Name: Sachin Palahalli Chandrakumar

EmoSense

Understanding Group Emotions through Individual face analysis

Description of the Problem:

Given the images of the groups, it is very difficult to gauge the collective mood of the group. So, the aim of this project is to analyze group images and identify emotions of the individuals within them using deep learning models and libraries.

Implementation:

To find a solution to the above problem, I have used multiple deep learning models. To detect the faces, I have used the yolo v8n-face model to detect the faces in the images. Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model used for image/video detection.

First, the faces are detected in an image using the YOLO model and the detected faces are extracted. We then use DeepFace library to classify the emotions of each face of the given image.

Steps involved in the implementation of the above approach:

1. Load the given model
2. Load the image which needs to be analyzed
3. Perform face detection on the given image and extract the bounding boxes and the corresponding faces respectively.
4. Use the DeepFace library to classify the emotions of each face in the given image.
5. Determine the Collective mood of the group using the Majority vote algorithm

Validation of the implemented approach:

To validate the implemented approach, I am using the data provided by interview Kickstart. I labeled around 50 diverse group images using Label-Studio and then exported the validation data in the YOLO format(Popular TXT format is created for each image file. Each txt file contains annotations for the corresponding image file, that is object class, object coordinates, height & width.)

A map of the emotion classes is created based on the labeling and then the true bounding box and the true emotions for each of the faces of each image is extracted.
For all the validation data, the true_bounding_box values and true_emotions are extracted from the validation data. The predicted bounding box values and also the predicted emotions values are obtained by performing the above steps of the implementation mentioned. Now, to analyze if the detections of the bounding boxes are correct , we use intersection over union to analyze how correctly the bounding boxes are predicted. Similarly, we obtain precision, recall, f1 score and accuracy to validate the emotions predicted and the group emotion.

Challenges Tackled:

1. One of the issues validating the data is that the predicted faces in a given image might not be in the order of the faces labeled. Hence, it is very important to make sure that we are comparing the right faces or else the analysis will be wrong.
To overcome this problem, for each labeled face bounding box, I find the best predicted bounding box and then I will map that corresponding face prediction values(bounding box and emotion) to the labeled face values. This helps in making sure that we are comparing the correct faces. The disadvantage of this approach is that, computationally it is expensive as we need O(m*n) time per image to map labeled and predicted faces where m is the number of faces labeled and n is the number of predicted faces for a given image. Another disadvantage is that, we are trusting the face detection algorithm to be somewhat accurate to at least have the bounding boxes closer to its true values. I analyzed the face detections and multiple models predict the bounding boxes covering the faces and hence this would actually give the correct face itself.

Labeled Images Order:

Predicted Images Order:

Predicted Images order after mapping to the labeled images:

2. Difference in the number of labeled faces and predicted faces: The other issue arises when the number of faces labeled and the number of predicted faces for a given image is different. In this case, we clearly know that there

is a mismatch in the number of faces labeled and predicted. But to map and analyze the accuracy accurately, for this case I have only considered the faces which were labeled and found a corresponding predicted bounding box using the approach mentioned in point 1. If for a labeled face, I don't find an intersection of the bounding box with any of the predicted bounding box, I'm not considering that face for evaluation. The reason is that, there will be no accurate face to use to compare and analyze it with. Similarly, I will ignore all the predicted faces which were not labeled.
For example: For a given image, if 10 faces were labeled and 13 faces were predicted. Now, out of those 10 faces labeled, if we found corresponding bounding boxes for only 9 faces, then we only consider those 9 faces to be analyzed.
3. The DeepFace method was failing to recognize the faces and had to use enforce_detection=False inorder to classify emotions. However, if the model is improved to recognize that the data passed is actually face or not accurately would help in getting better results. Even though all the data passed were faces, it was unable to detect it.

