# CSE666 Programming Assignment 1

---

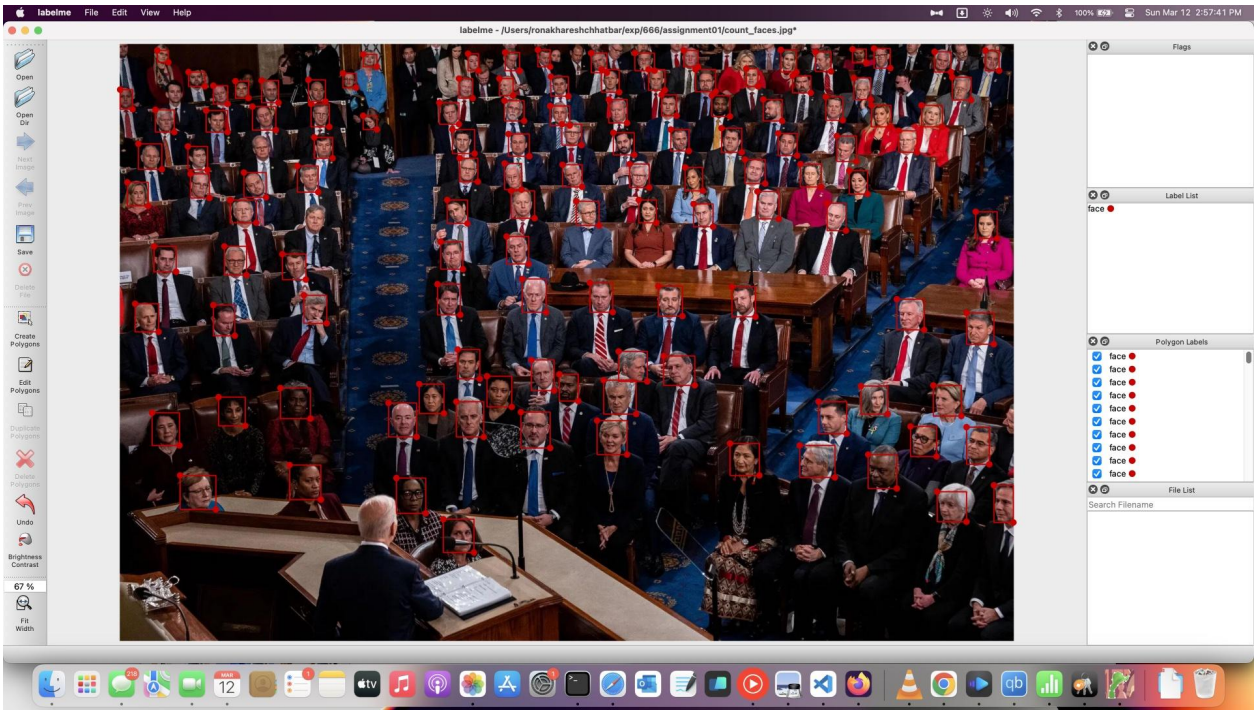
**Professor - Nalini Ratha**

**Ubname - ronakhar; person # - 50477951**
**Teammates - Ronak Haresh Chhatbar**

---

**Task1**: Annotation Mark bounding boxes for each face in the image and save them using any tool.

For simplicity, I am using **lableme** It provides multiple functionality, but in our case we only need to use bounding boxes.




I have labeled 129 images in the picture provided to us.

## Task2:

## Face detection: detect faces present in the image, label each region of interest (RoI) containing a face with a bounding box, and return the number of faces present in the image, i.e., the number of detected bounding boxes. Evaluate face detection models (deterministic models and/or pretrained networks) against bounding boxes from step 1.

I am utilizing the face-image-analysis repository for detecting faces in images. The MTCNN model is employed to identify the facial region of interest for each face. By applying the MTCNN face model to the "congress.jpg" image, 133 face images are extracted. After experimenting with different threshold values, it was determined that a value of 0.98 on "congress_faces.jpg" results in the maximum number of faces detected without false negatives.

```
def get_bounding_box(frame):
```

The "get_bounding_box" function retrieves 133 bounding boxes in x1, x2, y1, y2 format. In our case, the bounding boxes are crucial for the subsequent step, as we aim to perform additional analysis on each individual face.

```
def get_iou(bounding_boxes,hand_bbx,threshold):
```

The `get_iou` function takes three inputs, namely the bounding boxes generated from the MTCNN model, the groundtruth generated from labelme, and the IOU threshold. The function returns the count of images that have an IOU greater than the specified threshold.

