Project report on

Image Scraping and Classification Project

Submitted By

Mr. Abhinandan Kadam

ACKNOWLEDGMENT

It is my sensual gratification to present this report on IMAGE SCRAPING AND CLASSIFICATION PROJECT (Deep Learning Project). Working on this project was a good experience that has given me a very informative knowledge.

I would like to express my sincere thanks to MS. SAPNA VERMA for a regular follow up and valuable guidance provided throughout.

And I am also thankful to FlipRobo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

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.

Conceptual background of domain problem

Image Classification is a fundamental task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label. Typically, Image Classification refers to images in which only one object appears and is analyzed. In contrast, object detection involves both classification and localization tasks, and is used to analyze more realistic cases in which multiple objects may exist in an image.

Image classification is the primary domain, in which deep neural networks play the most important role of medical image analysis. The image classification accepts the given input images and produces output classification for identifying the labels for those images.

Data Collection:

As per the problem statement I have scraped images of saris, jeans and trousers from amazon website. To fetch more number of images I have used some of the filters so that we can get various kinds of images as well. For this project I have collected nearly equal amount of images from each category so that our model can be trained to recognize each class with good accuracy. Around 7331 images (sarees: 2406, Jeans: 2556 and Trousers: 2370).And also I have shared the image scraping script in GitHub repository.

Analytical Problem Framing

- **Mathematical/Analytical modeling of the problem**

In this project our task is to build a Deep Learning model to recognize a particular image. Here we need to identify images among three categories (Saree, Jeans, and Trouser). So we need to train a deep learning model with several images of these three types. As we have already scraped images from amazon.in so we will use these images to train our model.

```
In [2]: #loading data into a file directory
        from random import shuffle
        from glob import glob
        image_file = glob(r"C:\Users\Aniket\Desktop\amazon images\*.jpg")
        shuffle(image_file)
```

```
In [3]: #lets check the first 10 image files
        image_file[0:10]
```

```
Out[3]: ['C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_178.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Jeans_185.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Trouser_1390.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_2022.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Jeans_1175.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_2275.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_1337.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_1209.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Saree_700.jpg',
         'C:\\Users\\Aniket\\Desktop\\amazon images\\Trouser_1253.jpg']
```

I have loaded the folder of images into a file named as image_file, after that I have shuffled the file to feed our model.

Data Processing:

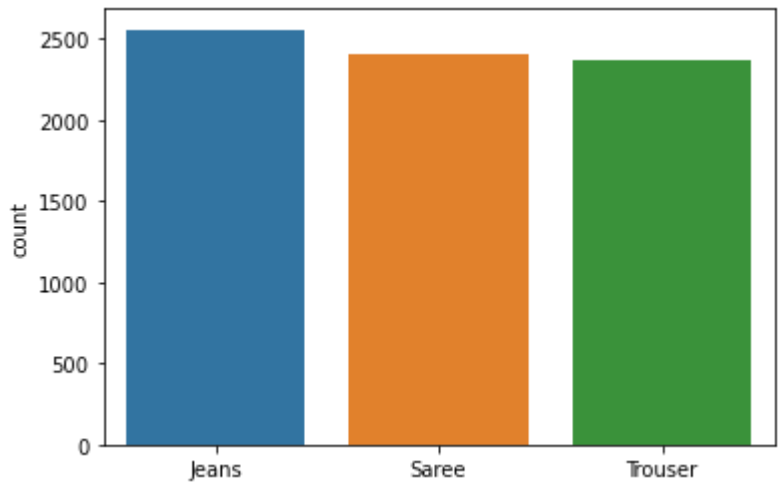
After loading the data first we need to get the labels for each image, as here we don't have labels readily available with us I will get the label for each image from corresponding file name using split function.

In [5]:
```
images = []
img_label = []
shape = (200,200)

for filename in image_file:
    img = cv2.imread(filename)

    # Resize all images to a specific shape
    if img is None:
        print('Wrong file:', filename)
    else:
        img = cv2.resize(img,shape)
        images.append(img)
        # Spliting file names and storing the labels for image in list
        img_label.append(filename.split("images")[1].split('_')[0][1:])
```

```
Wrong file: C:\Users\Aniket\Desktop\amazon images\Jeans_2555.jpg
```

In above code I have created empty lists for images and labels. I am resizing every image to shape (200,200) and getting labels from the file name using split function.

After getting image labels we will plot the count-plot for our labels to check the imbalance problem.



Looking at the above count-plot for labels we can say that we are having nearly same range of each category. Jeans are slightly more in numbers.

Then I will convert image labels into a sparse matrix using One Hot Encoding and also I am appending images as array into a list.

