual Background of the Domain Problem

In this section, the task is provided that we need to scrape images from e-commerce portal, Amazon. The clothing categories used for scraping will be:

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

We need to scrape images of these 3 categories and build your data from it. That data will be provided as an input to a deep learning problem. We need to scrape minimum 200 images of each categories. There is no maximum limit to the data collection. Since we are free to apply image augmentation techniques to increase the size of our data but make us sure the quality of data is not compromised.

Review of Literature

From the dataset I get to know that it is a classification problem and there are three categories which are Sarees, Trousers and Jeans. We have Scrap the images and built the model in deep learning.

Motivation for the Problem Undertaken

Building your own data and perform the task of model building and scraping the images from the online portal from amazone.in. Its was a great experience on working of a type of live project which was this. It's a very interesting and at the same time problematic to working on a dataset. It was a very challenging project.

Analytical Problem Framing

Data Sources and their formats

I got the data from the online portal amazon.in from their I scrap the images of three categories which are Sarees, Jeans and the Trousers Than I make a folder and save all the three categories images. The images are the in the format of .jpeg



The above is the Sarees Images which I have scraped and stored in a folder.



The above is the next categories which is jeans which is also scraped and stored all the images in a folder Jeans.



And the last category which is Trousers and the images of trousers are also scraped and stored in a folder.

Data Pre-processing Done

In the pre-processing stage, we'll prepare the data to be fed to the Keras model. The first step is clearing the dataset of null values. Then, we'll use one-hot encoding to convert categorical variables to numerical variables. Neural Nets work with numerical data, not categorical. We'll also split the data into training and testing set. Finally, we'll scale the data/standardize it so that it ranges from -1 to 1. This standardization helps both train the model better and allows it to converge easier.

Hardware and Software Requirements and Tools Used

- Hardware – Laptop
- Software - anaconda jupyter notebook
- Libraries- numpy, pandas, seaborn, matplotlib.pyplot, warnings,
- from selenium import webdriver :-

Selenium is an open-source web-based automation tool. Python language is used with Selenium for testing. It has far less verbose and easy to use than any other programming language.

The Python APIs empower you to connect with the browser through Selenium

The *selenium.webdriver* module provides all the Web Driver implementations. Currently supported Web Driver implementations are Firefox, Chrome, IE and Remote. The *Keys* class provide keys in the keyboard like RETURN, F1, ALT etc.

- Web driver:- We all know that we need to have browser drivers, .exe files like chromedriver.exe in case of windows environment or binary files like chrome driver in distributions, in order to run our selenium web driver automation scripts on chrome browsers.

And also we need to set the path of these files in our script like below or need to add location to the class path.

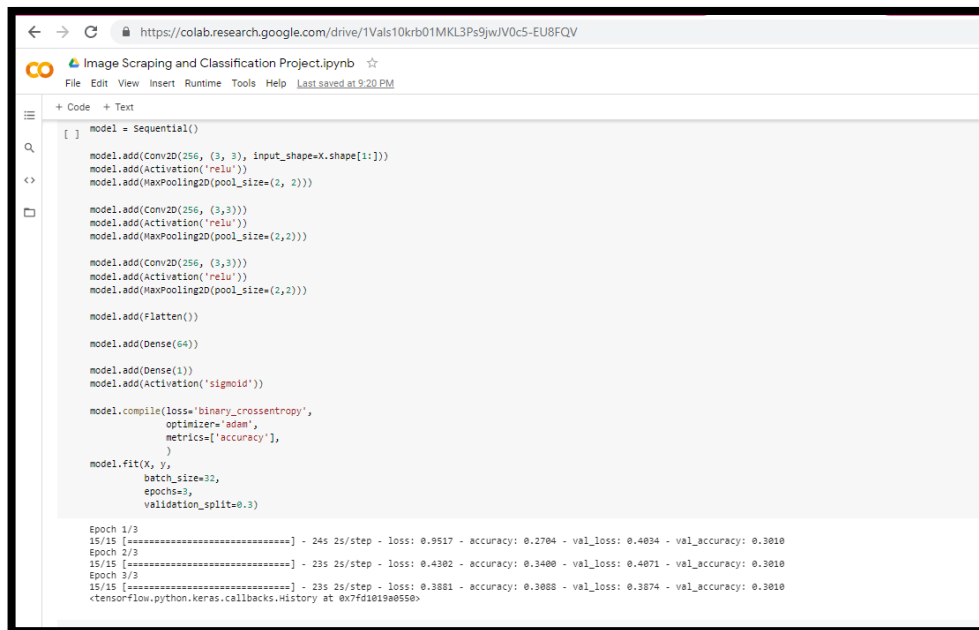
- from bs4 import BeautifulSoup -**Beautiful Soup** is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.
- Import shutil -Python shutil module enables us to operate with file objects easily and without diving into file objects a lot. It takes care of [low-level semantics](#) like creating file objects, closing the files once they are copied and allows us to focus on the business logic of our program
- Import requests:- Requests will allow you to send HTTP/1.1 requests using Python. With it, you can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way.
- Import Tensorflow:- Tensorflow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.
- Import random: - random imports the random module, which contains a variety of things to do with random number generation. Among these is the random () function, which