**Result evaluation:**

```
with threshold 0.4 we get :126 number of bounding boxes
with threshold 0.8 we get :126 number of bounding boxes
```

We have good bounding boxes from the IOU comparison.

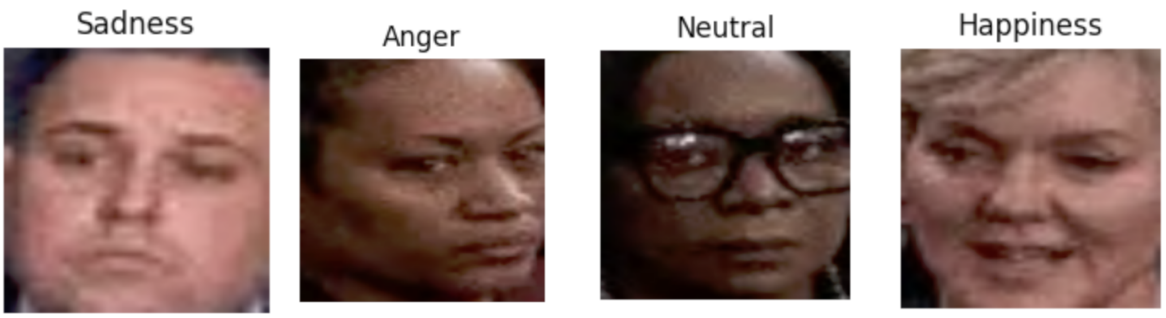
**Task3**:
Sentiment/Expression Analysis: For each identified ROI, analyze the expression to return metrics like anger/disgust/sadness, etc. Evaluate the performance of your model.

I am utilizing the face-emotion-recognition library for sentiment analysis. To improve inference speed, we are using the mobile_net7 model, which has seven output classes:
`{0: 'Anger', 1: 'Disgust', 2: 'Fear', 3: 'Happiness', 4: 'Neutral', 5: 'Sadness', 6: 'Surprise'}`

```
def get_emotions(frame,bounding_boxes,points):
```

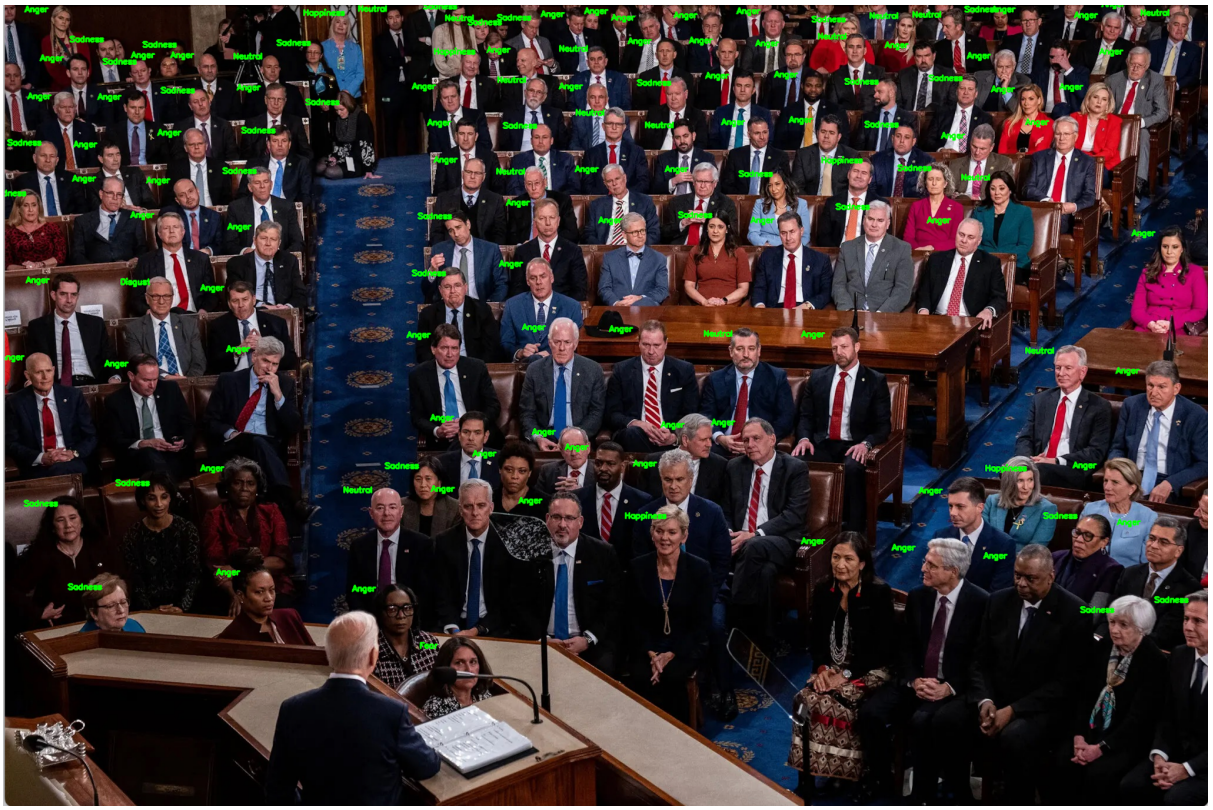
The "get_emotions" function takes the full image, along with the bounding boxes obtained in the previous step, and iterates over all 133 regions of interest (ROI) by slicing the image. The sliced image is then resized to (224, 224), as required by the mobile_net7 model, and passed for inference. This provides us with the appropriate prediction for each ROI, which is then saved in an "emotions" folder. Each detected face is saved as a separate file with the respective emotion as the filename.

