Week 2 Analysis

Ahmad Sarmad Ali

Face Detection Models:

Face detection model selection depends upon many factors but one of the most important factor is to test our detector on a **Benchmark** dataset. This dataset should have following properties must if I say more precisely it should contain the faces in following events:

DataSets	Make up	Illumination	Scale	Pose	Occlusion	Expression	Etnicity	Link
Wider Face	✓	✓	✓	✓	✓	✓	✓	Link
FFHQ	✓	✗	✗	✓	✓	✗	✓	LINK
UTKFace	✓	✓	✗	✓	✓	✓	✓	LINK

The wider face dataset is a complete face detection benchmark dataset used by many researchers, testing of open source face detectors and in competitions. Every new model or paper use this dataset as benchmark.

We have used the Wider face dataset for validation of RetinaFace, SCRFD and YOLOv5.

Wilder face dataset has 3 levels of difficulty: easy, medium and hard.

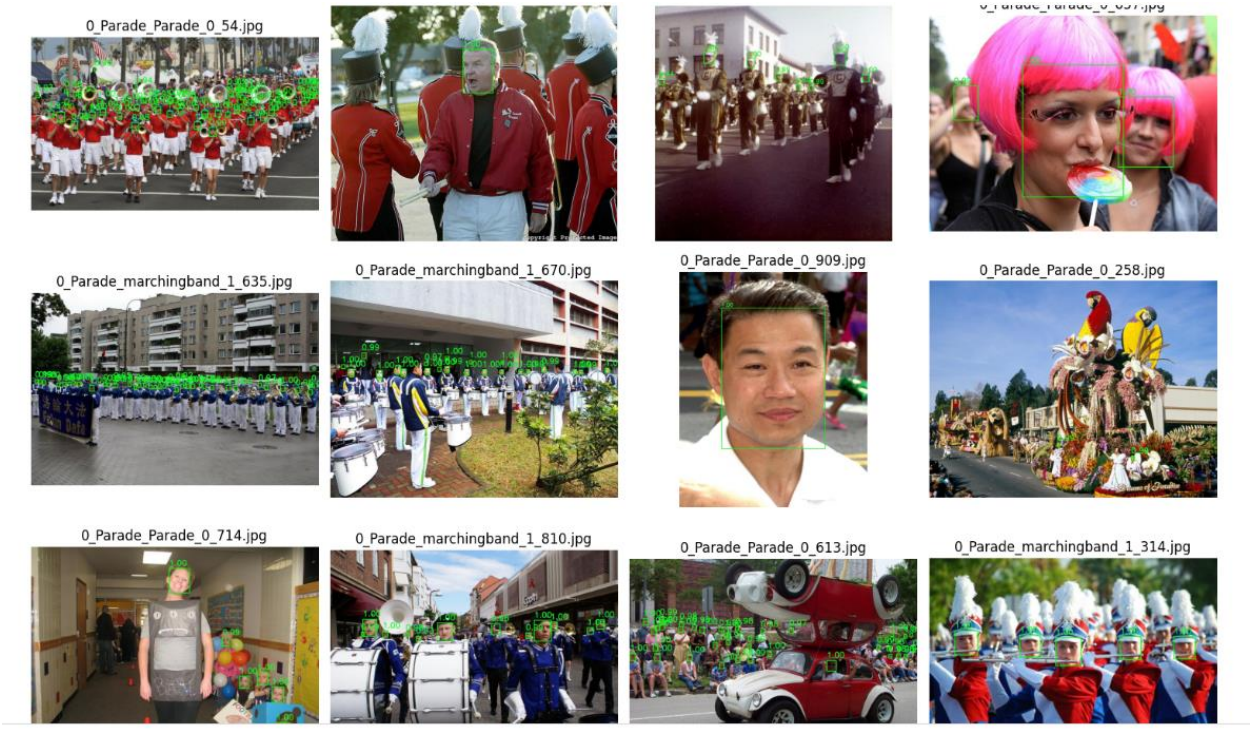
Models and Libraries:

We have used the following models and their accuracies are given. It is important to note that these are official results of model creators we have also tested models by ourselves to verify inference results. It is important to note that official results are more concerete.

Model	Easy	Medium	Hard	Link
RetinaFace	96.5,	96.5	90.4	Readme
SCRFD	95.40	94.01	82.80	Readme
YOLOv8n	94.5	92.2	79.0	Readme

Person Evaluation:

I have also tested these models on wider face dataset and these are their results. I have used a part of dataset just for understanding. The following image shows some random predicted samples from output folder by using widerface dataset and RetinaFace.



