Computer Vision Project Presentation

Title:Traffic Sign Detection using YOLO

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

Group Members:
Aarya Gupta
Abdullah Shujat
Sargun Singh Khurrana

Problem Statement and Scope

Problem Statement :

Develop a real-time Traffic Sign Detection system using YOLOv8 to enhance road safety and support autonomous navigation. The system aims to accurately detect and recognize traffic signs in images and videos using deep learning.

Scope:

Input: Traffic sign images & video streams from dashcams, surveillance cameras, or vehicle sensors.

Dataset: 4,969 samples, split into Train, Validation, and Test sets.

Output: Annotated images/video frames with bounding boxes, labels, and confidence scores.

Target Users:

Autonomous Vehicles & ADAS Systems – For real-time sign recognition, ensuring compliance with traffic rules.

Traffic Monitoring Authorities – To track violations using CCTV & IoT-based monitoring.

Urban Planners & Smart City Developers – To analyze traffic compliance and improve infrastructure planning.

Related Work and Baseline Methods

1. **SVM with Histogram of Oriented Gradients (HOG):** Uses **HOG** to extract shape & edge-based features. Classified using **Support Vector Machine (SVM)** for robust detection.
- **Reference**: Nabil Ahmed et al. – High-precision SVM-HOG model for Bangladeshi traffic signs.

2. **Convolutional Neural Networks (CNNs):** Learns hierarchical features directly from images. Effective for **real-time** traffic sign recognition.
- **Reference**: Deepali Patil et al. – CNN-based system that captures and processes traffic signs in video.

3. **Hybrid GIS & Machine Learning Approaches:** Combines **geo-tagged images** with ML for sign detection & positioning.
- **Reference**: Zihao Wu – Uses Google Street View & GoPro for mapping traffic signs.

Related Work and Baseline Methods

Baseline Models for Comparison

To evaluate **YOLO v8**, we compare against:

- **SVM + HOG:** Traditional machine learning benchmark for traffic sign detection.
- **CNN-based detection:** Standard deep learning approach widely used in image recognition.

Identified Gaps in Existing Approaches

- **SVM + HOG:** Limited generalization under varying lighting conditions and occlusions.
- **CNN-based Models:** High accuracy but computationally expensive, making them less suitable for real-time applications.
- **Hybrid GIS Models:** Require extensive **geo-tagged data**, which limits scalability and adaptability to new environments.

Datasets and Evaluation Metrics

Dataset Used: Traffic Sign Detection Dataset:

- Contains images of traffic signs and surrounding environments. Used for initial model training and validation of YOLOv8. Helps assess preliminary detection performance.

Evaluation Metrics:

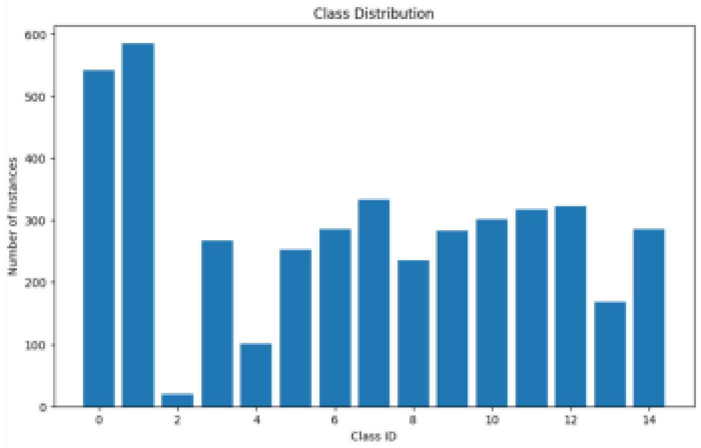
Precision: 94.2% overall

- **High:** Stop Signs (**97.3%**), Speed Limit Signs (**92-100%**)
- **Lower:** Red Light (**88.1%**), Green Light (**82.6%**)