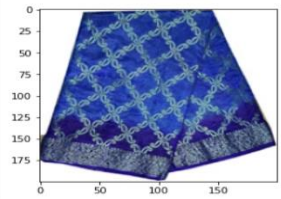
```
In [6]: # Converting labels into One Hot encoded sparse matrix
        img_label = pd.get_dummies(img_label).values

        # Converting train_images to array
        images = np.array(images)

In [7]: #lets check the amount of data we have
        len(images), len(img_label)

Out[7]: (7331, 7331)
```

Great we are having the data with 7331 images and their labels.

First image and its label:

```
In [12]: # Visualising Training data
         print(img_label[0])
         plt.imshow(images[0])
         plt.show()
```

```
[0 1 0]
```



Ten random images with labels:

Looking at above two figures we can see the images and their respective labels as a sparse matrix.

Jeans = [1 0 0], Saree = [0 1 0], Trouser = [0 0 0]

Model Building and Evaluation:

After doing all the required data processing and visualizing steps I have built 5 different Deep Learning models (Sequential) with different convolution layers and fully connected layers, activation functions and optimizers.

For this project I have used 4 convolution layers with filter size (3, 3), Max Pooling. And these models have been trained with 3 epochs with batch size 50.

Among these four models I have selected third model which is giving me a good accuracy, f1_score, precision and recall for as our final model *(Because of Ram/memory problem I have not applied Hyper tuning separately).*

Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

- As this is classification problem I am using accuracy score here.
- I have also checked for f1_score, precision, recall to ensure the better performance of our model. For this purpose I have built three functions to evaluate with these metrics.

In [14]:
```
#defining functions to calculate recall, precision and f1_scores
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Final Model:

In [20]:
```
# Creating a Sequential model
model3= Sequential()
#first convolution layer
model3.add(Conv2D(kernel_size=(3,3), filters=32, activation='tanh',kernel_initializer='he_uniform',
                  input_shape=(200,200,3)))
#second convolution layer
model3.add(Conv2D(filters=30,kernel_size = (3,3),activation='tanh'))
model3.add(MaxPool2D(2,2))
#third convolution layer
model3.add(Conv2D(filters=30,kernel_size = (3,3),activation='tanh'))
model3.add(MaxPool2D(2,2))
#fourth convolution layer
model3.add(Conv2D(filters=30,kernel_size = (3,3),activation='tanh'))

model3.add(Flatten())

#adding fully connected layers
model3.add(Dense(100,activation='relu',kernel_initializer='he_uniform'))
model3.add(Dense(3,activation = 'softmax'))

model3.compile(
              loss='categorical_crossentropy',
              metrics=['accuracy', f1_score, precision_m, recall_m],
              optimizer='adam'
             )
```

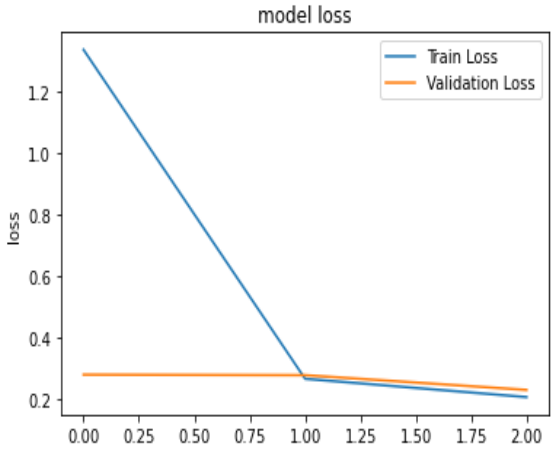
In [21]:
```
model3.summary()
```

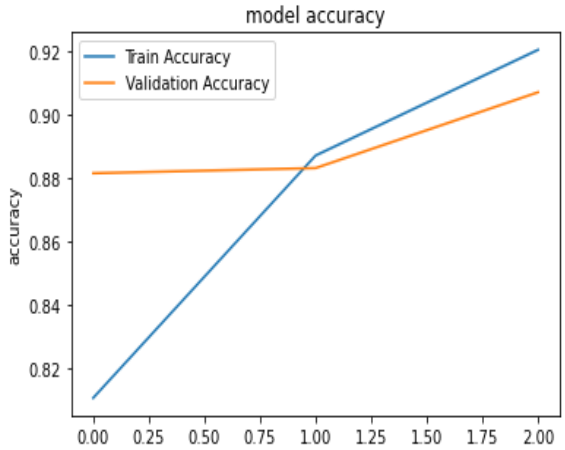
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 198, 198, 32)      896

 conv2d_9 (Conv2D)           (None, 196, 196, 30)      8670

 max_pooling2d_4 (MaxPooling  (None, 98, 98, 30)       0
 2D)

 conv2d_10 (Conv2D)          (None, 96, 96, 30)        8130

 max_pooling2d_5 (MaxPooling  (None, 48, 48, 30)       0
 2D)

 conv2d_11 (Conv2D)          (None, 46, 46, 30)        8130

 flatten_2 (Flatten)         (None, 63480)             0

 dense_4 (Dense)             (None, 100)               6348100

 dense_5 (Dense)             (None, 3)                 303

=================================================================
Total params: 6,374,229
Trainable params: 6,374,229
Non-trainable params: 0
_________________________________________________________________
```