generates random numbers between 0 and 1. Doing the import this way this requires you to use the syntax `random`.

- **Import cv2:-** OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
- **Import os :-** The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. Path* modules include many functions to interact with the file system.
- **Import tqdm:** - loading a page or doing a transaction, it always eases your mind whenever you see that small progress bar giving you an estimation of how long the process would take to complete or render. If you have a simple progress bar in your script or code, it looks very pleasing to the eye and gives proper feedback to the user whenever he executes the code. You can use the Python external library tqdm, to create simple & hassle-free progress bars which you can add in your code and make it look lively
- **Import pickle:-** Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network
- **Import time:-** The `time()` function returns the number of seconds passed since epoch.

Model/s Development and Evaluation

Run and Evaluate selected models



```
model = Sequential()

model.add(Conv2D(256, (3, 3), input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

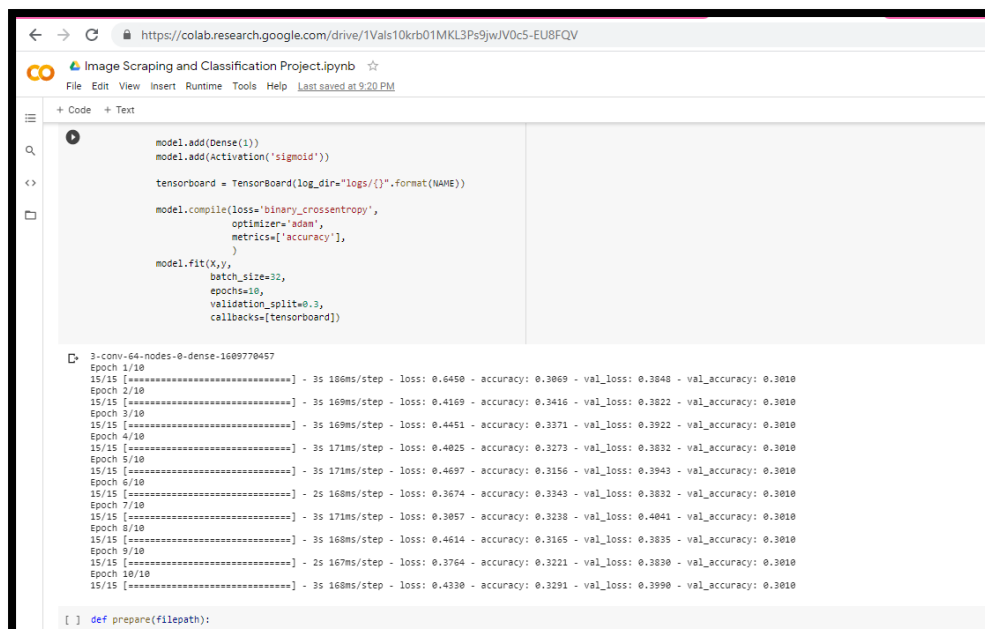
model.add(Dense(64))

model.add(Dense(1))
model.add(Activation('sigmoid'))

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

model.fit(X, y,
        batch_size=32,
        epochs=3,
        validation_split=0.3)

Epoch 1/3
15/15 [=====] - 24s 2s/step - loss: 0.9517 - accuracy: 0.2704 - val_loss: 0.4034 - val_accuracy: 0.3010
Epoch 2/3
15/15 [=====] - 23s 2s/step - loss: 0.4302 - accuracy: 0.3400 - val_loss: 0.4071 - val_accuracy: 0.3010
Epoch 3/3
15/15 [=====] - 23s 2s/step - loss: 0.3881 - accuracy: 0.3088 - val_loss: 0.3874 - val_accuracy: 0.3010
<tensorflow.python.keras.callbacks.History at 0x7fd1019a0550>
```



```
model.add(Dense(1))
model.add(Activation('sigmoid'))

tensorboard = TensorBoard(log_dir="logs/{}".format(NAME))

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