Therefore, the output for each individual image is stored in the "emotion" folder, with the filename in the format "<**file_number_emotion**>.png".



Since we don't have the evaluation dataset to compare our results, I have eyeballed the figures and noticed that many faces are predicted to be angry. As we don't have the right ground truth, I am hypothesizing that maybe the speaker was bringing up the topic that caused these faces some agitation, and the model predicted they would be angry.

Our image is 2048 X 1365 and contains 130 plus faces; the individual image faces get squashed and become pixelated. I would encourage the grader to go through the images and check that may of the images



**Emotions.png**

**Task4:**
Gender Classify each detected face by gender. Evaluate the performance of your model against gender information from the dataset.

For the task of gender classifcation I have used the git hub face_classification simple_CNN.81-0.96.hdf5 is provided and is a tensorflow model.

The model takes input dimension (48, 48,3) and returns the gender in this model we have only two gender that is '**man'** and **'woman**'

When just  passing the forehead-to-chin image, the model fails to predict the gender of the face. This is overcome by adding an extra boundary.

```
face_img=frame[y1-20:y2+10,x1-20:x2+10,:]
```

The model makes better predictions. This shows that the model needs to see the forehead and head-hair features to determine if it is a man or woman.

```
def get_gender(frame,bounding_boxes,gender_labels):
```

The function above takes three arguments one the image frame, the second argument it takes is bounding boxes to slice the image according to the bbx
Once we have ROI we

Upon applying the model, we classify 24 ROI as women and 109 as men.

The outputs for individual image is stored in **gender** folder with **file_name<gender_of person>.png**
For example, if the predicted gender is man, then the file name is file_name_1_man.png, and similarly for women, file_name_48_woman.png.

There are about 29 women in Congress_image.jpg; this includes those at the back and in the semi-faces:
While from it the model has identified total-prediction of
Total women predicted by model= 24
Total man-prediction: 109
13 women predicted right
11 man predicted wrong as women
All other 109 prediction are man
Precision for Women = Women right prediction / (women right prediction + man wrong prediction) = 12 / (12 + 17) = 0.41
Recall (also known as sensitivity) for women = women right prediction / ground truth of class 1 = 12 / 29 = 0.41
Precision for man = man right prediction / (man right prediction +women wrong prediction) = 92 / (92 + 17) = 0.84

Recall (also known as sensitivity) for man = man right prediction / ground truth of man = 92 / 104 = 0.88

F1 score for women = 2 * precision for women * recall for women / (precision for women + recall for women) = 2 * 0.41 * 0.41 / (0.41 + 0.41) = 0.41

F1 score for man = 2 * precision for man * recall for man / (precision for man + recall for man) = 2 * 0.84 * 0.88 / (0.84 + 0.88) = 0.86

