IMAGE SCRAPING AND CLASSIFICATION PROJECT

Submitted by:

Siddharth Mukherjee

ACKNOWLEDGMENT

I have taken efforts in this project however it would not have been completed without guidance from other people. I warmly acknowledge the invaluable supervision and an inspired guidance by our SME Swati Mahaseth, FlipRobo Technology.

I would also like to express my sincere thanks to Data trained Education and FlipRobo Technology for giving me an opportunity to work on this project.

I also want to express my gratitude towards my friends and family who have patiently extended all sorts of help for accomplishing this.

I am grateful to one and all who are directly or indirectly involved in successful completion of this project.

INTRODUCTION

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.

Problem Statement:

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.

ANALYTICAL FRAMING

The project begins with data collecting phase. We have scraped the clothing images from “Amazon.com”. The 3 categories are whose data is scraped is “Sarees”, “Trousers” and “Jeans”. First we have scraped link to those images using Selenium framework and then created directories to download and save scraped images. The categories are defined as Jeans: 0, Sarees: 1, Trousers: 2.

We collected the data from difference e-commerce website, www.Amazon.in. The data is scrapped using Web scraping technique and the framework used is Selenium. We have scraped 431 images for each category i.e. total of 1239 images is collected. Then I have created a directory to download images.

```
# Creating Directories
import os
import shutil
import requests

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')
```

```
# Dowloading images
for index, link in enumerate(sarees):
    print('Downloading {0} of 431 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)
```

```
for index, link in enumerate(trousers):
    print('Downloading {0} of 431 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)
```

```
for index, link in enumerate(jeans):
    print('Downloading {0} of 431 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)
```

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” and we have further divided data into train and test data where train data consists of 993 images and test data consists of 300 images.

Displaying Images

We have used matplotlib.image library to display images downloaded.

```
# 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()
```

C

We will now check for count of images in each folder.

```
# 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 ->  331
No.of Images of Jeans in train dataset ->  331
No.of Images of Trousers in train dataset -> 331
Count of Test Images
No.of Images of Sarees in test dataset->  100
No.of Images of Jeans in test dataset -> 100
No.of Images of Trousers in test dataset->  100
```

Model Building

First we will be defining dimensions of images and other parameters too. Then for data augmentation we defined training and testing set.

```
#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=331
test_samples=100
```

```
# 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 993 images belonging to 3 classes.
Found 300 images belonging to 3 classes.
```

Creating Model using Convolution Neural Network

```
# 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)

 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 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 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 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 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
_________________________________________________________________
None
```

We have next defined early stop criteria and model check point further saving the model as “best_model.h5”.

```
# 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])
```

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

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

	loss	accuracy	val_loss	val_accuracy
0	1.087826	0.444444	1.054933	0.520833
1	0.833277	0.604938	0.760027	0.635417
2	0.632747	0.654321	0.672222	0.708333
3	0.581894	0.706790	0.543458	0.666667
4	0.628642	0.654321	0.640937	0.677083
...	...	...	...	...
95	0.321039	0.870370	0.396202	0.833333
96	0.321461	0.854938	0.363202	0.875000
97	0.338626	0.845679	0.305929	0.895833
98	0.346047	0.827160	0.381366	0.875000
99	0.311635	0.869159	0.327870	0.875000

100 rows × 4 columns

Next we have loaded our saved model and predicted results for test images. We have printed the results for all the images in test folder.

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")
```

The result is printed as shown below:

Input Image is: img12.jpeg

Predicted Label is: [0]

Input Image is: img19.jpeg

Predicted Label is: [0]

Input Image is: img14.jpeg

Predicted Label is: [1]

Input Image is: img19.jpeg

Predicted Label is: [1]

Input Image is: img14.jpeg

Predicted Label is: [2]

Input Image is: img18.jpeg

Predicted Label is: [2]

CONCLUSIONS

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In conclusion, this project is about image classification by using deep learning via framework TensorFlow. The roles of epochs in CNN was 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%.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

In this project I was able to learn about Deep Neural Networks and Convulation Neural Network. I also learned techniques to scrap and download images in the specified directory using code.

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 vesatility to the project.