

Resilient Multi-Model Detection Under Sensor Failure (REU)

¹Aavash Kharel, ¹Raul Shakya, ¹Debashri Roy

¹The University of Texas at Arlington

OBJECTIVE

- **Primary Goal:** To train and deploy a YOLOv8 object detection model capable of accurately identifying custom target classes (X1, X2, X3, X5) in video footage.
- **Pipeline Automation:** Develop a streamlined Python preprocessing pipeline to instantly convert complex Scalabel JSON annotations into normalized YOLO coordinate formats.
- **Robust Evaluation:** Perform an automated, randomized 80/20 train/validation split across 11,000+ frames to ensure unbiased accuracy testing and prevent model overfitting.
- **Real-World Testing:** Deploy the final trained weights (best.pt) on unseen video footage to verify the model can accurately draw bounding boxes and track the subjects in practice.
- **Hardware Optimization:** Efficiently process the massive 11,000-frame dataset on a dedicated GPU by downscaling the inputs to various dimensions, preventing memory crashes while maintaining accuracy.

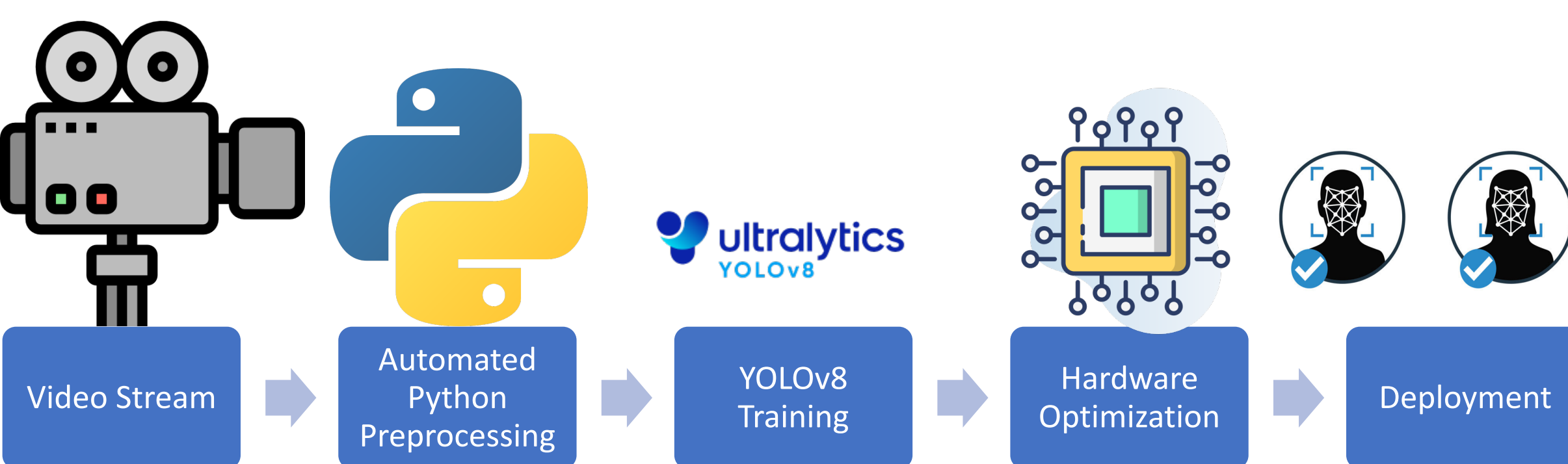


Fig: Schematic representation of types of robotics communication

BACKGROUND

- High-definition video analysis is increasingly critical for tracking custom subjects across large-scale datasets using deep learning models like YOLOv8.
- Existing pre-trained models excel at detecting generic objects but fail on custom, lab-specific targets (Classes X1, X2, X3, X5) without dedicated retraining on newly annotated data.
- Scaling up this training is difficult: manual annotation tools (like Scalabel) output absolute pixel coordinates which YOLO cannot read, and feeding full-resolution frames directly into a neural network triggers severe GPU Out-Of-Memory (OOM) crashes.
- This project fills that gap by developing an automated preprocessing pipeline that standardizes 11,000 frames into normalized coordinates, allowing for efficient, scaled-down GPU training without losing spatial accuracy.