Evaluation of the implemented approach:
I have considered the below evaluation metrics:

1. **Intersection over union(Iou):** It measures the overlap between the actual bounding box and the predicted bounding box. It helps in analyzing how accurately the face is detected.
2. **Precision:** It measures the accuracy of the positive predictions made by the model. It calculates the proportion of correctly classified emotions to all the emotions predicted by the model. Higher precision value implies that there are fewer false positives
   Precision = True Positives/(True Positives + False Positives)
3. **Recall:** It measures the ability of the model to identify all true positives in the dataset. Higher recall indicates lesser false negatives.
   Recall = True Positive /(True Positives + False Negatives)
4. **F1 Score:** It is a harmonic mean of precision and recall. It provides balance between precision and recall.
5. **Accuracy:** It gives the overall accuracy of emotions predicted

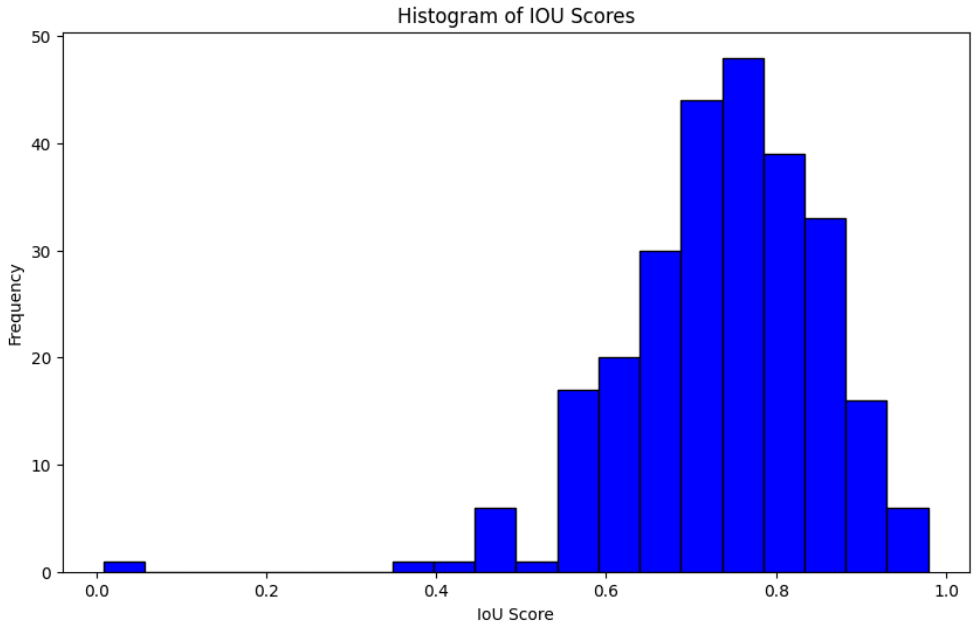
6. **Average Latency:** It gives the average time taken to process and predict desired output for each image.

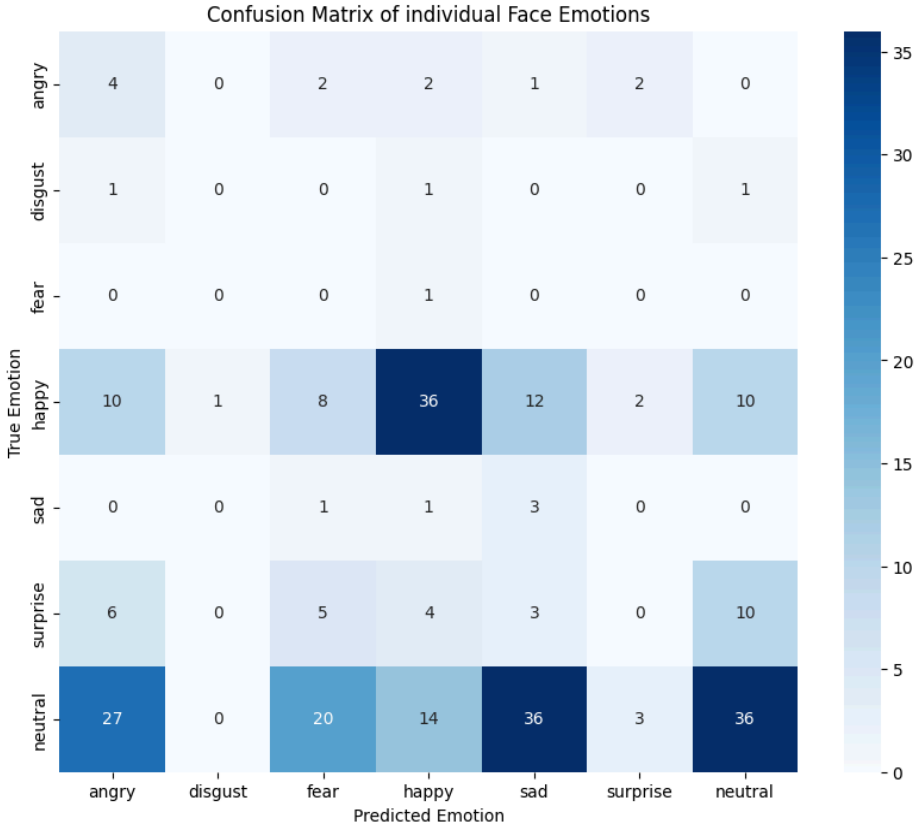
Below are the evaluation results obtained:
Model used = Yolo v8n-face
Super Resolution Model used = False
Emotion Detection using DeepFace Model