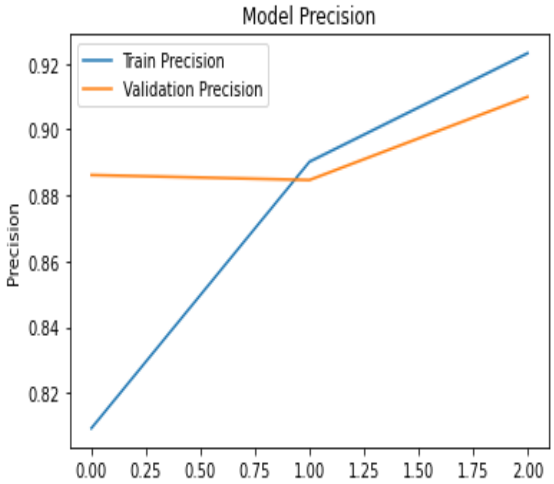
In [34]:
```
# Evaluating model on validation data
evaluate = model3.evaluate(x_test,y_test)
print(evaluate)
```

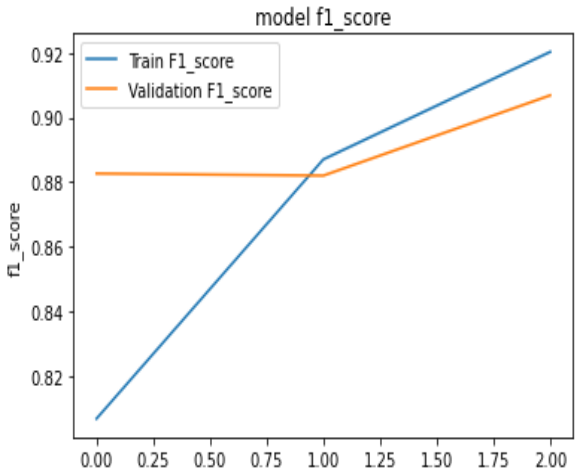
```
58/58 [==============================] - 42s 730ms/step - loss: 0.2285 - accuracy: 0.9073 - f1_score: 0.9074 - precis
ion_m: 0.9112 - recall_m: 0.9038
[0.22847656905651093, 0.9072558879852295, 0.9073639512062073, 0.911165177822113, 0.9037954807281494]
```

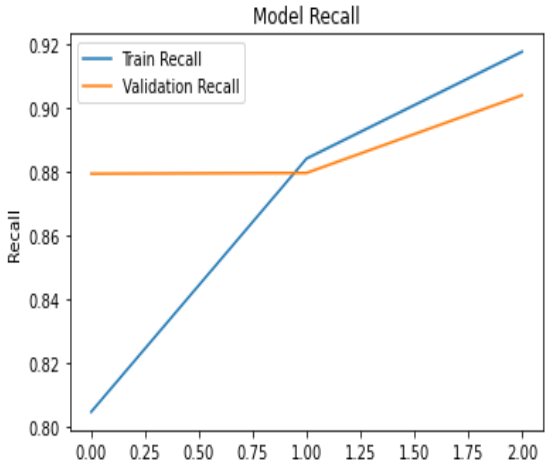
Model Summary: Here I have plotted graphs for model history with different evaluation metrics.











Testing actual vs predictions:

In [35]:
```
# Testing predictions and the actual label
checkImage = x_test[0:1]
checklabel = y_test[0:1]

predict = model.predict(np.array(checkImage))

output = { 0:'Jeans',1:'Saree',2:'Trouser'}

print("Actual :- ",checklabel)
print("Predicted :- ",output[np.argmax(predict)])
```

```
Actual :-  [[1 0 0]]
Predicted :-  Jeans
```

In [36]:
```
# Testing predictions and the actual label
checkImage = x_test[50:51]
checklabel = y_test[50:51]

predict = model.predict(np.array(checkImage))

output = { 0:'Jeans',1:'Saree',2:'Trouser'}

print("Actual :- ",checklabel)
print("Predicted :- ",output[np.argmax(predict)])
```

```
Actual :-  [[0 1 0]]
Predicted :-  Saree
```

Great, we can see our model is predicting the images accurately.

Conclusions:

For this image classification project I have scraped images from amazon.in, after doing required data processing and visualizing the images I have built several models among them one is selected with higher performances.

We can still improve our model performance by feeding large numbers of images to our model and by doing extensive hypertuning for different parameters.