University at Buffalo
School of Engineering and Applied Sciences

CSE 666 – Biometrics Image Analysis
Spring 2023

Assignment #1

Prepared by

Sakshamdeep Singh - 50475857

March 11, 2023

TABLE OF CONTENTS

Submission Folder Structure

```
└── 50475857_ssingh86_assignment01.zip
    ├── 50475857_ssingh86_assignment01_report.pdf
    ├── count_faces.jpg
    ├── data/
    │   ├── congress.tsv
    │   └── img/
    ├── README.MD
    └── src/
        ├── main.py
        ├── util.py
        └── results/
            ├── task1_results/
            │   ├── annotated_faces_boxes_vgg.csv
            │   ├── annotated_faces_boxes_json_vgg.json
            │   └── count_faces_vgg_annotations.jpg
            ├── detected_faces/
            ├── task_2_count_faces_annotated_faces.png
            ├── task_3_count_faces_annotated_faces.png
            ├── task_4_count_faces_annotated_faces.png
            ├── task_5_count_faces_annotated_faces.png
            ├── task_6_embeddings.npz
            └── task_7_count_faces_annotated_faces.png
```

External Libraries Used:

	OpenCv	Deepface	Face-reconition
Task 1: Annotation			
Task 2: Face Detection	✓		
Task 3: Expression Analysis	✓	✓	
Task 4: Gender Analysis	✓	✓	
Task 5: Face Pose Estimation	✓		✓
Task 6: Feature Extraction	✓		✓
Task 7: Face Recognition	✓		✓

Task 1: Annotation

- For the manual annotation, the VGG Image Annotator was used to annotate images and save the corresponding bounding boxes.
- The VGG Image Annotator can be found at: (https://www.robots.ox.ac.uk/~vgg/software/via/)
- Number of faces annotated = 135.

Figure 1: Faces annotated using VGG Image Annotator

Task 2: Face Detection

- For face detection, OpenCV library's Haar cascades, which are pre-trained machine learning models used for detecting faces, were used.
- Once the faces are detected, we will proceed with the next steps.
- Number of faces detected = 116.

Figure 2: Faces detected using OpenCV Haar cascade.

Evaluation:

- Number of faces detected = 116
- Number of faces annotated = 135
- Accuracy of detecting faces = 116/135 * 100 = 85.92%
- Out of total 135 manually annotated faces, only 116 were identified. Some of the faces were missed due to the following reasons.
    - Face is tilted on one side so face is not aligned vertically.

    - Person is looking sideways.



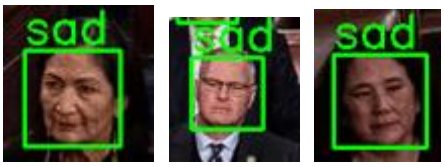
- Person is looking down.

- Occlusion


Task 3: Expression Analysis

- For this task, the deepface Python library was used, which internally uses VGG-Face as the default deep learning model.
- Out of 116 faces detected, each face was passed to the deepface API to obtain the emotion.
- The following six emotions were detected:

    - Sad: 47

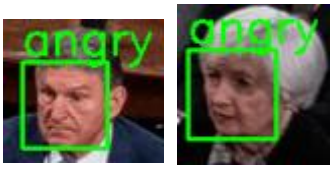
    - Neutral: 33

    - Fear: 6

    - Angry: 24

    - Disgust: 1

    - Happy: 5

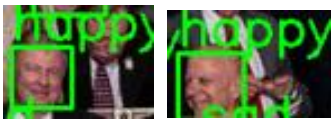
      

Figure 3: Emotions detected using deepface.

Evaluation:

- Emotions: ['neutral', 'sad', 'angry', 'happy', 'fear', 'disgust']
- Counts: [33, 47, 24, 5, 6, 1]
- Deepface library does a pretty good job at evaluating the sentiments.
- Most of the sentiments, as shown above, were identified correctly.
- One error it made is overclassifying ‘sad’ emotion, where most of the people in the room seem to have a neutral expression.

Task 4: Gender Analysis

- For gender analysis, the deepface Python library was used, which internally uses VGG-Face as the default deep learning model.
- The model identified 107 faces as male and 9 faces as female.

Figure 4: Gender detection using deepface library.

Evaluation:

- Emotions: ['Man', 'Woman']
- Counts: [107, 9]
- Most of the genders, as shown above were identified correctly. However, a lot of women were misclassified as men.
- Total Number of Men in the picture = 88
- Total Number of Women in the picture = 28
- Accuracy of the model = 97/116 * 100 = 83.62%
- Confusion Matrix:

	Man (Ground Truth)	Woman (Ground Truth)
Man (Predicted)	88	19
Woman (Predicted)	19	9

Task 5: Face pose estimation

- For face pose estimation, the yaw angle will be calculated to detect whether the person is looking forward or sideways.
- The face_recognition Python library was used to obtain facial landmarks.
- From the facial landmarks, we can obtain the nose bridge and chin to calculate the yaw angle.

Figure 5: Yaw angle detection using face_recognition library.

