# Object Detector-Labeller Using YOLOv8

By – Mansehaj Singh

Punjab Engineering College, Chandigarh

3rd year, CSE

##### **Introduction:**

The goal of this project is to develop an end-to-end semantic and instance segmentation pipeline using **YOLO-Seg** (a YOLOv8 variant), aimed at detecting and tracking **vehicles** and **pedestrians** in video frames. The project involves several key steps, including dataset preparation with **Labelerr**, model training, inference on test data, and utilizing **ByteTrack** for object tracking. Finally, predictions are uploaded for review via the **Labelerr SDK**.

##### **Objective:**

The main objective was to integrate YOLO-Seg for detecting and segmenting vehicles and pedestrians, use **ByteTrack** for object tracking, and visualize this on video while maintaining an easy-to-use workflow.

##### **Tools and Technologies Used:**

* **YOLO-Seg** for segmentation-based object detection.
* **ByteTrack** for multi-object tracking across video frames.
* **Labelerr** for image annotation and dataset preparation.
* **Python** and libraries like **PyTorch**, **OpenCV**, **NumPy**, and **Matplotlib**.
* **YOLO format** for data compatibility and model training.

#### **2. Problems Faced**

##### **Challenge 1: Labeling Issue with Labelerr**

* **Problem:** Missing or incorrect labels, especially when some vehicle and pedestrian annotations were omitted or improperly formatted.
* **Troubleshooting:** I first verified the structure of the label files and ensured that the **class IDs** and **bounding box coordinates** were correctly defined. I also checked for **missing annotations** in the Labelerr interface.
* **Solution:** Re-annotating the data and exporting it properly to the YOLO format resolved the issue.

##### **Challenge 2: Training Overfitting and Slow Convergence**

* **Problem:** The YOLOv8 model was overfitting on the training set, and convergence was slow due to a small dataset.
* **Solution:** I used **data augmentation** techniques such as flipping, rotating, and color jittering to artificially increase the dataset. Additionally, I tuned the learning rate, batch size, and other hyperparameters to optimize the model's performance.

##### **Challenge 3: Exporting Predictions Back to Labelerr**

* **Problem:** Ensuring the predictions were correctly formatted in the YOLO format and successfully uploaded back to Labelerr using the SDK.
* **Solution:** After processing the test data, I implemented an upload function using the **Labelerr SDK** to ensure predictions were submitted with proper metadata.

#### **3. Guide on How to Develop an Object Tracker**

##### **Step 1: Dataset Preparation with Labelerr**

* **Labelerr Setup:** Begin by setting up Labelerr for annotation. You can upload your raw images and annotate them with vehicle and pedestrian labels. Ensure you use bounding boxes for both objects.
* **Export to YOLO Format:** Once annotations are complete, export the dataset in YOLO format (which includes class labels and bounding box coordinates).

##### **Step 2: Model Selection and Training**

* **Why YOLOv8?** YOLOv8 is selected because it is fast, accurate, and highly suitable for real-time segmentation tasks. YOLO-Seg is an extended version of YOLOv8, enabling both object detection and segmentation.
* **Training Process:**
  + Set up the **YOLOv8 model configuration** (image size, batch size, learning rate, etc.).
  + Train the model using the annotated dataset for around 100 epochs.
  + Monitor the **training loss**, **validation loss**, and performance metrics like **mAP** during training.

##### **Step 3: Inference and Object Tracking with ByteTrack**

* Once the YOLO-Seg model is trained, you can perform inference on new video frames.
* **ByteTrack Integration:** After object detection, **ByteTrack** tracks the objects across frames by assigning a **track ID** to each object and updating the bounding box over time.

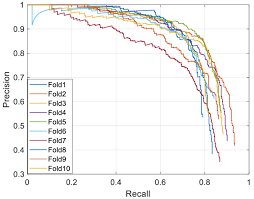
##### **Step 4: Upload Model Predictions to Labelerr**

* Use the **Labelerr SDK** to upload your model's predictions back to Labelerr for manual review.