RetinaFace:

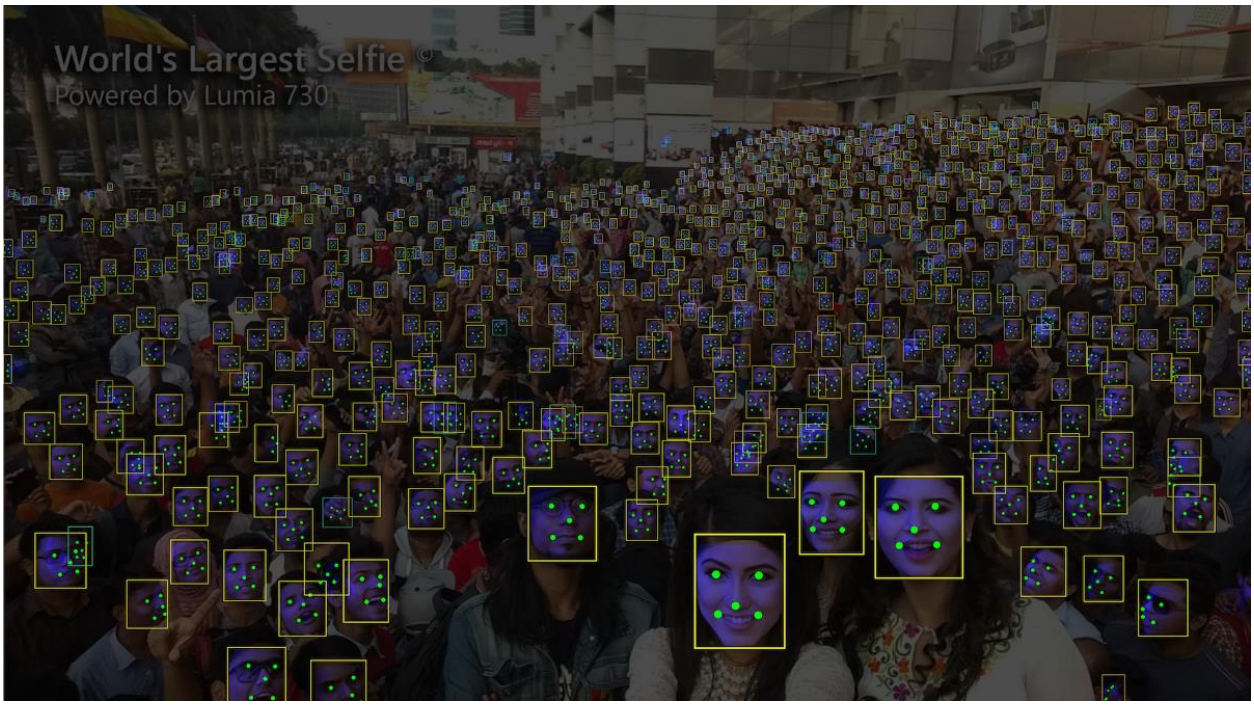
```
error = abs(total_faces - total_pred_faes)


accuracy = (1 - (error / total_faces)) * 100

print(f"Accuracy: {accuracy:.2f}%")
```

```
Accuracy: 93.81%
```