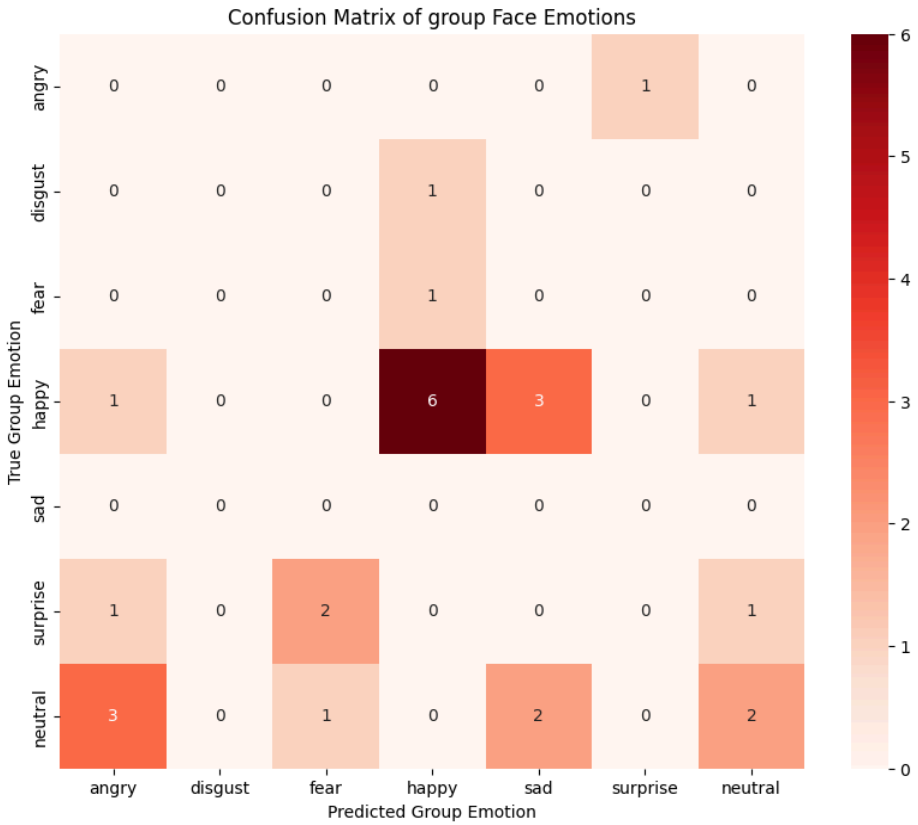
```
IoU Mean: 0.7371
Accuracy: 0.3004
Precision: 0.5144
Recall: 0.3004
F1 Score: 0.3572
Average Latency: 1.7818 seconds
```

We can see that the average IOU score is 0.73, which is a good score implying high overlap between the predicted bounding box and the true bounding box of the faces.
However the other evaluation scores are not good and are below par. One of the major reasons is that, for the detected faces, I can see that the predictions done using deepFace library are not that accurate and hence majorly leading to low accuracy and other evaluation scores. With better models to classify emotions accurately, we can definitely improve the accuracy and precision recall scores. The average latency is around 1.78 seconds from reading the image to detection of faces and then classifying the emotions of each face.

Below are the visual analysis of the results obtained:







Analysis of Different implementations:

1. Since the Emotion Classification using DeepFace was not up to the mark, I used a pre trained emotion detection model to analyze the emotions

   Below are the evaluation results obtained:
   Model used = Yolo v8n-face
   Super Resolution Model used = False
   Emotion Detection using pre trained emotion detection model:

```
IoU Mean: 0.7371
Accuracy: 0.0190
Precision: 0.0004
Recall: 0.0190
F1 Score: 0.0007
Average Latency: 1.9087 seconds
```

However, these results are much worse than the DeepFace implementation. It probably is trained very less and might not be able to generalize for the given validation set of data.

