**Project Report**

**Real-Time Object detection and tracking**

**Problem Statement:**

Develop a system capable of detecting and tracking specific objects in real-time within a video stream.

**Approach:**

Having worked on object detection problems before and seen the results of pre trained models (RCCN, fast RCCN and Faster RCCN and Yolo), the model of choice was Yolo. Although fast and faster RCCN models have an edge in performance, Yolo stands out on generalizability since it takes the whole context of the image.

**Implementation:**

The implementation of the model itself has been simplified for use. Since the actual implementation needs considerable time to yield useful results. For this implementation, I chose the Yolov8m You can use it on a dataset with Yolo based annotations or create your own self annotated data. I went with the option of self-annotated data. Data of images recorded with a front camera with different objects held and slight movements to add variation. The dataset itself is smaller to cater for time, but the results were promising. The annotations were done with labelImg software.

**The dataset:**

The dataset consisted of around 200 total images of which 27 were used for validation. The dataset had 5 classes Wajahat, painting, earbuds, hairbrush, keys. The dataset was made with images extracted from a video with the subject moving around objects.

**Object Detection and Tracking:**

The documented implementation allows for both object detection and tracking to be adjusted in a API call.

**Challenges Encountered:**

The choice of the model is one which asked for a judgment call. The judgment was between the time constraint and having to utilize the good that’s out there. After the choice, annotating data was another task, looking around for annotation techniques and coming to an effective one was small slope.

**Performance metrics:**

**Detection Accuracy:**

**A close-up of a white background

Description automatically generated**

**Confusion Matrix:  
A screenshot of a graph

Description automatically generated**

**Real-Time Processing Speed:**

The speed was observed for 3 different videos:

****

****

****

**Inference:**

There will be a pt file that serves as the inference center. Taking that file from the code base (running it in a colab environment), you can run the script:

Install Ultralytics:  
!pip install ultralytics

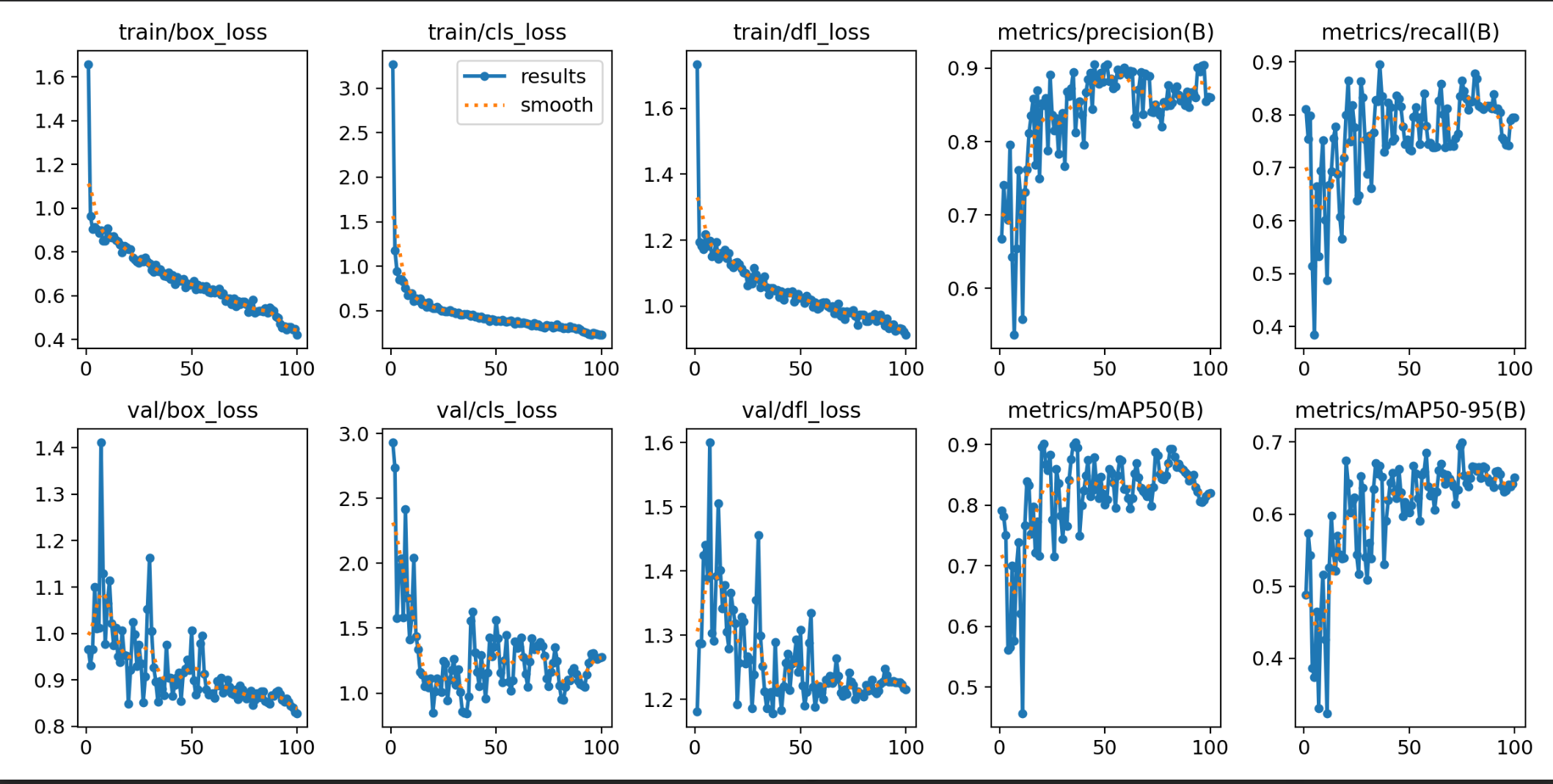
**Object Detection:**

Run the following Script:  
!yolo task=detect mode=predict model=custom\_model.pt show = True conf =0.5 source = building\_workers\_101.jpeg

**Object Tracking:**

In the source file, you can pass a particular video, just by its name.   
To run it on all images in some directory, we can just pass the directory in the source.  
For a live stream accessing a webcam we can set the source to zero.

**Results:**  
The results are stored in the runs/detect/predict (the number of the prediction) directory.

The runs/detect/train directory contains all the results of the training.   
**The loss values**:  


**The F1/Confidence Curve:**

**A graph of different colors

Description automatically generated**