Image Scraping and Classification Project

Prepared by:

ARCHANA KUMARI

Internship-29

SME Name:

SHWETANK MISHRA

ACKNOWLEDGMENT

I would like to convey my heartfelt gratitude to Flip Robo Technologies for providing me with this wonderful opportunity to work on a Machine Learning project “Image Scraping and Classification Project” and also want to thank my SME “Shwetank Mishra” for providing the dataset and directions to complete this project. This project would not have been accomplished without their help and insights.

I would also like to thank my academic “Data Trained Education” and their team who has helped me to learn Neural Networks and NLP.

Working on this project was an incredible experience as I learnt more from this Project during completion.

INTRODUCTION

1. Business Problem Framing

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision. We use Deep Learning to accomplish the task of image Classification.

In this project we have to classify whether is image is of a jeans, a trouser or a saree. We could we such model either for automatic segmentation of items using IOT and techniques for Industry 4.0.

This project contains two phases:

a. **Data Collection Phase**
b. **Model Building Phase**

2. Conceptual Background of the Domain Problem

Deep learning (DL) is a sub field to the machine learning, capable of learning through its own method of computing. A deep learning model is introduced to persistently break down information with a homogeneous structure like how a human would make determinations. To accomplish this, deep learning utilizes a layered structure of several algorithms expressed as an artificial neural system (ANN). The architecture of an ANN is simulated with the help of the biological neural network of the human brain. This makes the deep learning most capable than the standard machine learning models.

In deep neural networks every node decides its basic inputs by itself and sends it to the next tier on behalf of the previous tier. We train the data in the networks by giving an input image and conveying the network about its output. Neural networks are expressed in terms of number of layers involved for producing the inputs and outputs and the depth of the neural network.

3. Review of Literature

Recently, image classification is growing and becoming a trend among technology developers especially with the growth of data in different parts of industry such as e-commerce, automotive, healthcare, and gaming. The most obvious example of this technology is applied to Facebook. Facebook now can detect up to 98% accuracy in order to identify your face with only a few tagged images and classified it into your Facebook's album. The technology itself almost beats the ability of human in image classification or recognition.

One of the dominant approaches for this technology is deep learning. Deep learning falls under the category of Artificial Intelligence where it can act or think like a human. Normally, the system itself will be set with hundreds or maybe thousands of input data in order to make the 'training' session to be more efficient and faster. It starts by giving some sort of 'training' with all the input data (Faux & Luthon, 2012).

Image classification has become a major challenge in machine vision and has a long history with it. The challenge includes a broad intra-class range of images caused by colour, size, environmental conditions and shape. It is required big data of labelled training images and to prepare this big data, it consumes a lot of time and cost as for the training purpose only. In this project we will be using a transfer learning state of the art model for getting the best results that is VGG 16

4. Motivation for the Problem Undertaken

The problem was undertaken in order to classify images efficiently using best deep learning algorithms and data augmentation techniques.

Analytical Problem Framing

1. Mathematical/ Analytical Modelling of the Problem

1) Scrapped scraped images of all 3 classes that are men's jeans, men's trouser and saree for women from e-commerce website: Amazon.in
2) We have built our model by training it on this data.
3) We have used transfer learning to get state of the art results for our model.

2. Data Sources and their formats

➢ DATA COLLECTION PHASE:

- Scraped images of jeans for men, saree for women and trouser for men from website: Amazon.in using python script with selenium.
- All the data is in the .jpg image format.
- We have scrapped total 2040 images (680 each class).

➢ MODEL BUILDING PHASE:

After the data collection and preparation is done, we need to build an image classification model that will classify between these 3 categories (jeans for men, saree for women and trouser for men).

3. Data Pre-processing Done:

- We have first labelled the image data.
- We have manually removed irrelevant images that were downloading via the script.
- We have removed the duplicate images using a script.
- We have also performed multiple data augmentation techniques in order to train the model better or multiple angles and orientation of the same image.

4. Data Inputs- Logic- Output Relationships

We have given raw yet cleaned images to the machine learning model and the output produced is a label of the image on which the model is predicted.

5. State the set of assumptions (if any) related to the problem under consideration

- By observing Target Variable "label" it is already assumed that it is a Regression Problem and to understand it have to use Regression model.

6. Hardware and Software Requirements and Tools Used

- **Hardware used:**
  - **Processor**: 11th Gen Intel(R) Core(TM) i3-1125G4 @ 2.00GHz 2.00 GHz
  - **System Type**: 64-bit OS
- **Software used:**
  - **Anaconda** for 64-bit OS
  - **Jupyter** notebook

- **Tools, Libraries and Packages used:**

Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
from os import listdir
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

```
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")
```

Model/s Development and Evaluation

1. Identification of possible problem-solving approaches (methods)

- We will be first scraping the data from amazon and then cleaning the data to be further used for training the model.
- Splitting the data into train and test set for model evaluation.
- We would perform multiple data augmentation techniques on the images for better generalization of our model.
- We could check and identity an optimal batch size and number of epochs to Train the model using trial and error method also keeping the epoch loss for both training set and validation in mind.