Recall: 90.6% overall

- **High:** Stop Signs (**98.8%**), Speed Limits (**94.1-99.1%**)
- **Lower:** Red Light (**69.4%**), Green Light (**74.1%**)

- **mAP@0.5: 95.7%** – Strong detection performance
- **mAP@0.5-0.95: 82.97%** – Highlights challenges in precise localization



Name of Classes: Green Light, Red Light, Speed Limit 10, Speed Limit 100, Speed Limit 110, Speed Limit 120, Speed Limit 20, Speed Limit 30, Speed Limit 40, Speed Limit 50, Speed Limit 60, Speed Limit 70, Speed Limit 80, Speed Limit 90, Stop

System, Baseline results and Analyses (1/3)

System Overview

- **Architecture:** Utilizes YOLOv8 for real-time traffic sign detection.
- **Input:** Traffic images and video streams from dashcams and CCTV.
- **Output:** Annotated images with bounding boxes, labels, and confidence scores.

Processing Pipeline:

1. **Preprocessing:** Resizing, normalization, and augmentation.

2. **Model Inference:** YOLOv8 detects and classifies traffic signs.

3. **Output Generation:** Labeled images/videos with predictions.

System, Baseline results and Analyses (2/3)

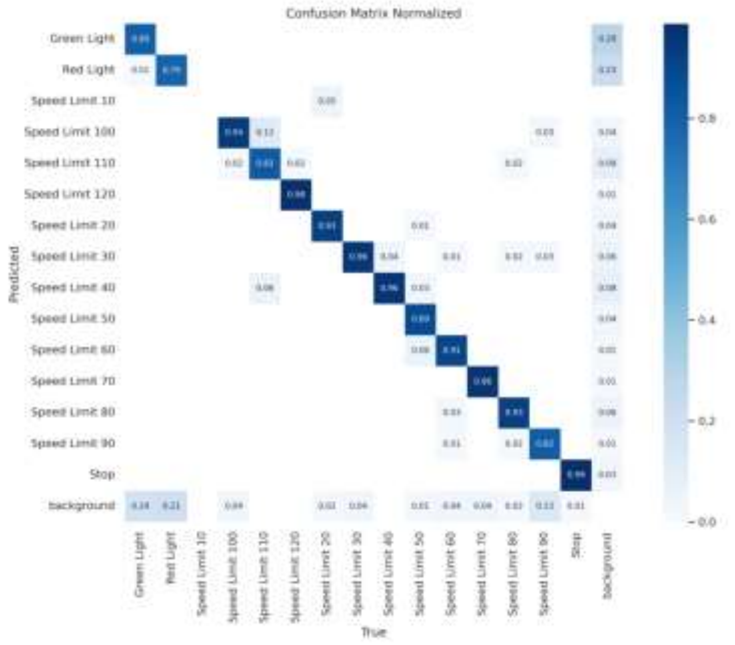
Performance Across Different Traffic Signs

Traffic Sign	Precision (%)	Recall (%)	Observations
Stop Sign	**97.3**	**98.8**	High accuracy due to distinct shape & color
Speed Limit	**92-100**	**94.1-99.1**	**High precision** but minor misclassification in similar speed categories.
Red Light	**88.1**	**69.4**	Missed detections in glare/reflection conditions
Green Light	**82.6**	**74.1**	Errors where trees are nearby

System, Baseline results and Analyses (3/3)

Common Failure Cases & Root Causes

1. **False Negatives (Missed Detections)**
   - Cause**:** Low contrast or occlusions in traffic scenes.
   - Example: Green lights misclassified when blended with surroundings.
2. **False Positives (Incorrect Detections)**
   - Cause: Similar-looking objects (billboards, advertisements).
   - Example: Misidentifying commercial signs as traffic signs.
3. **Overlapping Signs Confusion**
   - Cause: Multiple signs detected in close proximity.
   - Example**:** Stop sign partially hidden behind another object.
4. **Glare & Motion Blur Impact**
   - Cause: Sunlight reflections and high-speed vehicle movement.
   - Example: Incorrect bounding boxes on blurred traffic signs.



