NAME: REDWAN AHMED

INSTITUTION: UNIVERSITY OF GREENWICH

COURSE: APPLIED MACHINE LEARNING

INSTUCTOR: Dr. Dimitrios Kollias

DATE: 30-04-2021

INTRODUCTION

Facial detection is the capability of a system to detect an individual based on their facial attributes. In the modern world, face detection and recognition form part of the extensive biometrics that includes voice recognition, fingerprint and palm recognition, eye scan recognition, and the traditional signature recognition.

This project aims to develop a machine learning system that detects faces and physical attributes surrounding the face. These attributes include wrinkles, freckles, whether or not a face has glasses, and if yes, what type of glasses (normal or sunglasses). It also further predicts the hair colour and size of the hair top based on the data set provided.

RELATED WORK

The face recognition systems are been taken over by biometrics, its almost impossible to design a system which has a performance as close as a human. *"New computer vision and pattern recognition approaches need to be investigated. For instance, Facebook researchers are currently developing algorithms called "DeepFace" to detect whether two faces in unfamiliar photos are of the same person with 97.25% accuracy, regardless of lighting conditions or angles. As a comparison, humans generally have an average of 97.53% accuracy. This means that Facebook's facial-processing software has nearly the same accuracy as humans"* (Chowdhry, 2014).

To achieve this equality between human recognition and artificial recognition, knowledge and the different fields requires different perspectives such as psychology along with neural science are being implemented in biometrics to make it more accurate but in case of face recognising is almost 100% accurate.

DATASET PREPARATION

A data set containing images were provided for use. This was a collection of 1999 different images. For my dataset, the images were contained in two folders, which I eventually put under one folder for annotation purposes. Together with the file images was the course work annotation tools that I used for annotating the images.

To open the course work annotation tools, I used Visual Studio Code text editor. I supplied the required paths for the video_name, images and file name respectively in the merge_images_to_video.py file.

However, I made few modifications to the code since the initial code provided in this file threw in a -215 assertion failure or error. This error means that the merge_images_to_video.py file was trying to resize an image with either height or width as zero. This error may also occur if a path to a source or destination folder is not correctly referenced. I put the resize image statement in a try-except block which continues with the merging when it encounters the cv2error. This generated an image text document with 1666 images out of a sample size of 1999. The merge_images_to_video.py file eliminated the 334 images that could not be read. The images/photos might be null or contained a wrong format that python would not recognize. This file also generates a video.mp4 file that contains all the 1666 images in the image_names.txt file. The images in the video.mp4 are duplicated each once.

I further converted the video.mp4 to video.mov and uploaded it in the index.html file to commence annotation. Since all images contained two frames, I annotated once and skipped the duplicate image. The annotation included wrinkles, freckles, hair colour, hair top and no face or not human. I annotated all the 1666 images up to frame 3332. As a result of the annotations, a AU_video.txt file is generated and contains all the annotations for the images an N/A values for the duplicated images that I skipped.

The Process_of_output.py file provided is used to remove the N/A values and the false values in the Au_video.txt file. The resulting file is a list of annotations for the 1666 images. I, however, modified the file to append another column called hair visibility in the event that the hair colour and hair top are both not visible. Below is the code I added.

```
hair_color = outputs[int(line[7])]
hair_top = outputs[int(line[9])]

print(type(hair_color))
if hair_color == 8 & hair_top == 4:
    outputs.append(0) #not visible
else:
    outputs.append(1) #visible
```

From the above code, if Hair colour corresponds to 8 and hair top corresponds to 4 simultaneously for the same image, this means that hair is not visible, and we indicate it in the visibility column for each image.

To further simplify the annotation output, I imported the xslx library to print the output in excel, as shown below

```
file_path = 'C:/Users/myPc/Desktop/python projects/anotations/image_names.xlsx'


workbook = xlsxwriter.Workbook(file_path)
worksheet = workbook.add_worksheet()

row = 0
col = 0
for  Wrinkles, Freakles, Glasses, Hair_color, Hair_top, No_face in (outputs):
    worksheet.write(row, col + 0, Wrinkles)
    worksheet.write(row, col + 1, Freakles)
    worksheet.write(row, col + 2, Glasses)
    worksheet.write(row, col + 3, Hair_color)
    worksheet.write(row, col + 4, Hair_top)
    worksheet.write(row, col + 5, No_face)
    worksheet.write(row, col + 5, Hair visibility)
    row +=1

workbook.close()
```

The above code printed the annotations in an excel spreadsheet and had 1666 rows which correspond to the number of image names. In the excel spreadsheet, I converted it into a CSV file and saved it in my working directory.

I then read the annotations.csv file in jupyter notebook and assigned it the variable name **df** Using the command included below.

```
import pandas as pd

df = pd.read_csv('annotations.csv')
```

The first step in cleaning the data was to make sure that I also read the image_names.txt file and check for total length in the two files. i.e. annotations.csv and image_names.txt, which all matched to 1666.

The image_names in the text file contained their initial full path, and the next step was to eliminate the paths by dropping the slashes(/).

I went further to drop the rows where no face was visible, using the following command.

```
#dropping rows where no face is visible

df = df[df['No face'] == 0]

df.shape
```

the result of this code returned to be (1556, 9), which means that our data set contained 1556 visible faces.

At this point I went further to import matplotlib to visualize my data in graphs.