+ Code + Markdown

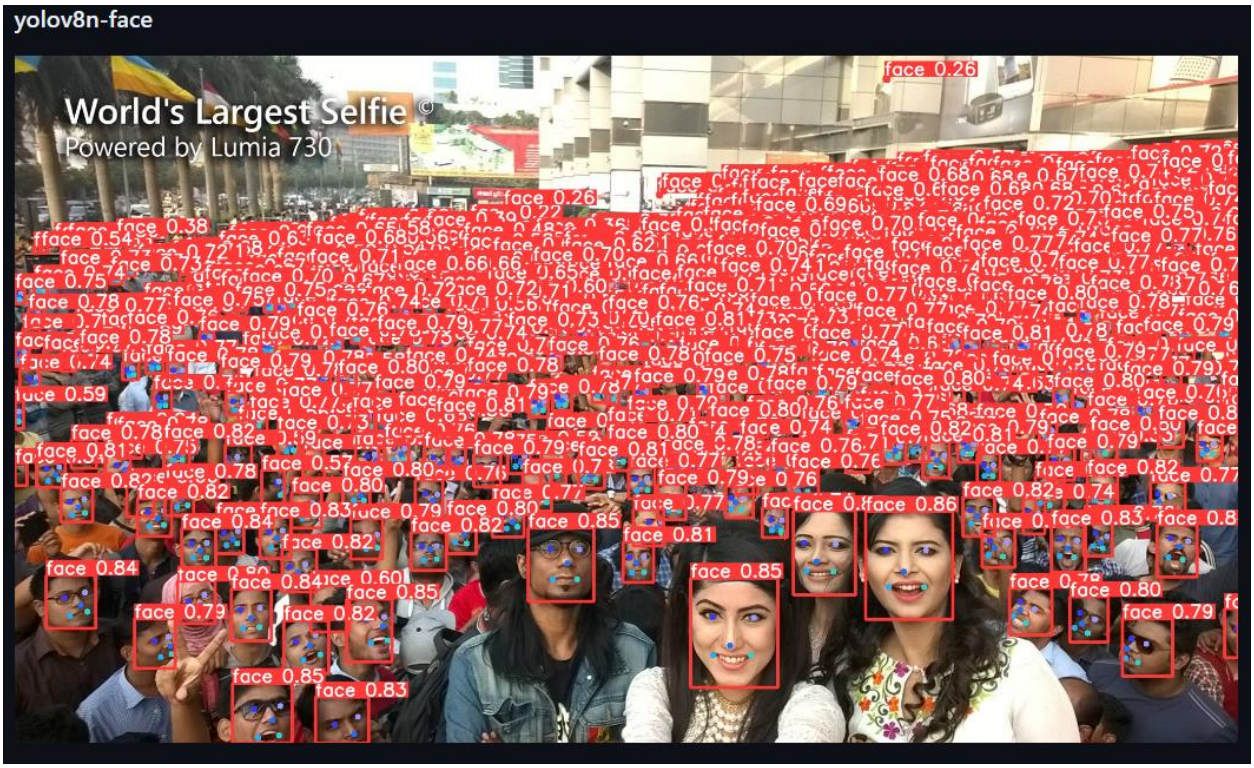


YOLOv8:

For yolo we have used nano v8 because larger models weights are not officialy available but we have used custom large model.

```
res=yolo.val(data="/kaggle/input/yoloannotations/data.yaml")
```

```
Ultralytics YOLOv8.2.78 🚀 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 268 layers, 43,607,379 parameters, 0 gradients, 164.8 GFLOPs
val: Scanning /kaggle/input/wider-face-yolo/valid/labels... 406 images, 0 backgrounds, 0 corrupt: 100%|██████████| 406/406 [00:00<00:00, 518.30it/s]
val: WARNING ⚠️ Cache directory /kaggle/input/wider-face-yolo/valid is not writeable, cache not saved.
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 26/26 [00:26<00:00,  1.03s/it]
                   all        406       7507       0.81      0.502      0.596      0.279
Speed: 0.7ms preprocess, 37.7ms inference, 0.0ms loss, 1.0ms postprocess per image
Results saved to runs/detect/val3
```



Code and Implementations:

I am giving the official code and my implementation results. I have also given the links to models above but here is further implementation.

Work	Demo	
Face Detecton	Video	CODE
Applying filter	Video	
Emotion based filter	Video (choppy due to memory fill)	
Model performance analysis	Nan	CODE

For the above application I have used two models **RetinaFace** and Har-Cascade.

- Used Har-Cascade due to small size

- Used RetinaFace for landmark detection

LANDMARK DETECTION:

For landmark detection we have used **RetinaFace** because it automatically outputs the landmrks along with face detection so we do not have to explicitly detect landmarks. The output format is as follows.


```
{
    "face_1": {
        "score": 0.9993440508842468,
        "facial_area": [155, 81, 434, 443],
        "landmarks": {
            "right_eye": [257.82974, 209.64787],
            "left_eye": [374.93427, 251.78687],
            "nose": [303.4773, 299.91144],
            "mouth_right": [228.37329, 338.73193],
            "mouth_left": [320.21982, 374.58798]
        }
```


You can view the landmarks in the code and above images shared.

Emotion Based filter:

- I have implemented a emotion based filter application using retinaface.
- It applies a mostache filter on smiling.
- The coordinates of facial land marks that we get from retinaface makes it much easier (not that much easy).
- We can use Nose, eyes and mouth coordinates (upper and lower lip) to detect change in emotions and apply filters respectively.
- Actually when person smiles his face width changes we are just calculating that change to apply filter.
- Although this is not best way I will discuss another method that I have found and we will use that which is much more efficient.

Landmark detection:

- As I have discussed above with the use of retinaface we will be able to detect several facial landmarks which will be helpful in further processing.
- We will be using another method for emotion based filter application.
- I have already shared the format of landmarks above.

Efficient way of Emotion based filter apllication:

- Another more efficient method that I was talking about for applying emotion based filter application is using a Neural network to detect emotions in each frame and they apply filters based on the output from the emotion predictor model.
- This method is way more efficient then the currently implemented method.
- One reason is that currently we are applying emotion based filters by guessing the change in coordinates of face which doesn't work always as it have flaws.
- Now I am working to train a custom emotion detector model that can detect the emotions and then we can apply filters based on that.
- I have also found a useful resource for that as well. Link

Conclusion:

So, till now I have finalized best models for all three tasks face detection, recognition and landmark detection. I have also implemented a emotion based filter application but it is not that much efficient. Also found another efficient method for doing that. So, I will try to implement it in next week (IA). Moreover the results of face recognition are not shown here but will share later on. Till now the best model according to above results and online available results is RetinaFace for face detection although **it is very heavy but fast models come at the cost of low accuracy.**

Link to InsightFace the parent organization of RetinaFace: Link