Peculiarities/Challenges in Dataset:

1. **Class Imbalance:** Some classes, such as stop signs and speed limits, are well-represented, whereas traffic light signs (Red Light, Green Light) have relatively fewer instances.
2. The dataset contains images captured in **different lighting conditions** (day, night, fog), which affects detection performance
3. Some images have **low resolution**,have **motion blur** or are **partially occluded,** making recognition more challenging

Next Steps (Future work)

Model Optimization: Fine-tune YOLOv8 with better hyperparameters for higher accuracy

Incorporate additional datasets: to enhance model generalization.

eg- GTSRB

Data Augmentation: Expand dataset coverage, particularly for Red and Green Lights, using advanced augmentation techniques.

User Interface Development: Integrate UI and real-time detection for a smoother experience.

Performance Testing: Conduct real-world evaluations for inference speed, accuracy, and resource efficiency.

Final Documentation: Compile results, methodologies, and findings into a comprehensive final report

Contributions

Aarya Gupta: Model Implementation, Training, EDA, Performance Evaluation, Report Writing

Abdullah Shujat: Dataset Collection, Data Augmentation, Report Writing, Literature Review, Presentation Preparation

Sargun Singh Khurana: Research and Testing, Data Preprocessing, Literature Review, System Debugging, Dataset Collection, Report Writing, Presentation Preparation.

Future Task Assignments

- **Aarya Gupta** → Research, Training model
- **Abdullah Shujat** → UI implementation, performance testing.
- **Sargun Singh Khurana** → UI improvement, Model fine-tuning,

End-Main Slide Deck

Feedback from Review Meeting

- **GIS-Based Mapping for Traffic Signs**
  - Use a smartphone app to capture traffic signs with GPS location.
  - Real-time registration of detected signs onto maps for better tracking.
  - Can be used for road infrastructure monitoring and urban planning.

- **Quantitative Error Analysis apart from existing evaluation metrics**
  - Deep dive into failure cases: where and why the model misclassified.
  - Conduct a detailed breakdown of poor-performing classes.
  - Improve model calibration and threshold tuning.

- **Real-Time Integration on a Small Database**
  - Test YOLOv8 on a limited real-time dataset for faster iteration.
  - Check latency, FPS, and performance constraints on real-world devices.
  - Optimize model size for mobile/edge deployment.

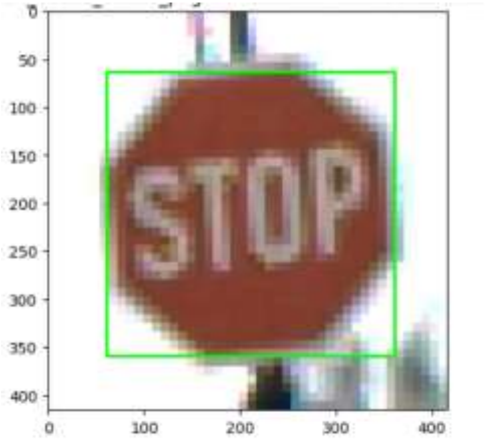
- **Scalability and Use Case Refinement**
  - Understand how the solution scales in different environments.
  - Evaluate crowdsourcing potential: users contributing traffic sign data.
  - Improve robustness by adding statistical filtering for noisy detections.