SETUP

- **Hardware Environment:** Dedicated Ubuntu workstation equipped with an NVIDIA GPU to handle hardware-accelerated deep learning workloads.
- **Software Stack:** Python, PyTorch (CUDA-enabled), and the Ultralytics YOLOv8 Nano (yolov8n.pt) base model, (yolov8m.pt), and (yolov8l.pt).
- **Dataset:** 11,000 individual frames extracted from native test video. Ground truth bounding boxes were manually drawn using the Scalabel annotation tool.
- **Target Classes:** Four specific custom subject classifications tracked in the dataset: X1, X2, X3, and X5.

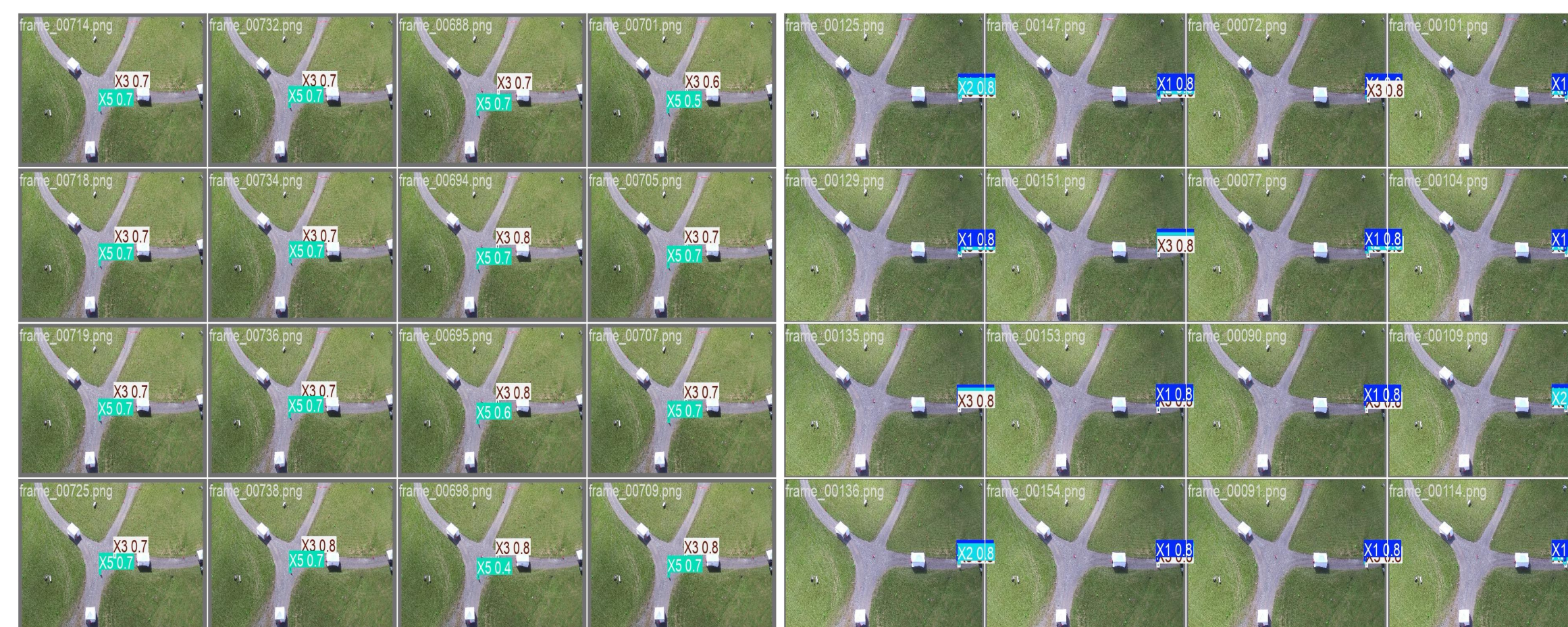
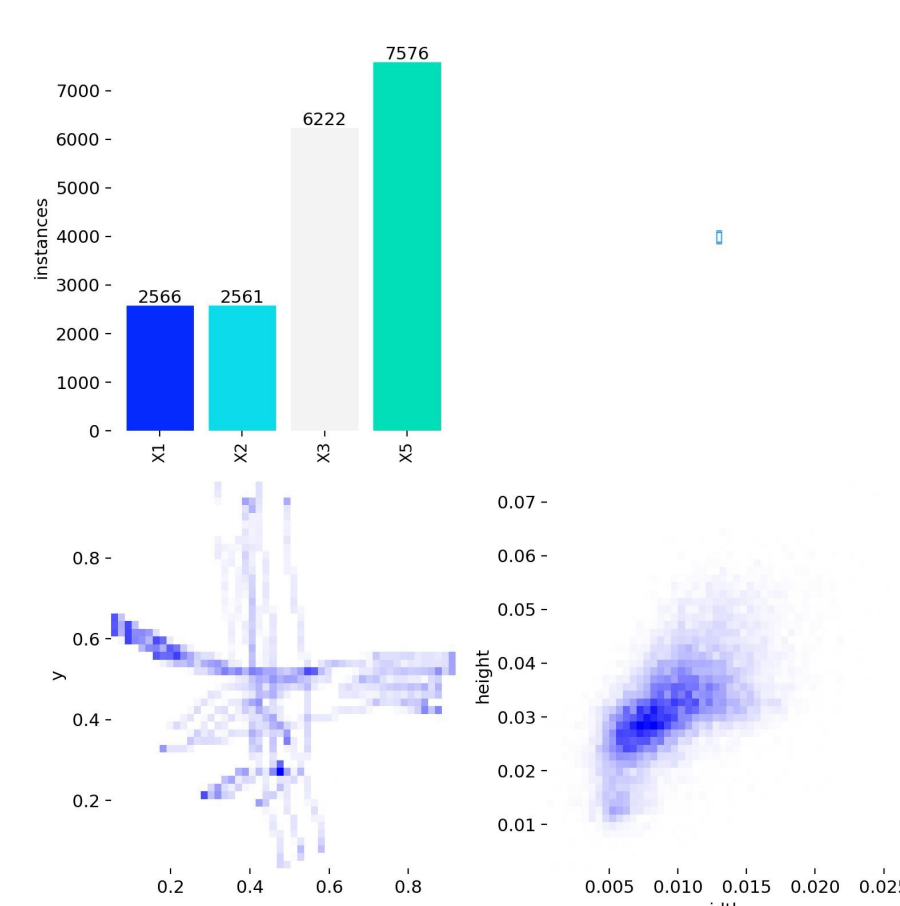