Evaluation:

- Model didn't seem to perform well using this approach.
- Probably the approach using nose bridge and chin to calculate the yaw angle is not robust enough.
- Adding more landmark features will help with the accuracy but in some of the faces, due to the small image size, fine landmarks are not detected properly.

Task 6: Feature Extraction

- For feature extraction, the face_recognition library was used to obtain the embeddings of each extracted face from the original image.
- Each extracted face was converted into an embedding of a 128-sized NumPy array.
- These embeddings/encodings were used in the next task for face recognition.

Task 7: Face Recognition

- For the given dataset, a dictionary of embeddings was created and stored in memory.
- For each Region of Interest (ROI), an embedding was created and matched with the list of known embedding stored in the above step.
- Euclidean distance was used as a similarity criterion.
- Matches are then stored in detected_faces directory with file name as the identified person's name.

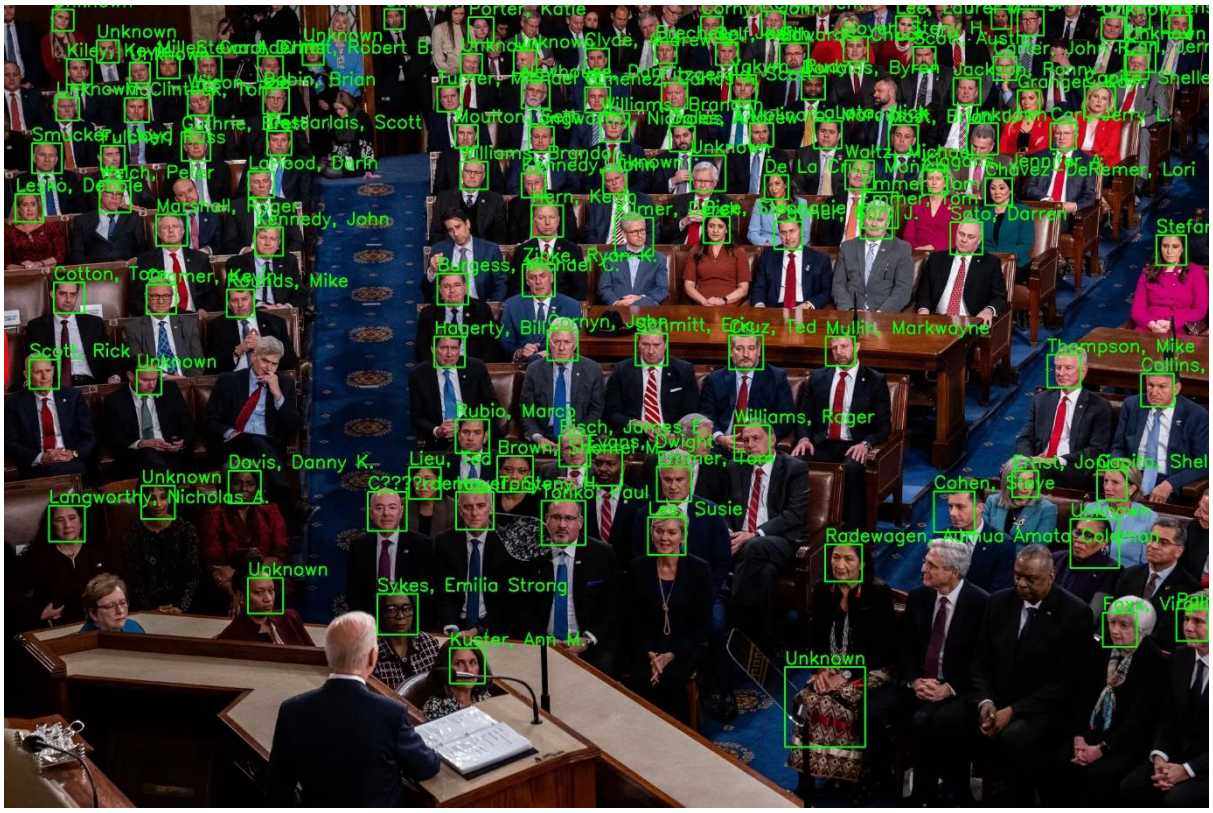


Figure 6: Faces recognized using Euclidean distance similarity function in face_recognition.

Evaluation:

- Number of Known faces: 95
- Number of Unknown faces: 21
- On crosschecking the 95 known faces manually,
    - Number of correctly identified faces = 67
    - Number of incorrectly identified faces = 28
    - Accuracy = 67/95 * 100 = 70.52%
    - Accuracy is pretty good considering the amount of data we had.

References

- https://www.robots.ox.ac.uk/~vgg/software/via/
- https://github.com/serengil/deepface
- https://face-recognition.readthedocs.io/en/latest/
- http://dlib.net/python/index.html#dlib.get_frontal_face_detector
- https://pypi.org/project/deepface/
- https://pypi.org/project/face-recognition/
- https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://en.wikipedia.org/wiki/Euclidean_distance
- https://github.com/ageitgey/face_recognition/tree/master/examples
- https://docs.python.org/3/