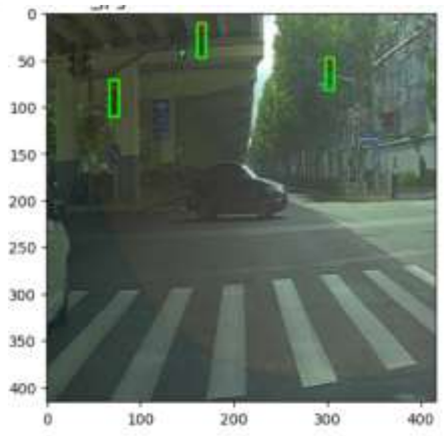
Next Steps

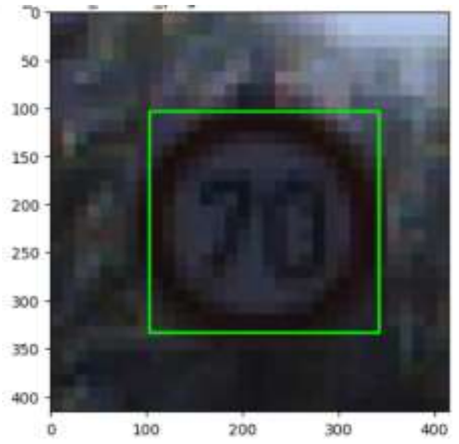
1. Implement GPS-based tracking and integrate with a mapping tool.
2. Perform detailed error analysis on misclassified cases.
3. Deploy a real-time prototype for small-scale testing.
4. Explore crowdsourcing options to expand the dataset.