F1 = (0.41 * 29 + 0.86 * 104) / (29 + 104) => 0.79

FIG-1

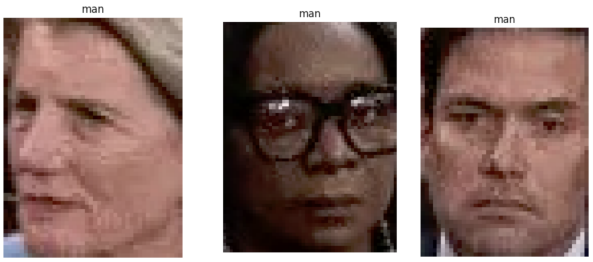
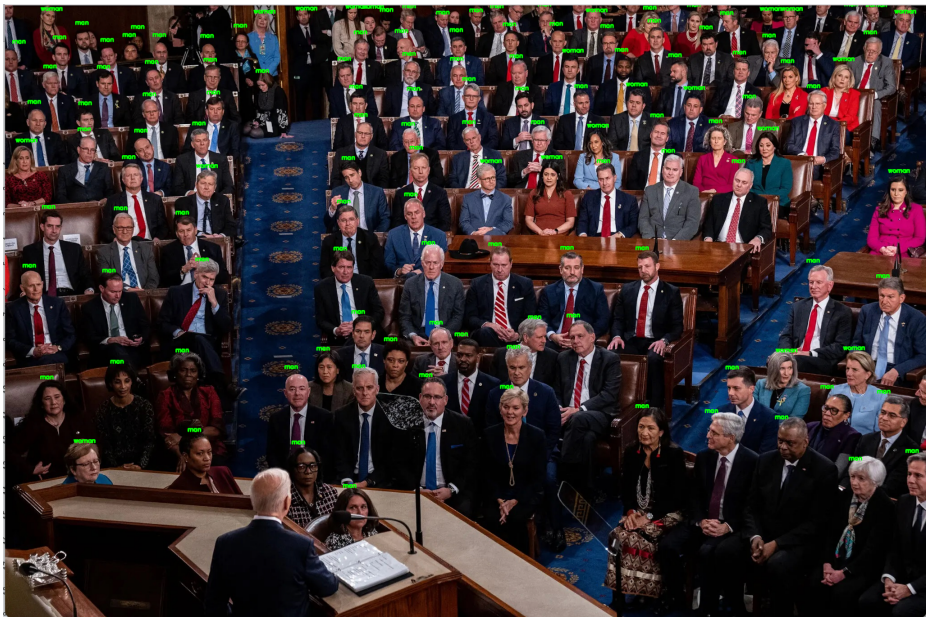


FIG-2

We notice in the fig-2 when the ROI is tightly bounded to the face the model predicts wrong results since it is not able to discern the facial features but for the same thing when we increase the ROI



**gender.png**

**Task5:**

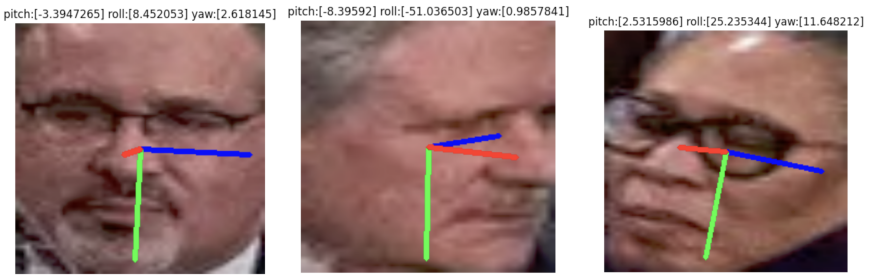
Face pose estimation: For each identified ROI, estimate the face pose to determine whether the attendee is looking straight ahead or to the side. Evaluate the performance of your model.

For the task of face pose estimation, I am using the [ Rotation Representation for Unconstrained Head Pose Estimation] library.

```
def get_headpose(frame,bounding_boxes):
```

The function takes two arguments. first is the ndarray of image, and second are the bounding box points; just like the previous cases, we apply the bounding boxes and get ROI for each face. Once we get the ROI, we resize it to (224, 224, 3) and then pass it to the model. The model outputs three values, which are pitch, yaw, and roll for the face. With these values, we understand a person's attention in the 3D world.

Upon getting the yaw pitch and roll from the model, in the above function we can use the values to get the right or left tilt of the face.
The output for all faces is stored in **headpose** folder
With the naming convention **image_name_right_tilt.png/ image_name_left_tilt.png**




Using the Yaw value I am predicting wether the head is tilting right or left

```
if yaw < -1.0:
      face_tilt = ("left_tilt")
elif yaw > 1.0:
      face_tilt = ("right_tilt")
else:
      face_tilt = ("no-tilt")
```

Since we don't have the ground truth for the yaw, pitch, and roll, I am eyeballing the images; the model performs well with yaw, pitch, and roll to determine the right and left.
Please check the headpose folder.