Fig: value batch for train 6

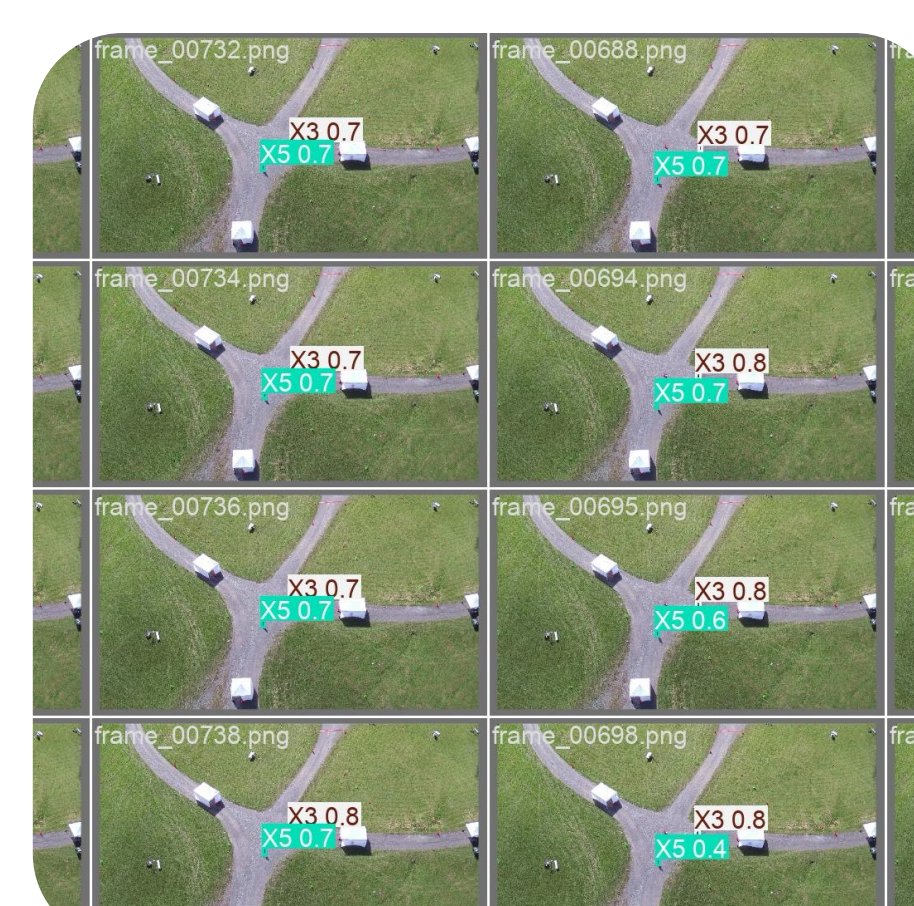
Fig: value batch for train 10

TEST PLAN

- **Data Parsing:** Developed a Python script to parse the exported Scalabel JS file and extract the raw bounding box coordinates for all 11,000 labeled frames.
- **Math & Normalization:** Converted absolute pixel values into YOLO's required 0.0–1.0 format by dividing the X and Y coordinates by the native image width and height.
- **Dataset Splitting:** Implemented an automated randomizer to shuffle the 11,000 frames, isolating 80% into a train directory and 20% into a val (validation) directory to ensure unbiased testing.
- **Model Training:** Executed the YOLOv8 training sequence for 100 epochs. Configured a batch size of various numbers (8,16,32) and downscaled the input images to different `imgsz` to optimize GPU VRAM usage and prevent memory crashes.



Train 6 Labels Graph



Train 6 Labels Grid

RESULTS

The evaluation compared two major training iterations: Train 6 (baseline) and Train 10 (optimized) to determine the optimal balance between training time and detection accuracy.

Performance Metrics:

Metric	Train 6 (baseline)	Train 10 (optimized)	Improvement
Epochs	200	250	+25%
Precision	88.6%	96.6%	+8.0%
Recall	81.6%	96.3%	+14.7%
mAP ₅₀	87.1%	97.3%	+10.2%
mAP ₅₀₋₉₅	40.1%	52.5%	+12.4%
Training Time	11.2 hrs	50.0 hrs	-