BACKUP
SLIDES

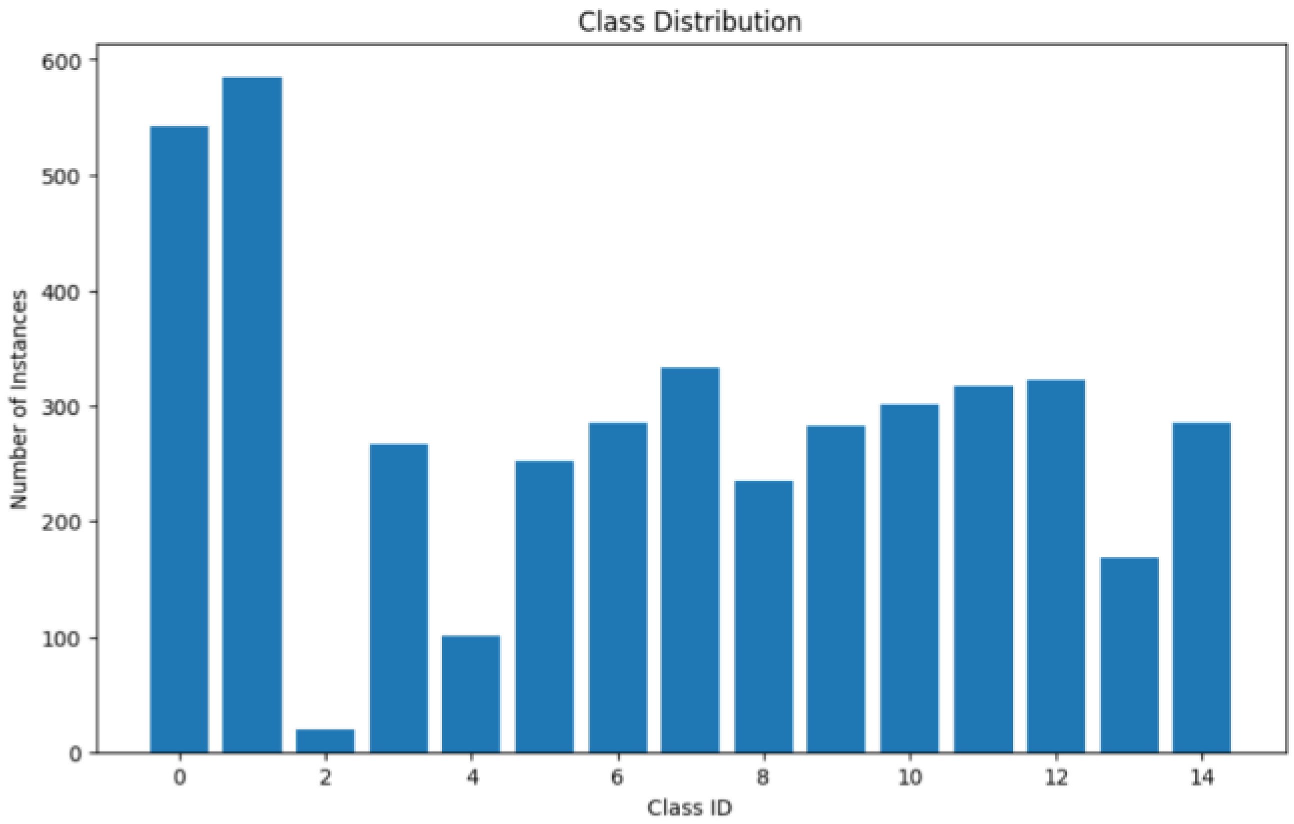
Sample Images



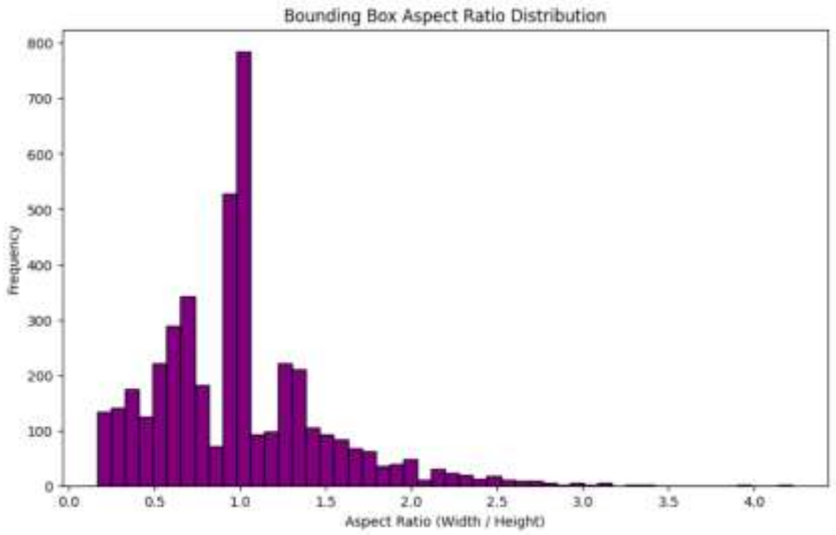


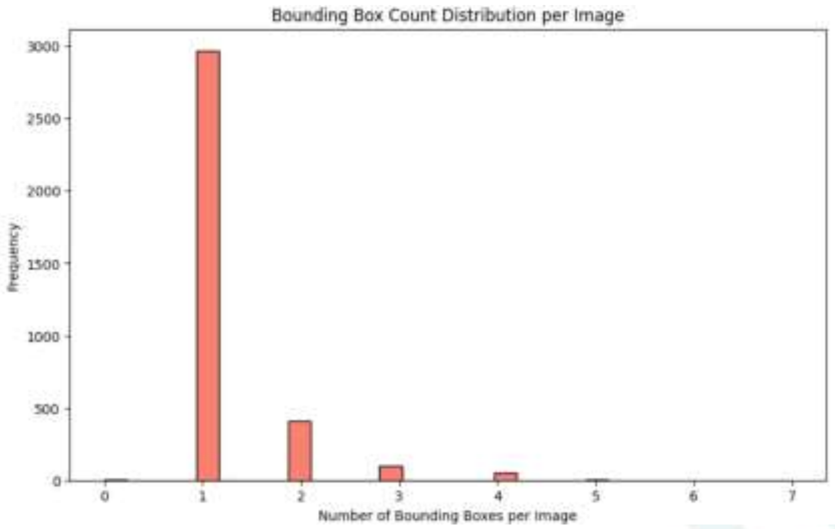


EDA - Datasets

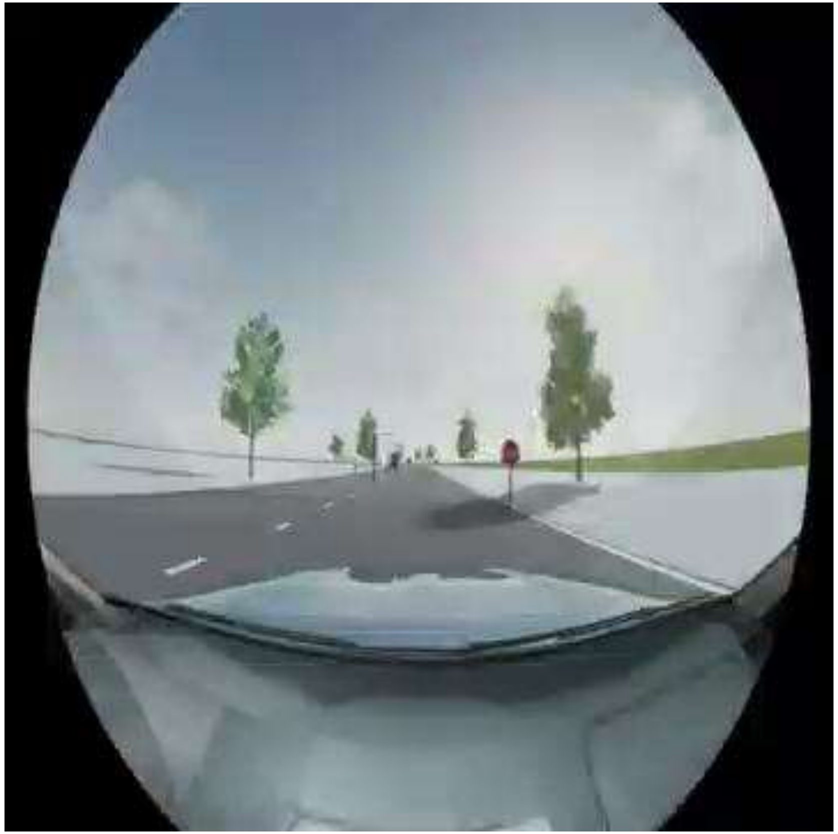


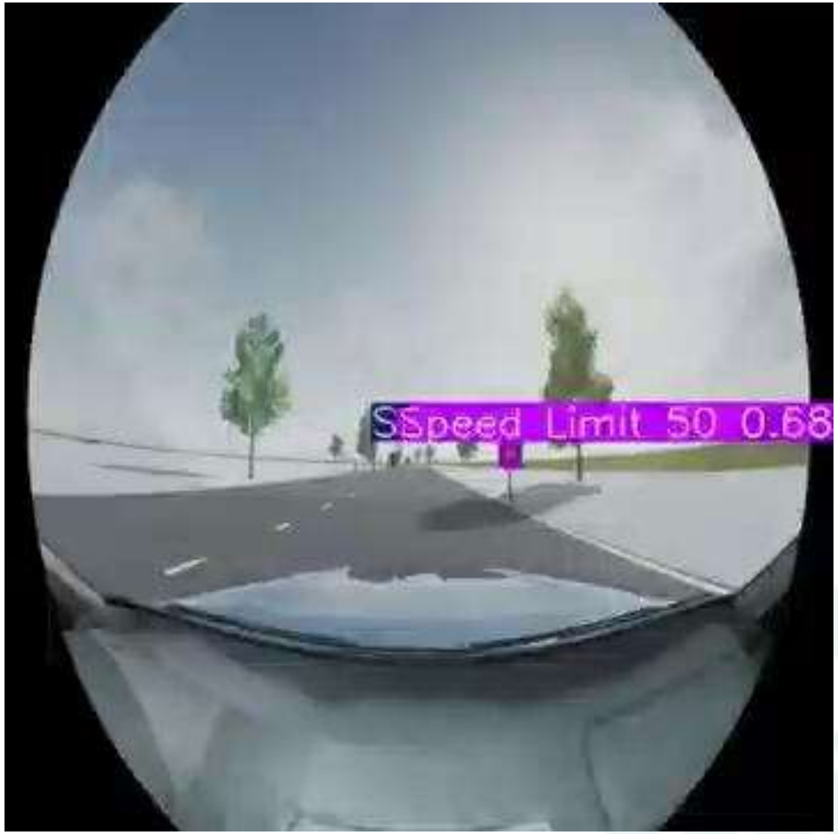