**Task6:**

Feature extraction: extract features for each face detected using pretrained models and store the generated embeddings for matching.

Feature extraction is used to save a representation of the face at a minimal value, while we can make the output of the feature according to our requirements.
For this task, I have utilized the implementation of ARC-face to load the model.
We load the model using this file and then generate the embeddings using the above model. The model is a Res34Arch model, and we generated an (1, 512) embedding for the given face. Using the embedding, we can compare faces and understand their similarity.

We implement the same face pipeline structure where we provide with an image and its corresponding boundingboxes and then slice the ROI for everyface and pass it to the model The input for the arc face model is size (112, 112, 3). We resize our ROI to this size and pass it to the model. The model generates a vector of size (1,512). This is a latent representation of the face using arcface.

```
def generate_embds(frame,bounding_boxes):
```

The function takes two arguments. first is the ndarray of image, and second are the bounding box points; just like the previous cases, we apply the bounding boxes and get ROI for each face. Once we get the ROI, we resize it to (112, 112, 3), and then pass it to the model. The model returns a vector that can be saved for an individual person using this embedding (1,512). We can use either cosine similarity or elucidian distance. In my case, I have taken the elucidian distance.

```
def save_embeddings(embeddings):
```

Save the embeddings of all 113 faces  into an numpy array of size (133,[1,1512])

## Task7:

Face recognition: Given a dataset containing images and names of lawmakers, identify each ROI and return a label with the Senator/Congressperson's name. (Note that not all attendees present will be included in the dataset since not all attendees are lawmakers. For such cases, your solution must return the label "Unknown."). Evaluate the performance of your matcher.

```
def make_comparassion(frame,bounding_boxes,encodings_of_sennators):
```

The above function takes three arguments. one is the image frame, the second is the bounding boxes of all faces, and the third is encodings_of_senators, which acts as a database for making comparisons.

Using the Arcface model, we generate two embeddings:

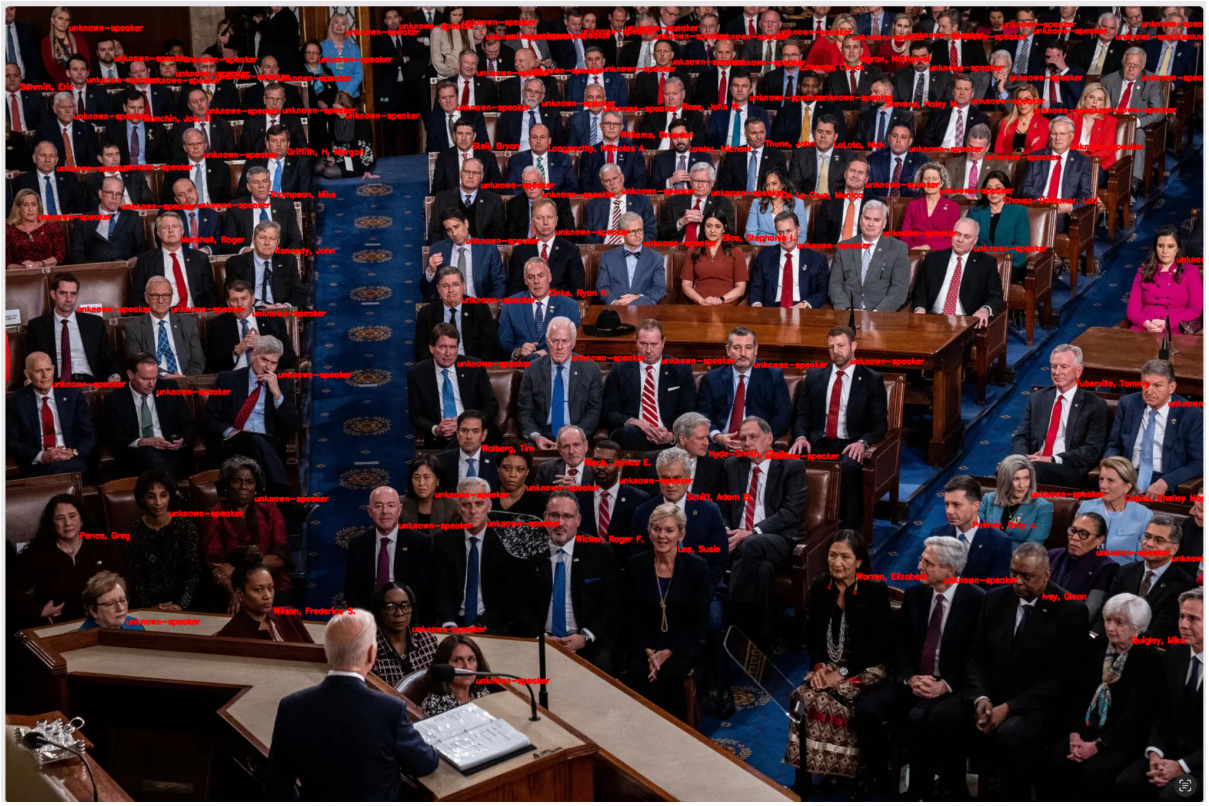
1) Database = we apply facedecteor to the dataset folder crop the face and then pass it through the Arcface model for every face we generate an embedding and store them in a pickle file
   Output: encodings_of_sennators.picklefaces