2. I also implemented increasing the resolutions of the images and tried to evaluate the model. Below are the results w.r.t implementing Super Resolution Model. I tried using a heavy model EDSR_x3 but it was taking too much time to even process a single image. Hence, I shifted to using a lighter model FSRCNN_x3
   Model used = Yolo v8n-face
   Super Resolution Model used = True
   Emotion Detection using DeepFace

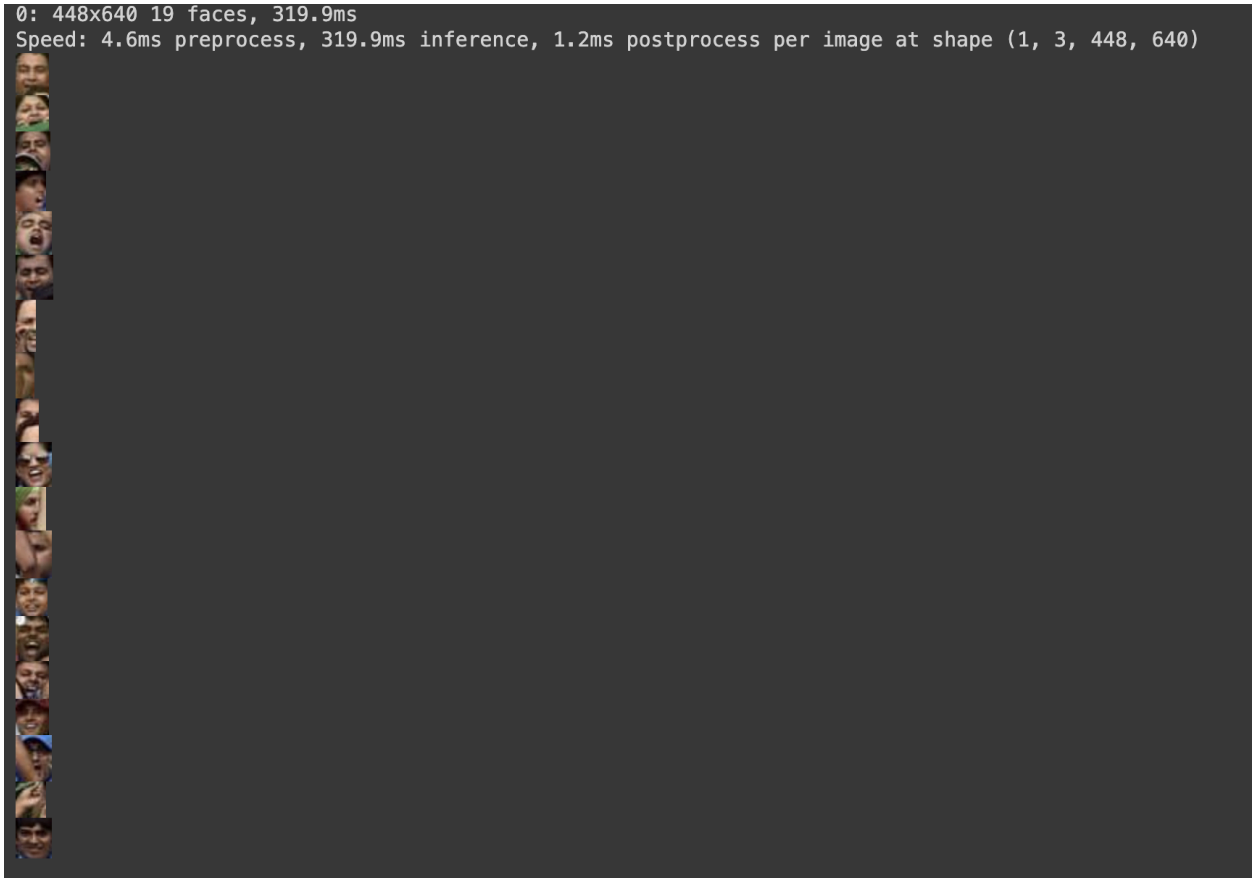
```
IoU Mean: 0.7371
Accuracy: 0.3004
Precision: 0.5144
Recall: 0.3004
F1 Score: 0.3572
Average Latency: 6.0889 seconds
```