EDA - Datasets

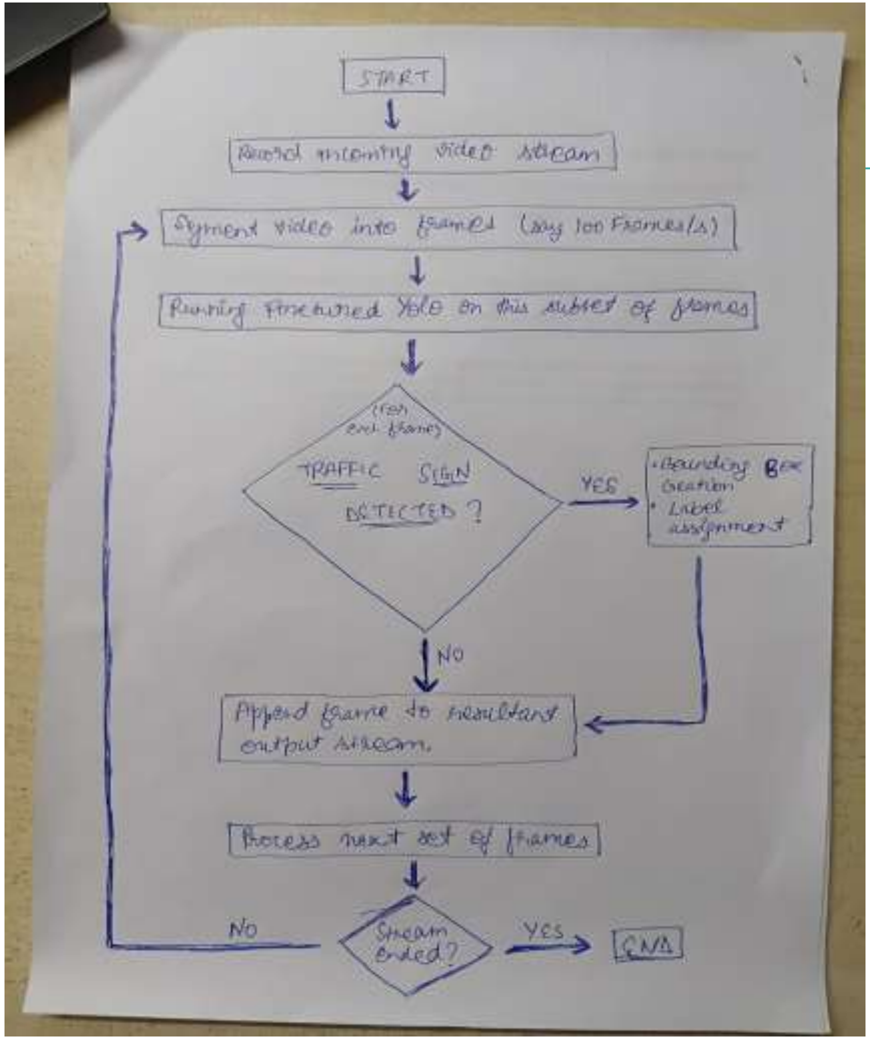




Input Test Video and Patched Result Video






START
Record incoming video stream
Segment video into frames (say 100 Frames/s)
Running Finetuned Yolo on this subset of frames
(for each frame)
TRAFFIC SIGN DETECTED ?
YES
• Bounding Box Creation
• Label assignment
NO
Append frame to resultant output stream.
Process next set of frames
NO
Stream ended?
Yes
END

Proposal: Pipeline for Real-Time Stream Processing