Since the scope of this project was to predict facial attributes, I generated a graph that shows how faces, wrinkles, freckles, glasses, hair_colour and hair_top vary in my dataset using matplotlib

I further replaced the parent objects with their corresponding child objects and stored them in another data frame. For example, I replaced glasses with three child objects, i.e. does_not_wear, Wears_normal, and sunglasses.

On printing the final data frame shape, I got (1556,20), which means that 20 columns were in the data frame.

Below is the output.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
```

```
Data columns (total 29 columns :
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Image_Name      1452 non-null   object
 1   brown           1556 non-null   int64
 2   black           1556 non-null   int64
 3   gray            1556 non-null   int64
 4   blond           1556 non-null   int64
 5   red             1556 non-null   int64
 6   white           1556 non-null   int64
 7   mixed           1556 non-null   int64
 8   other           1556 non-null   int64
 9   not_visible     1556 non-null   int64
 10  wrinkles        1556 non-null   int64
 11  Freakles        1556 non-null   int64
 12  do_not_wear     1556 non-null   int64
 13  wear_normal     1556 non-null   int64
 14  wear_sunglasses  1556 non-null   int64
 15  bald            1556 non-null   int64
 16  shaved          1556 non-null   int64
 17  has_few_hair    1556 non-null   int64
 18  has_thick_hair   1556 non-null   int64
 19  other_top       1556 non-null   int64
dtypes: int64(19), object(1)
```

I further imported Tensorflow and Keras to help generate image data. This image data helps to find how many images are valid together with their file names. Our model found 107 invalid filenames, which it ignored. This is shown in the input frame (In [189]), and below is the output:

```
/usr/local/lib/python3.7/dist-packages/keras_perprocessing/image/datafr
ame_iterator.py:282: UserWarning: found 107 invalid image filename(s) i
n x_col="Image_Name". These filename(s) will be ignored.
  .format(n_invalid, x_col)
```

MACHINE LEARNING METHOD

The first step in this project was to label the data manually and creating the annotations for use in the training and testing of the data. This makes our machine learning methodology a supervised method. The model simply compares the data in the annotations.csv file with newly received data to find errors and make predictions through this method.

In the training_code.ipnb (In[193]) file, I checked the total number of trainable parameters against the total parameters as shown below

```
Total params: 23,916,851
Trainable params: 23,916,595
Non-trainable params: 256
______________________________
```

I calculated and visualized the models precision and accuracy with these trainable parameters, which is an average of 0.89.

I have then modified and tested the model to learn how to predict the facial and hair attributes of a person.

To test the model, I supplied a text file containing paths to 12 images inclusive of 2 statute images, and the predictions of the model were as follows.

```
[['picture1.jpg', 1, 1, 1, 6, 3],
 ['Test_images.txt', 1, 1, 1, 6, 3],
 ['normalglasses.jpg', 1, 1, 0, 6, 3],
 ['freakles_img_2.jpg', 1, 1, 0, 6, 3],
 ['statute2.jpg', 1, 1, 1, 6, 3],
 ['picture4.jpg', 1, 1, 1, 6, 3],
 ['picture5.jpg', 1, 1, 1, 6, 3],
 ['picture2.jpg', 1, 1, 1, 6, 3],
 ['normalglasses2.jpg', 1, 1, 1, 6, 3],
 ['freakles_img_1.jpg', 1, 1, 1, 6, 3],
 ['sunglasses.jpg', 1, 1, 1, 6, 3],
 ['statute.jpg', 1, 1, 0, 6, 3]]
```

Although the model performed the prediction, it failed to recognize that the two statute images did not have a face. This is due to the fact that we dropped the no face attribute during data cleaning.

One demerit of this methodology is that I underwent a very time-consuming process during labelling, as a slight labelling mistake or inaccuracy could lead to wrong predictions and a wrong output.

FUTURE WORK

"When algorithms are left to work independently, they discover and identify the interesting hidden patterns or groupings within a dataset that would not have been identified by using supervised algorithms. In the coming years, as the language evolves, more improvements in supervised machine learning algorithms can be seen. It's no wonder to say that this ML application will indeed affect the future of ML and result in more accurate analysis" (Medium 2021).

Since my model detects faces in human beings, it could further be extended to recognize human beings. It could be used in biometric scans such as face unlock on phones. If further research can be done given the chance it can also work as or action detection to detect crimes and to solve other incidents.

REFERENCES

Bressoud, T., White, D., 2020. Tabular Model: Structure and Formats, in: Introduction to Data Systems. Springer International Publishing, pp. 145–173. doi:10.1007/978-3-030-54371-6_6

Chowdhry, A., 2014. Facebook's DeepFace Software Can Match Faces With 97.25% Accuracy. Forbes. Available at: https://www.forbes.com/sites/amitchowdhry/2014/03/18/facebooks-deepface-software-can-match-faces-with-97-25-accuracy/?sh=3afbc80d54fc [Accessed April 16, 2021].

(moodlecurrent ,2021)"Applied machine learning course work" Available at: https://moodlecurrent.gre.ac.uk/pluginfile.php/2149543/assignsubmission_file/submission_files/569379/Redwan_Ahmed_Machine%20learning%20Report.pdf?forcedownload=1

Medium. 2021. *What is the Future of Machine Learning?*. [online] Available at: <https://codeburst.io/what-is-the-future-of-machine-learning-f93749833645> [Accessed 16 April 2021].

(Stackoverflow, 2021) "Unexpected keyword argument 'ragged' in Keras" Available at: https://stackoverflow.com/questions/58878421/unexpected-keyword-argument-ragged-in-keras