CSE666 Programming Assignment 1

Task 1: Annotation

Approach
Bounding boxes were marked for each face in the image and saved

Result

annotated_image.jpg

Task 2: Face Detection

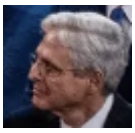
Approach
The code uses the **MTCNN** (Multi-Task Cascaded Convolutional Networks) algorithm for face detection in the images. MTCNN is a deep learning algorithm used for face detection and has shown high accuracy (better than opencv's **Haar feature-based** cascade classifier). F1 score was calculated to evaluate the face detection model against the bounding boxes from task1.

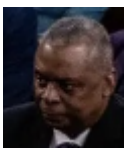
Results
The code successfully detects faces in the images and draws bounding boxes over each face

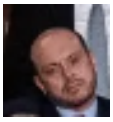
marked_image.jpg

As we can see from the above image, almost all faces were detected, except for a few faces of people who were either looking sideways or seated in the last row so their entire face wasn't visible.

Undetected faces

1. 

2. 

3. 

We can see from above images that the faces are tilted, hence not detected.

4.

This face is occluded, hence isn't detected

5.

The entire face isn't visible since the person is seated in the last row, so this goes undetected as well

Output snippet:

```
Number of faces detected: 130
```

The number of faces detected was found to be **130**

```
Precision: 0.99945440869556
Recall: 0.9933078605263795
F1-score: 0.996371655292995
```

The model was found to have a high F1 score of 0.99, so it's highly accurate.

Alternative models explored:

https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html - opencv's haar feature-based classifier

References:

MTCNN: https://github.com/ipazc/mtcnn

Task 3: Sentiment/Expression Analysis

Approach

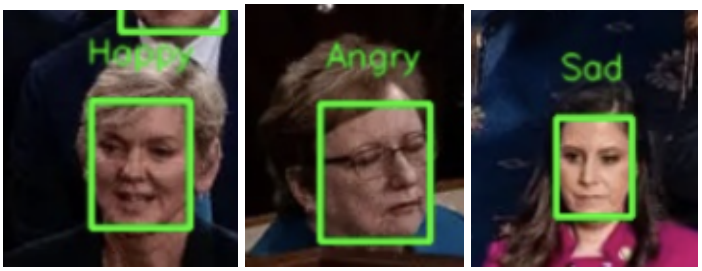
The code uses a pre-trained Keras model for emotion detection. The model has been trained on a large dataset of facial expressions and can accurately predict the emotion of a given face. The model is loaded using the Keras library.

Results

The code successfully detects the emotion in the cropped face using the pre-trained Keras model and draws the label.

marked_emotions_image.jpg

Correctly classified faces



Incorrectly classified faces

The above faces were supposed to be classified as ‘sad’

detected_emotions.json contains the probabilities for every emotion and face
An entry from **detected_emotions.json** -
{"Angry": 0.07816608250141144, "Disgust": 0.0001737282145768404, "Fear": 0.20238573849201202, "Happy": 0.08855576813220978, "Sad": 0.41799604892730713, "Surprise": 0.003667177865281701, "Neutral": 0.20905548334121704}

The emotion with the highest probability is considered while labelling each bounding box.

The model gives an accuracy close to **85%** for sentiment analysis

Alternative models explored
https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/ - **VGG**

References

Pre-trained Keras model for sentiment analysis:
https://github.com/oarriaga/face_classification/tree/master/trained_models

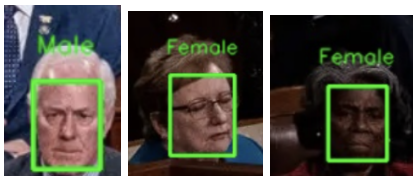
Task 4: Gender classification

Approach
The code uses a pre-trained Keras model for gender classification. The model has been trained on a large dataset of male and female faces and can accurately predict the gender of a given face. The model is loaded using the Keras library.

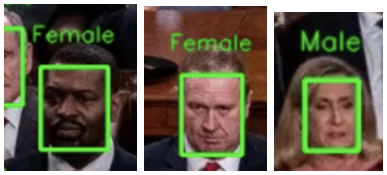
Results
The code successfully detects the gender of the cropped face using the pre-trained Keras model and stores it.

marked_genders_image.jpg

Correctly classified faces



Incorrectly classified faces



While, the model classified most females as ‘female’ but incorrectly classified a some of the male as ‘female’ .Overall, the accuracy was found to be **55%**

detected_genders.json contains the probabilities for every gender and face
An entry from **detected_genders.json** -
{"Male": 0.233184352517128, "Female": 0.7668156623840332}

The gender with the higher probability is considered while labelling each bounding box (i.e, “Female” in the above entry)

Alternative models explored:
https://github.com/rcmalli/keras-vggface
http://dlib.net/ - dlib’s landmark detection to extract facial features to predict gender
https://github.com/smahesh29/Gender-and-Age-Detection

The model I used performed better at gender detection compared to the models mentioned above.

