IMAGE SCRAPING AND CLASSIFICATION PROJECT

Submitted by:

AJITESH KUMAR

ACKNOWLEDGMENT

I would like to thank FlipRobo Technologies for giving me the opportunity to work on this project. I am very grateful to DataTrained team for providing me the knowledge which helped me a lot to work on this project. Reference sources are:

1. Google

2. YouTube

3. Stackoverflow

4. Analyticsvidhya

INTRODUCTION

- ## Business Problem Framing

  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.

- ## Conceptual Background of the Domain Problem

  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.

- ## Motivation for the Problem Undertaken

  The project was given by FlipRobo Technologies as part our internship program. This project helped us to work on the Deep Learning and computer vision. This also helped to understand the concept in better manner.

Analytical Problem Framing

- Data Sources and their formats

  Our dataset contains total 719 images which are scraped from the e-commerce website amazon.in All the images are in JPEG format.

  There are total 3 classes in our dataset:

  a. Sarees (Women)

  b. Jeans (Men)

  c. Trousers (Men)

```
# re-size all the images to this
img_size = [224, 224]

train_path = 'E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/train'
valid_path = 'E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/test'
```

```
# useful for getting number of output classes
folders = glob('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/train/*')
```

```
# checking for the image folders
folders
```

```
['E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/train\\Jeans(Men)',
 'E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/train\\Saree(Women)',
 'E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/train\\Trouser(Men)']
```

```
len(folders)
```

```
3
```

- Hardware and Software Requirements and Tools Used

  We have used computer having i3-4th Gen processor with 4GB RAM and 64-bit windows 10 operating system.

```
import tensorflow as tf
print(tf.__version__)
```

```
2.6.0
```

```
import pandas as pd
import seaborn as sns
import os
from skimage.io import imread
from skimage.transform import resize
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  We used following algorithms to train and test our final model.

  I. VGG16

  II. ResNet50

  III. Inception

- Run and Evaluate selected models

1. VGG16

```
# Import the VGG16 library as shown below and add preprocessing layer to the front of VGG
# Here we will be using imagenet weights

vgg16 = VGG16(input_shape=img_size + [3], weights='imagenet', include_top=False)
```

```
# don't train existing weights
for layer in vgg16.layers:
    layer.trainable = False
```

```
# our layers
x = Flatten()(vgg16.output)
```

```
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=vgg16.input, outputs=prediction)
```

```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy'])
```

```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

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

```
# Provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images dat
                                                 target_size = (224, 224),
                                                 batch_size = 20,
                                                 class_mode = 'categorical')
```

```
Found 620 images belonging to 3 classes.
```

```
test_set = test_datagen.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data/tes
                                            target_size = (224, 224),
                                            batch_size = 20,
                                            class_mode = 'categorical')
```

```
Found 99 images belonging to 3 classes.
```

```
# fit the model
# Run the cell. It will take some time to execute
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=20,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
C:\Users\Ajitesh\anaconda3\lib\site-packages\keras\engine\training.py:1972: UserWarning: `Model.fit_generator` is deprecated an
d will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/20
31/31 [==============================] - 297s 9s/step - loss: 0.3886 - accuracy: 0.8290 - val_loss: 0.1264 - val_accuracy: 0.93
94
Epoch 2/20
31/31 [==============================] - 415s 14s/step - loss: 0.1092 - accuracy: 0.9613 - val_loss: 0.0658 - val_accuracy: 0.9
798
Epoch 3/20
31/31 [==============================] - 503s 16s/step - loss: 0.0838 - accuracy: 0.9677 - val_loss: 0.0273 - val_accuracy: 1.0
000
Epoch 4/20
31/31 [==============================] - 422s 14s/step - loss: 0.0347 - accuracy: 0.9952 - val_loss: 0.0143 - val_accuracy: 1.0
000
Epoch 5/20
31/31 [==============================] - 576s 19s/step - loss: 0.0267 - accuracy: 0.9968 - val_loss: 0.0127 - val_accuracy: 1.0
000
```