We can see that the results are similar to not using the super resolution model. However, the average latency has drastically

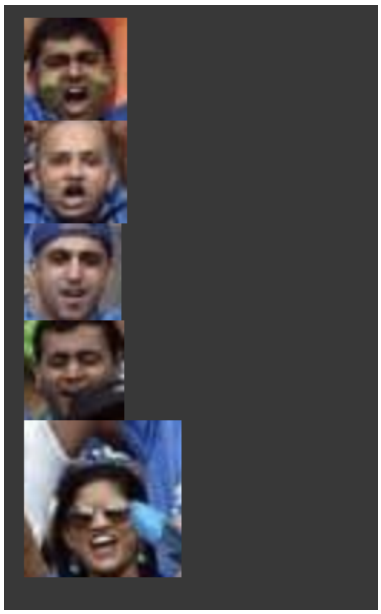
increased to approximately 6 seconds. Given time constraints, this approach might not be good and it also did not improve the results.

3. The Yolo v8n-face model was pretty good in accurately detecting the faces given an image. However, I also tried using non-ML algorithmic techniques such as HaarCascade to extract these individual faces to analyze how it performs. I analyzed that it didn't perform well compared to the Yolo v8n-face model and used to detect much fewer faces compared to the Yolo v8n-face model.
   Example: For the image given:

   Below are the detected images with two different models.
   Faces Detected using Yolo Model:



Faces Detected using HaarCascade Model:



This was the case with other images as well, as faces detected using HaarCascade were much lesser than the Yolo model used. Hence, I used the Yolo model.

Out of all the different implementations, using the Yolov8n-face model of face detection and using DeepFace for emotion recognition gave the best results. Using the SuperResolution method to improve image quality didn't help in improvising the results obtained. Maybe using a better SuperResolution model with a better computational system might help in improving the results.

Future Improvements:

This implementation is still far from perfect. It is a good step at implementing a perfect algorithm to analyze the various emotions in a group.

1. Since, training the Face detection and emotion classification model with the data I had, it would have given better results, especially with emotion classification. Given the time constraint, labeling and training would have taken a lot of time. However, doing so would definitely improve the results.
2. We need a better emotion detection model to predict the emotions of the face as the models I used were very less accurate. Having said already, fine tuning to the data I had would have increased the accuracy.
3. Improvising the labeling method and making sure that we don't have to map the predicted bounding boxes and the true bounding boxes as it would consume more time to match the bounding boxes. Having an efficient implementation would drastically reduce the inference time as well. Currently, only to map the bounding boxes will take O(m*n) time for each image where m is the number of labeled bounding boxes and n is the number of predicted bounding boxes.
4. The lighter version of the SuperResolution Model did not help in improving results and in future should try using a better model with higher computational power system in order to tackle the latency issue.

5. The DeepFace method was failing to recognize the faces and had to use enforce_detection=False inorder to classify emotions. However, if the model is improved to recognize that that data passed is actually face or now accurately would help in getting better results.

Instructions to run the code submitted:

Directories Required:

1. annotations/images
2. annotations/labels
3. models
4. face_path
5. data

Load all the models required in the models folder. I used the below models:

1. yolov8n-face.pt for face detection
2. Face_model.h5 for face emotion classification
3. FSRCNN_x3.pb(lighter model) model for Super Resolution
4. EDSR_x3.pb(heavier model) for Super Resolution

For Validation, I have used the YOLO format after labeling using Label-Studio. Add all the validation images into annotations/images and corresponding labels into annotations/labels.
Face_path is created to store temporary face data, which is used to pass for the DeepFace method.

Run the following commands before running the code:
pip install opencv-python torch torchvision ultralytics
pip install deepface opencv-python

There are two extract_image_data(image_path,model) methods, you can use either of the one based on the model you are using. Here, one methods is implemented based on Yolo face detection model and the other method is implemented based on Haar Cascade detection model.
Similarly, there are two classify_emotions(faces) methods. You can use either one. One method uses the DeepFace method to classify emotions

while the other uses a pre pre-trained model to classify emotions. By default, I'm using the DeepFace method.

The code written is very scalable, as we don't have to change anything internal and can just plug the methods which we need to use accordingly and hence allows the scope to add new implementations without worrying about altering the code too much.