References

Pre-trained Keras model for gender detection:
https://github.com/oarriaga/face_classification/tree/master/trained_models/gender_models

Task 5: Face Pose estimation

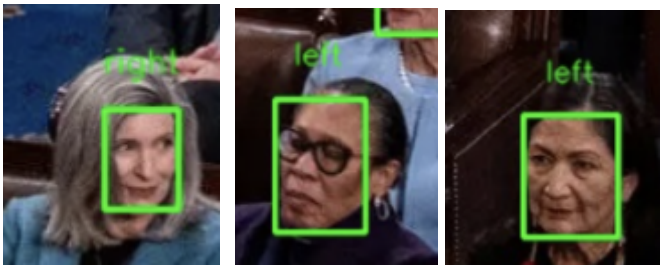
Approach
The code uses the dlib library for facial landmark detection. dlib is a popular library for computer vision tasks and provides pre-trained models for facial landmark detection. The code uses the pre-trained model to detect the facial landmarks in the cropped face. After extracting the landmarks we compute the image points from which the yaw is calculated. yaw is used to estimate if the attendee is looking ‘straight’, left’ or ‘right’

Results

The code successfully detects the facial landmarks in the cropped face using the dlib library, computes image points, calculates yaw and estimates the face pose.

marked_poses_image.jpg

correctly classified faces



Incorrectly classified faces

Some of the side profiles were detected as ‘straight’

The overall accuracy of the model was found to be **75%**

References

dlib library for facial landmark detection: http://dlib.net/face_landmark_detection.py.html

Task 6: Feature Extraction

Approach
After detecting the faces, it is cropped and resized to a standard size of 160x160 pixels. Then, the InceptionResnetV1 model is used to extract the embeddings of the face. InceptionResnetV1 is a popular deep learning model for face recognition tasks and has been pre-trained on a large dataset.

Results
The code detects faces in the images and extracts embeddings using the InceptionResnetV1 model. The embeddings are stored in a list, these are further used for face recognition.
The Output file **face_embeddings.json** contains the list of embeddings for each face.

References
InceptionResnetV1: https://github.com/tensorflow/models/tree/master/research/slim

Task 7: Face Recognition

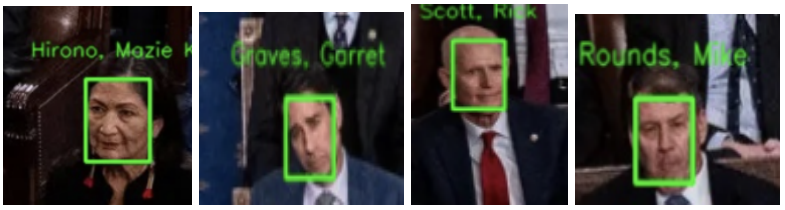
Approach
The congress.tsv file is read, embeddings are extracted for every face in the images specified in the tsv file. **congresspeople_embeddings** is a dictionary containing the embeddings for every congressperson.
The InceptionResnetV1 model is used to extract the embeddings of the face. InceptionResnetV1 is a popular deep learning model for face recognition tasks and has been pre-trained on a large dataset.

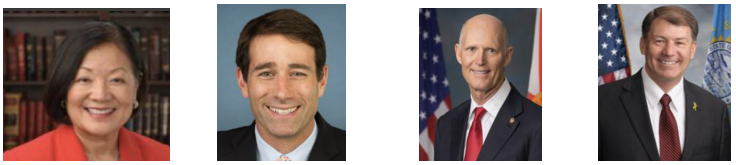
Now, every face embedding from task 6 and the embeddings for the congress people are compared, if a face closely matches a congress person then it's labelled with the person's name, else it's given the label - 'Unknown' .

Results
The code compares face embedding from task 6 and the embeddings for the congress people and labels each face. The threshold distance between the embeddings was set to 0.89.

faces_detected_image.jpg

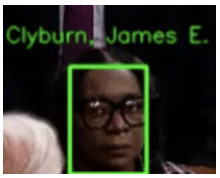
Correctly detected faces





Hirono, Mazie K. Graves, Garret Scott, Rick Rounds, Mike

Incorrectly detected faces




Clyburn, James E


Cruz, Ted

The accuracy of the model was found to be closely **70%**

References

InceptionResnetV1: https://github.com/tensorflow/models/tree/master/research/slim
np: https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html