model.fit(X,y,
        batch_size=32,
        epochs=10,
        validation_split=0.3,
        callbacks=[tensorboard])

3-conv-64-nodes-0-dense-1609770457
Epoch 1/10
15/15 [=====] - 3s 106ms/step - loss: 0.6450 - accuracy: 0.3069 - val_loss: 0.3848 - val_accuracy: 0.3010
Epoch 2/10
15/15 [=====] - 3s 169ms/step - loss: 0.4169 - accuracy: 0.3416 - val_loss: 0.3822 - val_accuracy: 0.3010
Epoch 3/10
15/15 [=====] - 3s 169ms/step - loss: 0.4451 - accuracy: 0.3371 - val_loss: 0.3922 - val_accuracy: 0.3010
Epoch 4/10
15/15 [=====] - 3s 171ms/step - loss: 0.4025 - accuracy: 0.3273 - val_loss: 0.3832 - val_accuracy: 0.3010
Epoch 5/10
15/15 [=====] - 3s 171ms/step - loss: 0.4697 - accuracy: 0.3156 - val_loss: 0.3943 - val_accuracy: 0.3010
Epoch 6/10
15/15 [=====] - 2s 168ms/step - loss: 0.3674 - accuracy: 0.3343 - val_loss: 0.3832 - val_accuracy: 0.3010
Epoch 7/10
15/15 [=====] - 3s 171ms/step - loss: 0.3057 - accuracy: 0.3238 - val_loss: 0.4041 - val_accuracy: 0.3010
Epoch 8/10
15/15 [=====] - 3s 168ms/step - loss: 0.4614 - accuracy: 0.3165 - val_loss: 0.3835 - val_accuracy: 0.3010
Epoch 9/10
15/15 [=====] - 2s 167ms/step - loss: 0.3764 - accuracy: 0.3221 - val_loss: 0.3830 - val_accuracy: 0.3010
Epoch 10/10
15/15 [=====] - 3s 168ms/step - loss: 0.4330 - accuracy: 0.3291 - val_loss: 0.3990 - val_accuracy: 0.3010

[ ] def prepare(filepath):
```

Key Metrics for success in solving problem under consideration

Tensor Board is a handy application that allows you to view aspects of our model, The way that we use Tensor Board with Keras is via a Keras callback. There are actually quite a few Keras callbacks, and you can make your own. Definitely check the others out: [Keras Callbacks](#). For example, Model Checkpoint is another useful one. For now, however, we're going to be focused on the Tensor Board callback.

The Convolution Neural Network gained popularity through its use with image data, and is currently the state of the art for detecting what an image is, or what is contained in the image.

