

AI-Powered License Plate Detection System

*Course: ITAI 1378 – Computer Vision and
AI*

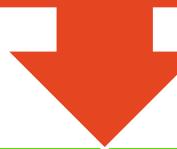
Student: Jeremy Okyere

*Tier: Tier 2 (Project with custom dataset
and model fine-tuning)*

The Problem

Real-World Issue:

Law enforcement officers spend hours manually reviewing footage to identify vehicles linked to crimes or terror-related activities.



Who Cares:

Law enforcement
agencies

Security and border
control organizations

Smart city traffic
management teams



Why It Matters:

Faster and more reliable license-plate detection allows authorities to prevent attacks, locate suspects quickly, and strengthen public safety.

Proposed Solution

Goal:

- Develop an **AI-based license-plate detection system** capable of identifying and localizing plates from images and videos.

How It Works:

- Camera → Image Input → YOLOv8 Model → Detected Plate → Output (Bounding Box / Cropped Plate)

Technical Approach

| Component | Details |
|------------------|---|
| CV Technique | Object Detection |
| Model | YOLOv8 (SOTA real-time detector) |
| Framework | TensorFlow + Ultralytics |
| Why YOLOv8? | Combines speed and accuracy — ideal for real-time surveillance and mobile deployment |
| Development Env. | Google Colab + Python 3.10 + OpenCV + Matplotlib |
| Colab Link | https://colab.research.google.com/drive/1a2aEJMculwLyhtg0kplgxojlu5RtDsIO |

Data Plan



Dataset Source:

Kaggle “License Plate Detection Dataset”
Supplement with web-scraped vehicle images for diversity



Size & Type:

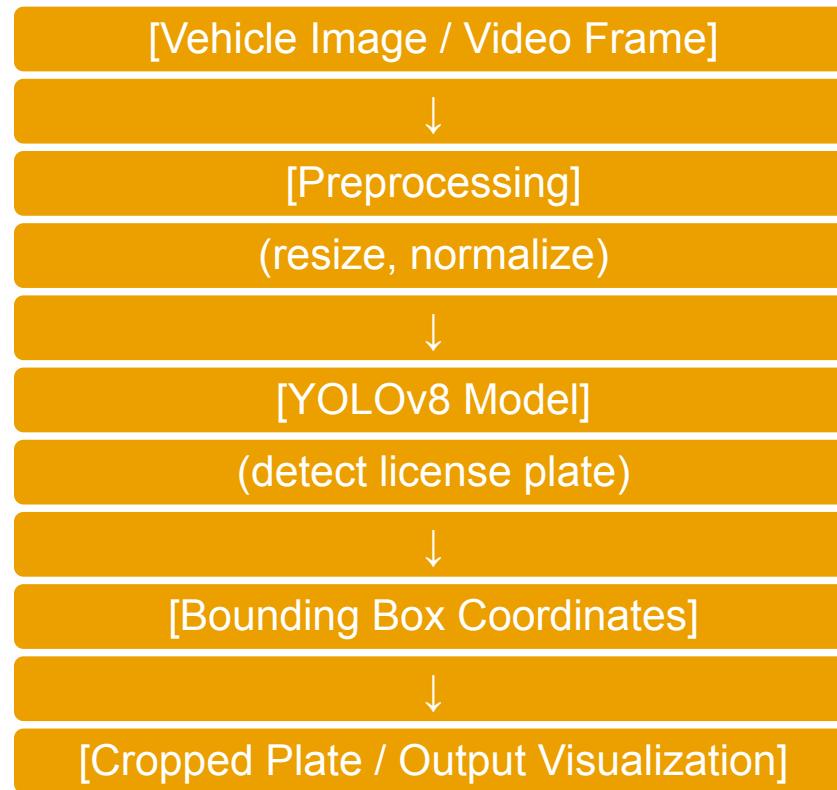
~500 images of vehicles (urban & highway scenes)
YOLO-formatted labels with bounding boxes



Preparation Steps:

Clean and resize images to 640×640 px
Augment dataset (rotation, brightness, blur) via Roboflow
Split into train / validation / test sets (70 / 20 / 10)

System Architecture



Success Metrics

| Metric | Target | Purpose |
|------------------------|----------------------|--|
| mAP@0.5 | $\geq 90\%$ | Measure detection accuracy |
| FPS | ≥ 20 frames/sec | Ensure real-time processing |
| False Positives | $\leq 5\%$ | Maintain reliability in diverse lighting |
| Latency | ≤ 1 s/image | Guarantee fast response for video feeds |

Week-by-Week Plan

| Week | Task | Milestone |
|---------|---|-------------------|
| Week 10 | Gather dataset, set up Colab | Dataset ready |
| Week 11 | Train YOLOv8 model | Model initialized |
| Week 12 | Evaluate and tune parameters | Accuracy >85% |
| Week 13 | Integrate into demo app (streamlit or video) | Demo functional |
| Week 14 | Final testing and polish slides | Model finalized |
| Week 15 | Present | Project complete |

Challenges & Backup Plans

| Challenge | Impact | Mitigation Plan |
|---------------------------------|-----------------|--|
| Limited data diversity | Lower accuracy | Use Roboflow augmentation / collect web data |
| GPU limitations on Colab | Slower training | Use Kaggle Notebook / Heidi server |
| Low performance in night scenes | Reduced recall | Add low-light images / adjust brightness in training |

Resources Needed

Compute: Google Colab
(GPU T4 / A100)

Libraries: TensorFlow,
Ultralytics, OpenCV, Matplotlib

Data Tools: Kaggle, Roboflow

Cost: \$0 (fully open-source)

Estimated Training Time: ~3
hours per epoch on Colab GPU