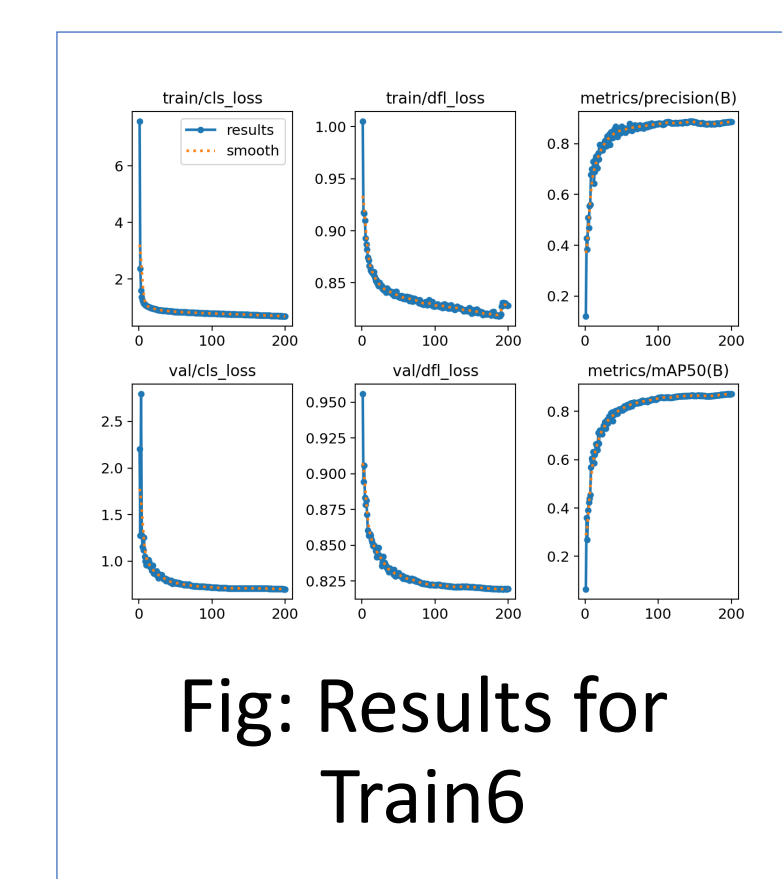
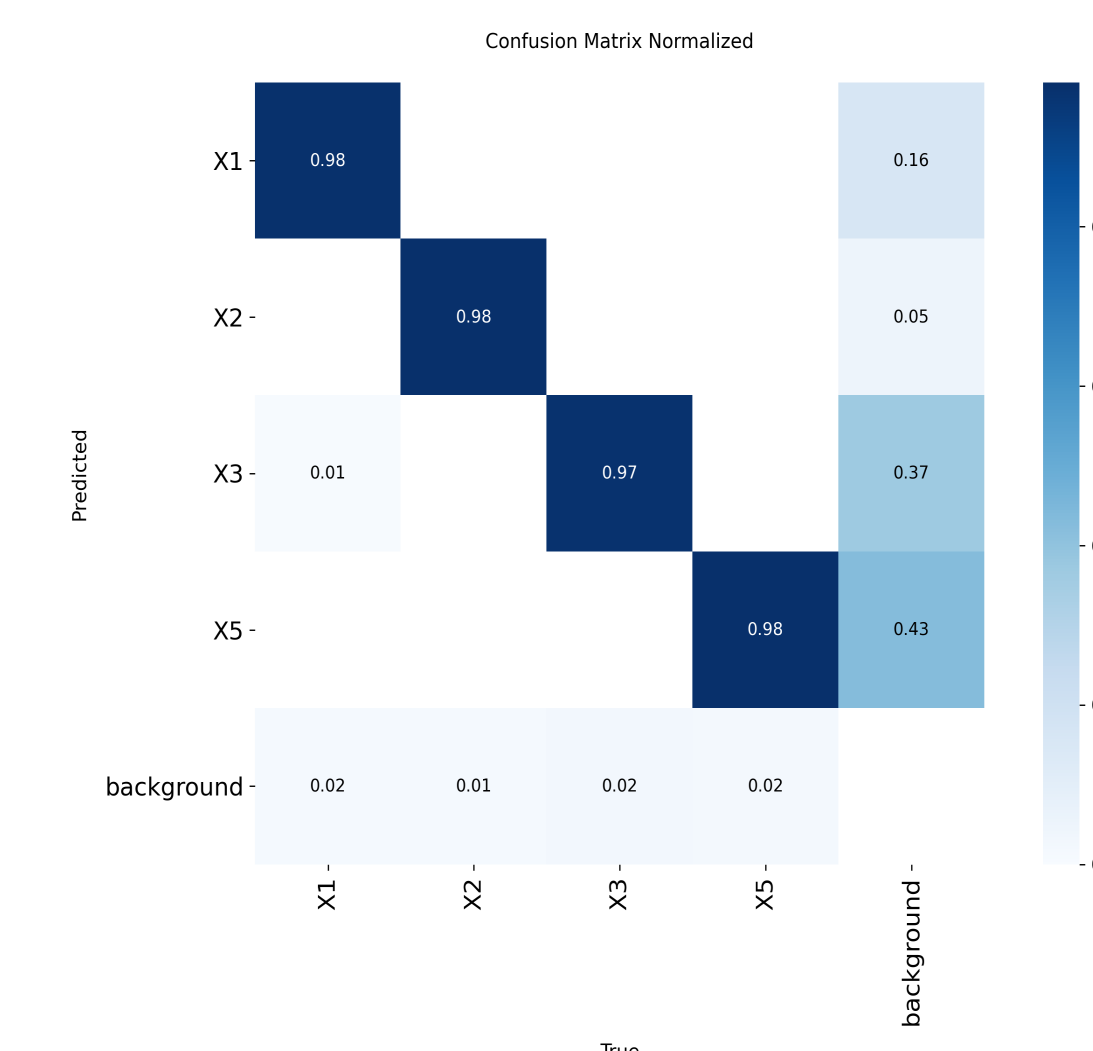


