# Face Detection using YOLOv8

## 1. Introduction

Face detection is a core computer vision task that involves locating human faces in images or videos. It is widely used in applications such as biometrics, security, photo tagging, and human-computer interaction. In this project, I implemented a face detection system using YOLOv8, one of the most recent and efficient object detection models from Ultralytics.

The goal was to train a custom YOLOv8 model on a face detection dataset and evaluate its performance using standard object detection metrics.

## 2. Dataset

The dataset was obtained from Roboflow, a platform that provides annotated datasets for computer vision. It contained images annotated with bounding boxes around faces. The dataset was split into:

- Training set:
- Validation set:
- Test set:

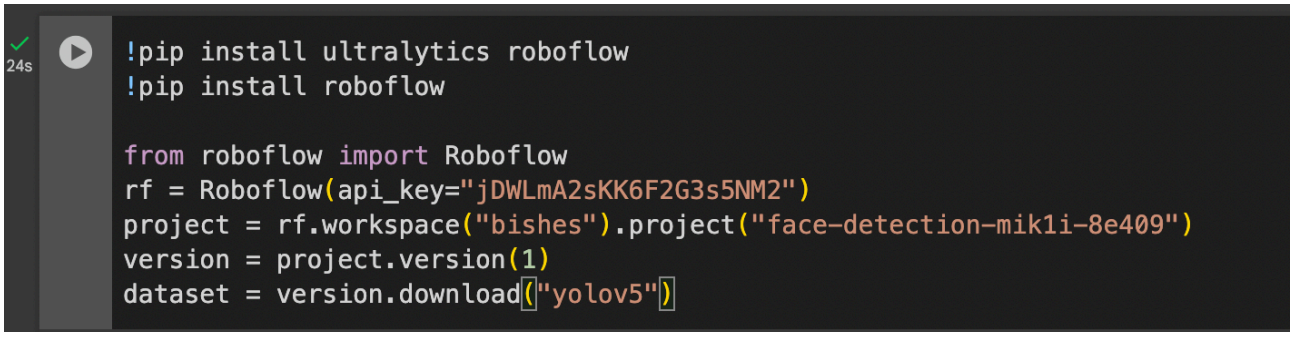
The images included various face orientations,
The link to the dataset: ***Dataset***

## 3. Model & Training Setup

I used the YOLOv8n (nano) model, which is the smallest and fastest YOLOv8 variant, suitable for quick experimentation and limited compute.

The training configuration was as follows:

- Epochs: 25
- Batch size: 16
- Image size: 640×640
- Optimizer & loss: Default YOLOv8 settings
- Environment: Python with the Ultralytics library



```
[6]  from ultralytics import YOLO


     model = YOLO('yolov8n.pt')

     results = model.train(
         data='Face-Detection-1/data.yaml',
         epochs=25,
         imgsz=640,
         batch=16,
         name='face-detection'
     )
```

## 4. Results

🔷 Training Results

The training process generated performance curves for precision, recall, and mean Average Precision (mAP).

- Precision: Measures how many predicted faces were correct.
- Recall: Measures how many actual faces were detected.
- mAP@0.5: Evaluates detection performance at an IoU threshold of 0.5.
- mAP@0.5:0.95: Evaluates across multiple IoU thresholds, giving a stricter measure.

Final results after training:

- Precision: 0.97
- Recall: 0.95.
- mAP@0.5: 0.98
- mAP@0.5:0.95: 0.75

| Class | Images | Instances | Box(P | R | mAP50 | mAP50-95): |
|---|---|---|---|---|---|---|
| all | 38 | 49 | 0.978 | 0.959 | 0.982 | 0.755 |

After the model is trained, the generated weights , best.pt, is used for validation and tests
The weights location: ***here***


🔷 Validation Results

The validation split was used to monitor generalization during training. YOLOv8 produced bounding box predictions on unseen images.

- Precision: 0.97
- Recall: 0.95
- mAP@0.5: 0.98
- mAP@0.5:0.95: 0.75

🔷 Test Results

The test split was used for final evaluation. The trained model achieved the following results:

- Precision: 0.92
- Recall: 0.905
- mAP@0.5: 0.963
- mAP@0.5:0.95: 0.68

***Note: face-detection, val 3, val 4 contains the results, evalutation metrics, images it trained was with and the batches for train, validation and test images***

5. Discussion

The model successfully learned to detect faces in a variety of conditions. From the metrics:

- A high precision indicates the model rarely predicts false positives.
- A high recall means it detects most of the actual faces.

- The mAP scores show how well the model balances precision and recall across different thresholds.

However, performance could be further improved. The model sometimes struggled with:

- Small faces in the background
- Faces under poor lighting
- Occluded or partially visible faces

6. Conclusion & Future Work

In this project, I trained and evaluated a YOLOv8-based face detection model using a dataset from Roboflow. The results demonstrate that YOLOv8 can effectively detect human faces with strong accuracy, even with a relatively small model (YOLOv8n).

Future improvements could include:

- Training for more epochs
- Using a larger YOLOv8 variant (s, m, or l) for higher accuracy
- Collecting more diverse training data
- Applying stronger data augmentation techniques

Overall, this project provided valuable experience in training object detection models and evaluating them with standard metrics.

The predicted results for code:

```
[11] prediction = model.predict(source="Face-Detection-1/test/images", save=True, conf=0.25)
```

Are in this folder : Here