Example: {"name of sennator":[1,512]}
We do this for all images in the dataset folder we have 489 embeddings in our database

```
for name, encodings in encodings_of_sennators.items():
            dist = EuclideanDistance(encodings, temp_encoding_face_encoding)
            # print(dist)
            if min_dist > dist:
                min_dist = dist
                if name in list_names:
                    list_names.remove(name)
                    identity = name
                else:
                    identity = 'unknown-speaker'
```

Now the comparison, We take the embeddings generated in the previous **task6**
For every face in count_faces.jpg, we make a running comparison of a single face with all the faces in the database, select the match with the lowest score (Elucdian distance), and assign the label to the face. Once an assigned face is removed, we can't have duplicates. We might lose the right prediction if for the correct face the confidence is low using the remove face once the face is matched the ideal comparison will be
Every face compared with all 489 and select the lease score facematch that should give an perfect match.

In this task, we have low accuracy, but the faces that are matched are of high confidence.



**Comparison.png**

**References:**

Task1:

https://github.com/wkentaro/labelme

Task2:

https://github.com/ipazc/mtcnn

**Task3:**

**https://github.com/HSE-asavchenko/face-emotion-recognition**

**Model-link:** mobilenet_7.h5

```
@inproceedings{savchenko2021facial,
  title={Facial expression and attributes recognition based on multi-task learning of lightweight neural networks},
  author={Savchenko, Andrey V.},
  booktitle={Proceedings of the 19th International Symposium on Intelligent Systems and Informatics (SISY)},
  pages={119--124},
  year={2021},
  organization={IEEE},
  url={https://arxiv.org/abs/2103.17107}
}

@inproceedings{Savchenko_2022_CVPRW,
  author    = {Savchenko, Andrey V.},
  title     = {Video-Based Frame-Level Facial Analysis of Affective Behavior on Mobile Devices Using EfficientNets},
  booktitle = {Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops},
  month     = {June},
  year      = {2022},
  pages     = {2359-2366},
  url={https://arxiv.org/abs/2103.17107}
}

@article{savchenko2022classifying,
  title={Classifying emotions and engagement in online learning based on a single facial expression recognition neural network},
  author={Savchenko, Andrey V and Savchenko, Lyudmila V and Makarov, Ilya},
  journal={IEEE Transactions on Affective Computing},
  year={2022},
```

```
  publisher={IEEE},
  url={https://ieeexplore.ieee.org/document/9815154}
}
```

Task4:
https://github.com/oarriaga/face_classification
Model-link:
https://github.com/oarriaga/face_classification/blob/master/trained_models/gender_models/simple_CNN.81-0.96.hdf5

**Task5:**
**https://github.com/thohemp/6DRepNet**
Model-link:
https://drive.google.com/drive/folders/1V1pCV0BEW3mD-B9MogGrz_P91UhTtuE_?usp=sharing

```
@INPROCEEDINGS{9897219,
  author={Hempel, Thorsten and Abdelrahman, Ahmed A. and Al-Hamadi, Ayoub},
  booktitle={2022 IEEE International Conference on Image Processing (ICIP)},
  title={6d Rotation Representation For Unconstrained Head Pose Estimation},
  year={2022},
  volume={},
  number={},
  pages={2496-2500},
  doi={10.1109/ICIP46576.2022.9897219}}
```

**Task6:**
Arcface-paper
Model-loading-file
Repo-used

**Task7:**
Arcface-paper
Model-loading-file
Repo-used