The basic CNN structure is as follows: Convolution -> Pooling -> Convolution -> Pooling -> Fully Connected Layer -> Output

Convolution is the act of taking the original data, and creating feature maps from it. Pooling is down-sampling, most often in the form of "max-pooling," where we select a region, and then take the maximum value in that region, and that becomes the new value for the entire region. Fully Connected Layers are typical neural networks, where all nodes are "fully connected." The convolution layers are not fully connected like a traditional neural network.

directory: Directory where the data is located. If **labels** is "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.

labels: Either "inferred" (labels are generated from the directory structure), or a list/tuple of integer labels of the same size as the number of image files found in the directory. Labels should be sorted according to the alphanumeric order of the image file paths (obtained via **os.walk(directory)** in Python).

label_mode: - 'int': means that the labels are encoded as integers (e.g. for **sparse_categorical_crossentropy** loss). - 'categorical' means that the labels are encoded as a categorical vector (e.g. for **categorical_crossentropy** loss). - 'binary' means that the labels (there can be only 2) are encoded as **float32** scalars with values 0 or 1 (e.g. for **binary_crossentropy**). - None (no labels).

class_names: Only valid if "labels" is "inferred". This is the explicit list of class names (must match names of subdirectories). Used to control the order of the classes (otherwise alphanumerical order is used).

color_mode: One of "grayscale", "rgb", "rgba". Default: "rgb". Whether the images will be converted to have 1, 3, or 4 channels.

batch_size: Size of the batches of data. Default: 32 or 36.

image_size: Size to resize images to after they are read from disk. Defaults to (256, 256). Since the pipeline processes batches of images that must all have the same size, this must be provided.

shuffle: Whether to shuffle the data. Default: True. If set to False, sorts the data in alphanumeric order.

seed: Optional random seed for shuffling and transformations.

validation_split: Optional float between 0 and 1, fraction of data to reserve for validation.

subset: One of "training" or "validation". Only used if **validation_split** is set.

interpolation: String, the interpolation method used when resizing images. Defaults to **bilinear**. Supports **bilinear**, **nearest**, **bicubic**, **area**, **lanczos3**, **lanczos5**, **gaussian**, **mitchellcubic**.

follow_links: Whether to visits subdirectories pointed to by symlinks. Defaults to False.

Visualizations

```
Image Scraping and Classification Project.ipynb
File Edit View Insert Runtime Tools Help Last saved at 9:20 PM

+ Code + Text
Connect Editing

[ ] import pandas as pd
import numpy as np
import os
import tensorflow as tf

import cv2
from tensorflow import keras

from tensorflow.keras.models import Sequential, Model
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import random

plt.figure(figsize=(20,20))

test_folder = r'/content/drive/MyDrive/3images/Sarees'
for i in range(5):
    file = random.choice(os.listdir(r'/content/drive/MyDrive/3images/Sarees'))
    image_path=os.path.join(r'/content/drive/MyDrive/3images/Sarees',file)
    img=mpimg.imread(image_path)
    ax=plt.subplot(1,5,i+1)
    ax.title.set_text(file)
    plt.imshow(img)
```