2. **Testing of Identified Approaches (Algorithms)**

1. Linear Regression
2. Random Forest Regressor
3. KNN Regressor
4. Gradient Boosting Regressor
5. Decision Tree Regressor

3. **Run and evaluate selected models**

After collecting the data, we need do train the data. For that we created a main folder called “Clothes” in current working directory inside which we further created two folders called “train” and “test”.

In “train” folder we have kept 476 images from each clothing category and remaining 204 images we have kept in “test” folder for each category. Hence, we got 476 images for training and 204 for testing.

```
# 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()
```

```
file1 = listdir(r'Clothes/train')
file2 = listdir(r'Clothes/test')
file1, file2
```

```
(['Jeans_Images', 'Sarees_Images', 'Trousers_Images'],
 ['Jeans_Images', 'Sarees_Images', 'Trousers_Images'])
```

```
# Count of images in each folder
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 ->  476
No.of Images of Jeans in train dataset ->  476
No.of Images of Trousers in train dataset -> 476
Count of Test Images
No.of Images of Sarees in test dataset->  204
No.of Images of Jeans in test dataset -> 204
No.of Images of Trousers in test dataset->  204
```

```
#Defining dimensions of images and other parameters
input_shape=(128,128,3)
img_width=128
img_height=128
batch_size=12
epoch=100
train_samples=476
test_samples=204
```

```
# 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')

# Test 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 1428 images belonging to 3 classes.
Found 612 images belonging to 3 classes.
```

```
# 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 (MaxPooling  (None, 30, 30, 32)       0
 2D)
```

```
# 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)
```

```
# 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 1/100
39/39 [==============================] - ETA: 0s - loss: 0.9239 - accuracy: 0.5470
Epoch 1: val_accuracy improved from -inf to 0.74510, saving model to best.h5
39/39 [==============================] - 10s 216ms/step - loss: 0.9239 - accuracy: 0.5470 - val_loss: 0.6556 - val_accuracy: 0.
7451
Epoch 2/100
39/39 [==============================] - ETA: 0s - loss: 0.6750 - accuracy: 0.6538
Epoch 2: val_accuracy did not improve from 0.74510
39/39 [==============================] - 8s 193ms/step - loss: 0.6750 - accuracy: 0.6538 - val_loss: 0.5532 - val_accuracy: 0.7
206
```

Saving The Predictive Model

```
#Saving the best model
model.save('best_model.h5')
```

```
table = pd.DataFrame(model.history.history)
table
```

	loss	accuracy	val_loss	val_accuracy
0	0.923883	0.547009	0.655589	0.745098
1	0.675023	0.653846	0.553201	0.720588
2	0.578256	0.702991	0.612633	0.700980
3	0.517959	0.696581	0.732663	0.627451
4	0.474891	0.775641	0.458003	0.818627
...	...	...	...	...
95	0.145168	0.948718	0.102825	0.960784
96	0.126360	0.957265	0.072620	0.985294
97	0.162492	0.923077	0.093065	0.990196
98	0.128126	0.933761	0.121797	0.955882
99	0.129991	0.957265	0.089098	0.955882

100 rows × 4 columns

Prediction

```
#Loading the saved model
saved_model = load_model('best_model.h5')

#creating instances where elements from test directory will be called
test_jeans=r'Clothes\test\Jeans_Images'
test_saree=r'Clothes\test\Sarees_Images'
test_trouser=r'Clothes\test\Trousers_Images'
```

```
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")
```

4. Key Metrics for success in solving problem under consideration

- When it comes to the evaluation of a data science model's performance, sometimes accuracy may not be the best indicator.
- Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm's performance.
- In the Rating Prediction problem that we are trying to solve, the data is balanced. So, accuracy score nearly tells the right predictions. So, the problem of overfitting in this problem is nearly not to occur. So here, we are using an accuracy score to find a better model.

5. Visualization

6. Interpretation of the Results

Prediction

```
#Loading the saved model
saved_model = load_model('best_model.h5')

#creating instances where elements from test directory will be called
test_jeans=r'Clothes\test\Jeans_Images'
test_saree=r'Clothes\test\Sarees_Images'
test_trouser=r'Clothes\test\Trousers_Images'
```

```
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: img477.jpeg
```

```
1/1 [==============================] - 0s 38ms/step
Predicted Label is: [0]
```

CONCLUSION

1. Key Findings and Conclusions of the Study

In conclusion, this project is about image classification by using deep learning via framework Tensor-Flow. The roles of epochs in CNN were able to control accuracy and also prevent any problems such as overfitting. Thus, we were able to classify the three categories of images with an accuracy of 86.4%.

2. Learning Outcomes of the Study in respect of Data Science

In this project we were able to learn about Deep Neural Networks and Convolution Neural Network. We also learned techniques to scrap and download images in the specified directory using code.

3. Limitations of this work and Scope for Future Work

While we couldn't reach out goal of 100% accuracy in image classification, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here.

This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others.

Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.