IMAGE SCRAPPING AND CLASSIFICATION PROJECT

Submitted by:

Vaishali Patil-Chavhan

ACKNOWLEDGMENT

I wish to express my sincere gratitude to **DataTrained** Academy and **FlipRobo Technologies** who gave me the opportunity to do the **IMAGE SCRAPPING AND CLASSIFICATION PROJECT**. It helped me to do a lot of research and I have grasped many new things.

I have put good effort in this project. However it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. I would also thankful to the blogs/articles through online platforms which gave me a lot of information in finishing this project within the limited time.

INTRODUCTION

Business Problem Framing:

Data scrapping used for collecting data of particular site for insufficient data and for image classification CNN method is used to teach a machine about that particular image. More than 25% of the whole revenue in E-Commerce is attributed to apparel & accessories. a serious problem they face is categorizing these apparels from just the pictures especially when the categories provided by the brands are inconsistent. This poses a stimulating computer vision problem that has caught the eyes of several deep learning researchers.

Conceptual Background of the Domain Problem:

Data science plays an important role to solve business problems by which companies increase their profits and improve business strategies. Our objective is to create a Classification model that classifies the image of each clothing category (which is scraped from an E-Commerce website) focusing on changing trends.

Motivation for the Problem Undertaken:

Every problem begins with ideas that are further developed and inspired to address a variety of situations and circumstances. Learning the theoretical background for data science or machine learning are often a frightening experience, because it involves multiple fields of mathematics and an extended list of online resources. By proper practical research and practice I can become better in this field. These suggestions are derived from my mentors/SME's and my own experience in the beginner projects.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem:

- Data is collected through web scraping of an E-Commerce website.
- Data is scraped using python libraries such as selenium and web driver (chrome driver).

Data Sources and their formats:

- The data scraped is in the form of images which is in .jpg format.

- Data is collected through web scraping of an E-Commerce website (Amazon). Three category of clothes are scrapped namely Sarees (women), Trousers (men) and Jeans (men).

Data Pre-processing Done:

The obtained data is split into train and test (80:20) and as input and output respectively using split folders.

```
! pip install split_folders
```

```
Collecting split_folders
  Downloading split_folders-0.4.3-py3-none-any.whl (7.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.4.3
```

```
#splitting the data
import splitfolders

input = "/content/drive/MyDrive/Category"
Output = "/content/drive/MyDrive" #where you want the split datasets saved. one will be created if none is set

splitfolders.ratio(input, output="Output", seed=42, ratio=(0.8,0.2))
```

```
Copying files: 718 files [02:26,  4.92 files/s]
```

Hardware and Software Requirements and Tools Used:

Processor Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed RAM 4.00 GB
System type 64-bit operating system, x64-based processor
Edition Windows 10 Pro
Software: The complete project is done using Jupyter Notebook and Google Colab.

Libraries:

Numpy, Keras, Tensorflow, Matplotlib, SKlearn, cv2.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Scrapped images are classified into three categories and are tested using the model.

Testing of Identified Approaches (Algorithms):

Inception model is used and Imagenet as weights, optimizer RMSprop, convolution 2D filters, loss as categorical_crossentropy and accuracy as metrics.

Run and Evaluate selected models:

Hyper parameter tuning is done to know the best fit parameters such as filters and layers by setting some number of trials and found the metrics of the model.

```
[16] #model cost and optimization
     model.compile(
         loss='categorical_crossentropy',
         optimizer='rmsprop',
         metrics=['accuracy']
     )
```

```
[17] train_datagen = ImageDataGenerator(rescale = 1./255,
                                        shear_range = 0.2,
                                        zoom_range = 0.2,
                                        horizontal_flip = True)

     test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
[18] # Make sure you provide the same target size as initialied for the image size
     training_set = train_datagen.flow_from_directory('/content/Output/train',
                                                      target_size = (80,80),
                                                      batch_size = 16,
                                                      class_mode = 'categorical')

     Found 574 images belonging to 3 classes.
```

```
[19] test_set = test_datagen.flow_from_directory('/content/Output/val',
                                                 target_size = (80,80),
                                                 batch_size = 16,
                                                 class_mode = 'categorical')

     Found 144 images belonging to 3 classes.
```

```
# fit the model
fitting = model.fit(
  training_set,
  validation_data=test_set,
  epochs=20,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
Epoch 1/20
36/36 [==============================] - 13s 254ms/step - loss: 0.5773 - accuracy: 0.8235 - val_loss: 0.4030 - val_accuracy: 0.8889
Epoch 2/20
36/36 [==============================] - 7s 206ms/step - loss: 0.3481 - accuracy: 0.9152 - val_loss: 0.3219 - val_accuracy: 0.9144
Epoch 3/20
36/36 [==============================] - 7s 208ms/step - loss: 0.2888 - accuracy: 0.9338 - val_loss: 0.2817 - val_accuracy: 0.9213
Epoch 4/20
36/36 [==============================] - 7s 207ms/step - loss: 0.2331 - accuracy: 0.9495 - val_loss: 0.2452 - val_accuracy: 0.9259
Epoch 5/20
36/36 [==============================] - 7s 206ms/step - loss: 0.2391 - accuracy: 0.9373 - val_loss: 0.2304 - val_accuracy: 0.9306
Epoch 6/20
36/36 [==============================] - 7s 206ms/step - loss: 0.2022 - accuracy: 0.9547 - val_loss: 0.2367 - val_accuracy: 0.9213
Epoch 7/20
36/36 [==============================] - 7s 206ms/step - loss: 0.1854 - accuracy: 0.9535 - val_loss: 0.2124 - val_accuracy: 0.9213
Epoch 8/20
36/36 [==============================] - 7s 205ms/step - loss: 0.1679 - accuracy: 0.9570 - val_loss: 0.2742 - val_accuracy: 0.9213
Epoch 9/20
36/36 [==============================] - 8s 208ms/step - loss: 0.1547 - accuracy: 0.9588 - val_loss: 0.2105 - val_accuracy: 0.9213
Epoch 10/20
36/36 [==============================] - 8s 212ms/step - loss: 0.1791 - accuracy: 0.9576 - val_loss: 0.1803 - val_accuracy: 0.9444
Epoch 11/20
36/36 [==============================] - 8s 213ms/step - loss: 0.1442 - accuracy: 0.9733 - val_loss: 0.1841 - val_accuracy: 0.9444
Epoch 12/20
36/36 [==============================] - 8s 210ms/step - loss: 0.1609 - accuracy: 0.9611 - val_loss: 0.1884 - val_accuracy: 0.9259
Epoch 13/20
36/36 [==============================] - 8s 213ms/step - loss: 0.1354 - accuracy: 0.9652 - val_loss: 0.2158 - val_accuracy: 0.9306
Epoch 14/20
36/36 [==============================] - 8s 213ms/step - loss: 0.1510 - accuracy: 0.9640 - val_loss: 0.1363 - val_accuracy: 0.9583
Epoch 15/20
36/36 [==============================] - 8s 213ms/step - loss: 0.1186 - accuracy: 0.9791 - val_loss: 0.1578 - val_accuracy: 0.9259
Epoch 16/20
36/36 [==============================] - 8s 212ms/step - loss: 0.1302 - accuracy: 0.9663 - val_loss: 0.1519 - val_accuracy: 0.9306
Epoch 17/20
36/36 [==============================] - 8s 212ms/step - loss: 0.1585 - accuracy: 0.9617 - val_loss: 0.1364 - val_accuracy: 0.9630
Epoch 18/20
36/36 [==============================] - 8s 210ms/step - loss: 0.1314 - accuracy: 0.9768 - val_loss: 0.1677 - val_accuracy: 0.9259
Epoch 19/20
36/36 [==============================] - 8s 210ms/step - loss: 0.1443 - accuracy: 0.9628 - val_loss: 0.1367 - val_accuracy: 0.9630
Epoch 20/20
36/36 [==============================] - 8s 211ms/step - loss: 0.1462 - accuracy: 0.9605 - val_loss: 0.1353 - val_accuracy: 0.9537
```

Hyper parameter tuning of the models:

```
# We got good accuracy but let's do hyper parameter tuning
def build_model(hyperparameter):
  model = keras.Sequential([
    keras.layers.Conv2D(
        filters=hyperparameter.Int('conv_1_filter', min_value=32, max_value=128, step=16),
        kernel_size=hyperparameter.Choice('conv_1_kernel', values = [3,5]),
        activation='relu',
        input_shape=(80,80,3)
    ),
    keras.layers.Conv2D(
        filters=hyperparameter.Int('conv_2_filter', min_value=32, max_value=128, step=16),
        kernel_size=hyperparameter.Choice('conv_2_kernel', values = [3,5]),
        activation='relu'
    ),
    keras.layers.Flatten(),
    keras.layers.Dense(
        units=hyperparameter.Int('dense_1_units', min_value=32, max_value=128, step=16),
        activation='relu'
    ),
    keras.layers.Dense(3, activation='softmax')
  ])

  model.compile(optimizer=keras.optimizers.RMSprop(hyperparameter.Choice('learning_rate', values=[1e-2, 1e-3])),
              loss='categorical_crossentropy',
              metrics=['accuracy']
  )

  return model
```