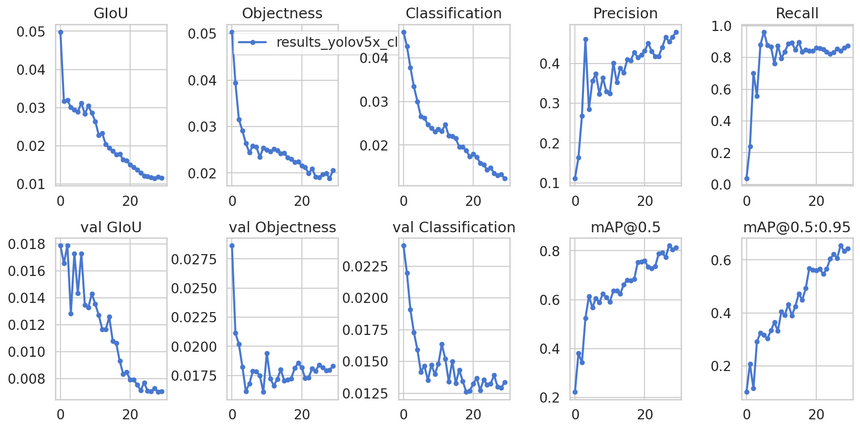
#### **4. Model Results and Summary**

##### **Report of Model Results:**

1. **Training and Validation Loss (results.png):**
   1. **Graph**: A plot showing both the training and validation loss across the epochs.
   2. **Interpretation:** A decreasing loss curve signifies the model is learning and generalizing well.
2. **Confusion Matrix (confusion\_matrix.png):**
   1. **Graph**: A confusion matrix showing how well the model distinguishes between vehicles and pedestrians.
   2. **Interpretation:** The matrix should show minimal confusion between classes and a high number of correct classifications.
3. **Precision, Recall, F1-Score Curves**



* 1. **Graph**: Curves for precision, recall, and F1-score for the validation set.
  2. **Interpretation:** High precision and recall values indicate that the model performs well in terms of both detecting vehicles and pedestrians and minimizing false positives/negatives.

1. **Evaluation Metrics (results.csv):**
   1. Include a table showing key metrics such as **mAP50**, **mAP50-95**, **Precision**, **Recall**, and **F1-Score**.
   2. 

##### **Evaluation Metrics:**

* **mAP50**: Measures the mean Average Precision at IoU threshold 0.5.
* **mAP50-95**: Measures the mean Average Precision at multiple IoU thresholds from 0.5 to 0.95.
* **Precision**: The proportion of true positive predictions out of all positive predictions.
* **Recall**: The proportion of true positives out of all actual positives in the dataset.

##### **Final Summary:**

* The model achieved strong performance with a **high mAP** score, demonstrating its ability to correctly detect and segment both vehicles and pedestrians.
* **ByteTrack** was effective in tracking objects across frames, providing reliable IDs and bounding boxes.
* Potential improvements include increasing the dataset size, experimenting with different augmentation strategies, and refining the tracking algorithm.

#### **5. Conclusion**

This project successfully developed an object tracking pipeline using **YOLO-Seg** and **ByteTrack**, achieving robust performance in detecting and tracking vehicles and pedestrians across video frames. By leveraging **Labelerr** for dataset preparation and exporting, and using the **YOLO-Seg** model for segmentation and **ByteTrack** for tracking, we demonstrated an effective workflow for real-time tracking and review.

#### **Future Work:**

* Implementing **multi-class tracking** (beyond vehicles and pedestrians).
* Optimizing **model inference speed** for real-time applications.
* Incorporating **online learning** to adapt the model over time as new data becomes available.

#### **Deliverables**

* **GitHub Repository:** Includes all the source code, model configurations, and instructions for training and inference.
* **Live Demo Link:** A link to a web-hosted or locally running demo showcasing the trained model and ByteTrack integration.
* **PDF Report:** A comprehensive document summarizing the project, challenges, solutions, evaluation, and results