Fig: Results for Train6

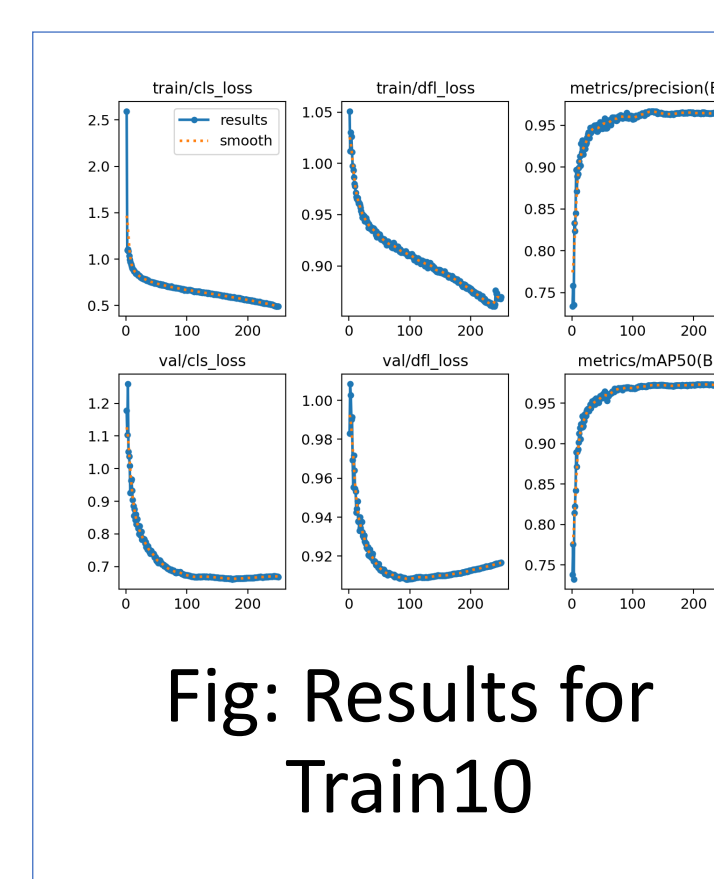


Fig: Results for Train10

Fig: Normalized confusion matrix for Train10

CONCLUSIONS

- The custom Python suite successfully automated the conversion of complex Scalabel JSON annotations into normalized YOLO formats, enabling seamless **preprocessing of 11,000+ frames**.
- Through an automated 80/20 train/validation split, the model achieved tight convergence. The minimal variance between training and validation loss curves confirms that the model successfully **generalized the features of classes X1, X2, X3 and X5 without overfitting**.
- Extending the training to 250 epochs and optimizing hyperparameters resulted in a **97.3% mAP50**, a significant **10.2% improvement** over the baseline.
- Deployment of the best.pt weights on unseen footage demonstrated **96.6% precision rate**. This ensures that the detection system is robust enough for real-time monitoring and resilient against potential sensor inaccuracies.

CONTACTS

Aavash Kharel: axk8168@mavs.uta.edu
 Raul Shakya, : rxs6339@mavs.uta.edu
 Debashri Roy: debashri.roy@uta.edu