2. ResNet50: ¶

```
# Adding data-augmentation parameters to ImageDataGenerator

train_res = ImageDataGenerator(rescale = 1./255., rotation_range = 40, width_shift_range = 0.2,
                               height_shift_range = 0.2, shear_range = 0.2, zoom_range = 0.2,
                               horizontal_flip = True)

test_res = ImageDataGenerator(rescale = 1.0/255.)

train_generator = train_res.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images data
                                                batch_size = 20, class_mode = 'categorical', target_size = (224, 224))

validation_generator = test_res.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images
                                                    batch_size = 20, class_mode = 'categorical', target_size = (224, 224))
```

```
Found 620 images belonging to 3 classes.
Found 99 images belonging to 3 classes.
```

```
# Importing the base model
from tensorflow.keras.applications import ResNet50
base_model = ResNet50(input_shape=(224, 224,3), include_top=False, weights="imagenet")
```

```
#We're using the basic ResNet model, so we will keep the layers frozen and only modify the last layer
for layer in base_model.layers:
    layer.trainable = False
```

```
#Build and Compile the Model

from tensorflow.keras.applications import ResNet50
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Flatten, GlobalAveragePooling2D

base_model = Sequential()
base_model.add(ResNet50(include_top=False, weights='imagenet', pooling='max'))
base_model.add(Dense(3, activation='sigmoid'))
```

```
# tell the model what cost and optimization method to use

base_model.compile(optimizer = 'SGD', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
#Fitting the model
resnet_history = base_model.fit_generator(
  train_generator,
  validation_data=validation_generator,
  epochs=10,
  steps_per_epoch=len(train_generator),
  validation_steps=len(validation_generator)
)
```

```
C:\Users\Ajitesh\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1969: UserWarning: `Model.fit_generator
` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/10
31/31 [==============================] - 559s 17s/step - loss: 0.3902 - accuracy: 0.8581 - val_loss: 1.5974 - val_accuracy: 0.3
333
Epoch 2/10
31/31 [==============================] - 529s 17s/step - loss: 1.7576 - accuracy: 0.7645 - val_loss: 4.5000 - val_accuracy: 0.3
636
Epoch 3/10
31/31 [==============================] - 544s 18s/step - loss: 2.5507 - accuracy: 0.7226 - val_loss: 4.3851 - val_accuracy: 0.4
141
Epoch 4/10
31/31 [==============================] - 517s 17s/step - loss: 1.0502 - accuracy: 0.8032 - val_loss: 1.9233 - val_accuracy: 0.4
242
Epoch 5/10
31/31 [==============================] - 526s 17s/step - loss: 0.7828 - accuracy: 0.8306 - val_loss: 2.5447 - val_accuracy: 0.4
141
```

3. Inception:

```
# Add our data-augmentation parameters to ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255., rotation_range = 40, width_shift_range = 0.2,
                                   height_shift_range = 0.2, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1.0/255.)
```

```
# Creating train and test set

train_inception = train_datagen.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images
                                                    batch_size = 20, class_mode = 'categorical', target_size = (150, 150))

test_inception = test_datagen.flow_from_directory('E:/Flip Robo/Project - 8 - Image Scraping and Classification Project/Images da
                                                  batch_size = 20, class_mode = 'categorical', target_size = (150, 150))
```

```
Found 620 images belonging to 3 classes.
Found 99 images belonging to 3 classes.
```

```
#loading the base model
from tensorflow.keras.applications.inception_v3 import InceptionV3
inception_model = InceptionV3(input_shape = (150, 150, 3), include_top = False, weights = 'imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 34s 0us/step
87924736/87910968 [==============================] - 34s 0us/step
```

```
#freeze the layers:
for layer in inception_model.layers:
    layer.trainable = False
```

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras import layers

x = layers.Flatten()(inception_model.output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)

# Add a final sigmoid layer with 3 node for classification output
x = layers.Dense(3, activation='sigmoid')(x)

model = tf.keras.models.Model(inception_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001), loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
#fitting the model
inc_history = model.fit_generator(train_inception, validation_data = test_inception,
                                  steps_per_epoch = len(train_inception), epochs = 10)
```

```
C:\Users\Ajitesh\anaconda3\lib\site-packages\keras\engine\training.py:1972: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