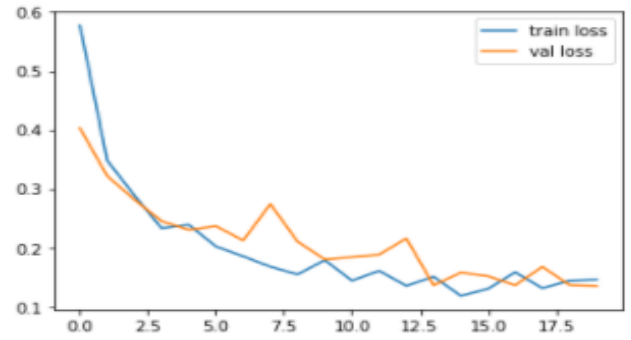
Key Metrics for success in solving problem under consideration:

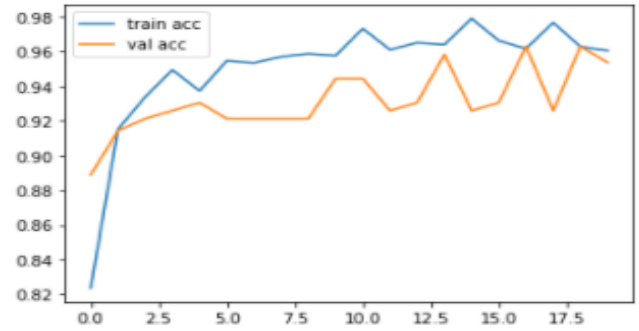
The metrics used for the model are training and validation accuracy.

Visualizations:

```
[23] #plot the loss
     plt.plot(fitting.history['loss'],label='train loss')
     plt.plot(fitting.history['val_loss'],label='val loss')
     plt.legend()
     plt.show()
     plt.savefig('loss_val')

     #plot the accuracy
     plt.plot(fitting.history['accuracy'],label='train acc')
     plt.plot(fitting.history['val_accuracy'],label='val acc')
     plt.legend()
     plt.show()
     plt.savefig('accuracyval')
```






```
<Figure size 432x288 with 0 Axes>
```

From the above plots, it is clear that Accuracy of both training and validation accuracy increasing and both the loss are decreasing respectively.

Interpretation of the Results:

- Given imagenet as weights for the inception model in layers.
- Hyper parameter tuning is done for best fit parameters and good performance metrics.

Testing Predictions:

```
[100] # test the model
      from google.colab import files
      uploaded = files.upload()
```

Choose Files img167.jpg

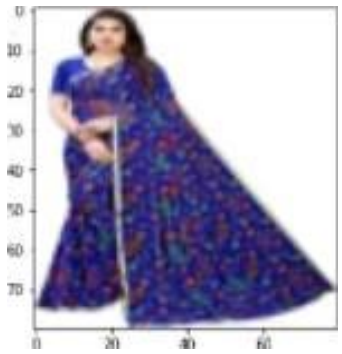
- **img167.jpg**(image/jpeg) - 20182 bytes, last modified: 7/16/2021 - 100% done

Saving img167.jpg to img167.jpg

```
# image file
new_image = plt.imread('img167.jpg')
img = plt.imshow(new_image)
```

```
[102] #resize image
      from skimage.transform import resize

      img = plt.imshow(resized_image)
```



```
      #model predictions
      predictions = model.predict(np.array([resized_image]))

      predictions
```

```
array([[3.6682714e-23, 1.0000000e+00, 8.2262754e-24]], dtype=float32)
```

```
      list_index = [0,1,2]


      for i in range(3):
        for j in range(3):
          if x[0][list_index[i]] < x[0][list_index[j]]:
            temp = list_index[i]
            list_index[i] = list_index[j]
            list_index[j] = temp

      print(list_index)
```

```
[Yes  training_set.class_indices
```

```
[106] classification = ['ins','San  '.' TiW  s a  ']
```

```
[107] #predicted image name is
      classification[list_index[i]]
```

CONCLUSION

Key Findings and Conclusions of the Study:

Predicting the image uploaded to Google colab and predicted with the model and got the expected result.

Learning Outcomes of the Study in respect of Data Science:

- By using selenium scrapping images from amazon made easy.
- By using optimizers, layers and filters observed that there is difference in scoring.
- RMSprop optimizer gave best score.