From the above code I visualized the 5 random sarees images from the directory.

```
Image Scraping and Classification Project.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 9:20 PM

+ Code + Text

DATADIR = r"/content/drive/MyDrive/3images"

CATEGORIES = ["Jeans", "Sarees", "Trousers"]

for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap='gray')
        plt.show()

        break

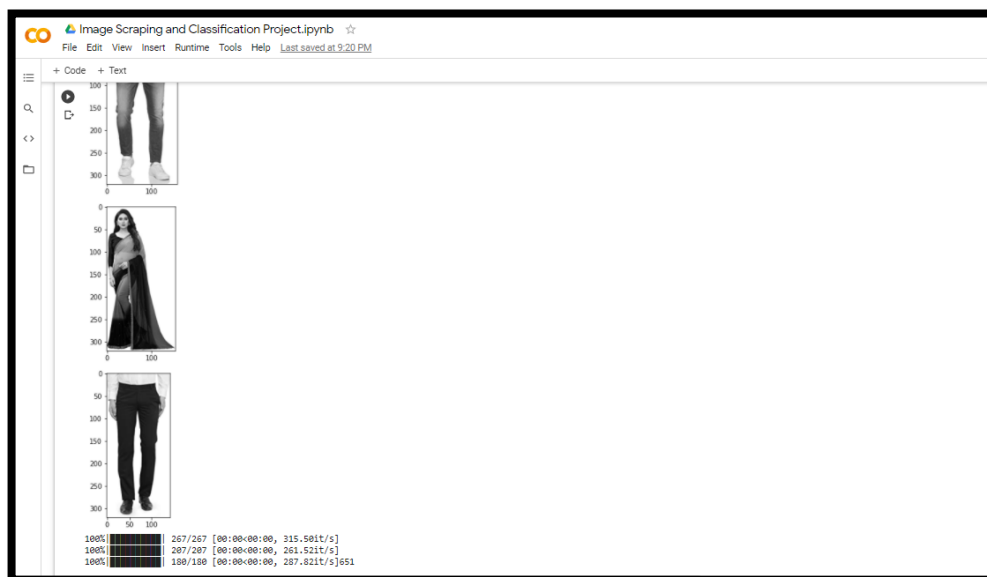
training_data = []

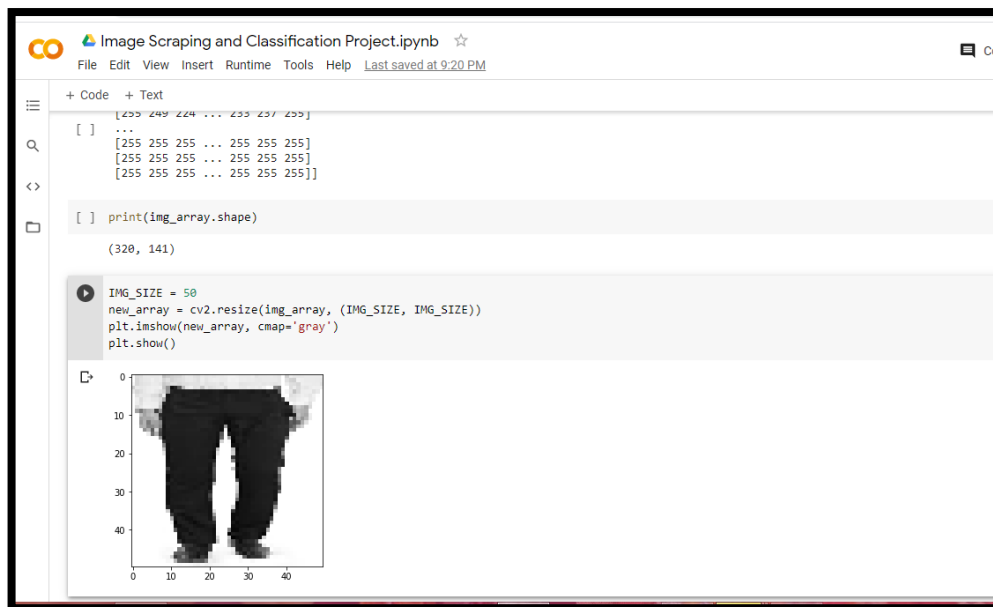
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)

        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])

            except Exception as e:
                pass

create_training_data()
print(len(training_data))
```





Conclusion

Key Findings and Conclusions of the Study

From the dataset which I generated my own through web scraping of the online portal is the finding of sarees jeans and trousers I get to know it is really important to understand the data. I get to know that each feature play a very import role to understand the data. Data format plays a very important role in the visualization and Applying the models and algorithms.

Learning Outcomes of the Study in respect of Data Science

Object detection is a computer vision task that involves both localizing one or more objects within an image and classifying each object in the image. It is a challenging computer vision task that requires both successful object localization in order to locate and draw a bounding box around each object in an image, and object classification to predict the correct class of object that was localized. 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.

This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data.