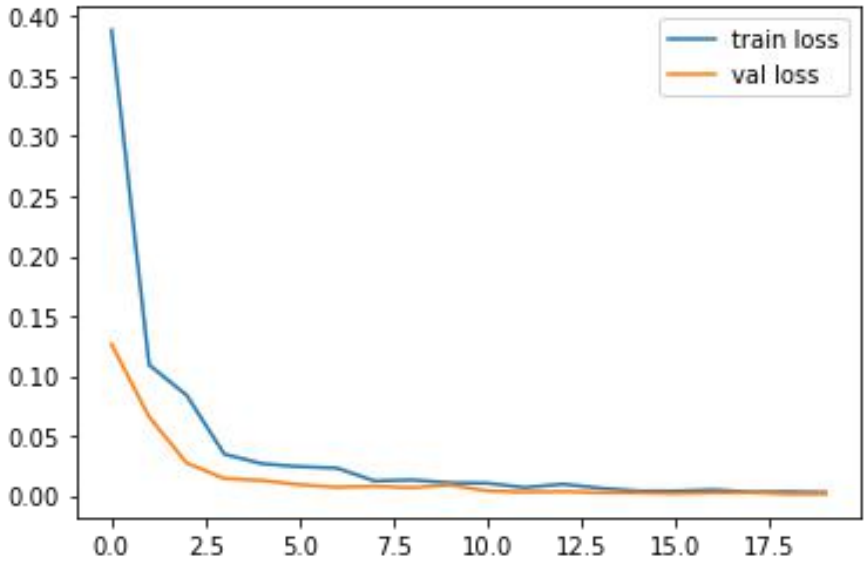
```
Epoch 1/10
31/31 [==============================] - 47s 2s/step - loss: 0.9882 - accuracy: 0.8339 - val_loss: 0.4182 - val_accuracy: 0.8687
Epoch 2/10
31/31 [==============================] - 43s 1s/step - loss: 0.7498 - accuracy: 0.8468 - val_loss: 1.2689 - val_accuracy: 0.7172
Epoch 3/10
31/31 [==============================] - 42s 1s/step - loss: 0.4139 - accuracy: 0.8887 - val_loss: 1.3001 - val_accuracy: 0.8182
Epoch 4/10
31/31 [==============================] - 38s 1s/step - loss: 0.4984 - accuracy: 0.8839 - val_loss: 0.5975 - val_accuracy: 0.8687
Epoch 5/10
31/31 [==============================] - 37s 1s/step - loss: 0.5348 - accuracy: 0.8694 - val_loss: 0.4899 - val_accuracy: 0.8990
Epoch 6/10
31/31 [==============================] - 37s 1s/step - loss: 0.4820 - accuracy: 0.8839 - val_loss: 0.5641 - val_accuracy: 0.8889
Epoch 7/10
31/31 [==============================] - 37s 1s/step - loss: 0.4102 - accuracy: 0.8887 - val_loss: 0.3534 - val_accuracy: 0.9394
Epoch 8/10
31/31 [==============================] - 45s 1s/step - loss: 0.4278 - accuracy: 0.8742 - val_loss: 0.6487 - val_accuracy: 0.8687
Epoch 9/10
31/31 [==============================] - 39s 1s/step - loss: 0.3140 - accuracy: 0.9290 - val_loss: 0.8149 - val_accuracy: 0.8586
```

- Visualizations

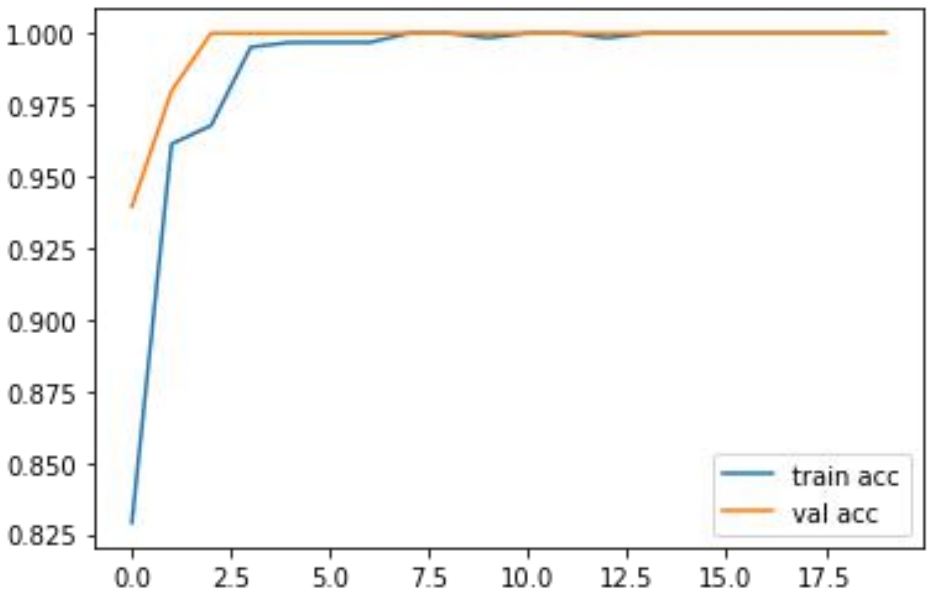
Random images from the dataset are:



The graph between the training loss and validation loss is:



The graph between the training accuracy and validation accuracy is:



- Interpretation of the Results

  VGG16 model performed best among all the other models which we used for the image classification project. It can be improved by increasing the number of images in our dataset.

CONCLUSION

- Key Findings and Conclusions of the Study

  Our final model VGG16 gives accuracy of 1.000.

- Learning Outcomes of the Study in respect of Data Science

  The project helped us to gain the knowledge of different algorithms of Deep Learning and computer vision.

- Limitations of this work and Scope for Future Work

  Since, Jeans and Trousers looks quite similar to each other and so our model was getting confused in some cases and it was not giving accurate results. This can be improved by providing more images in the dataset.