LICENSE PLATE DETECTOR

Computer Vision Capstone Project - ITAI 1378

Student: SM Tawhid
Email: w211161582@student.hccs.edu

Tier: Tier 1
Tools: Google Colab, YOLOv8, Roboflow

PROBLEM STATEMENT

- Manual identification of license plates is slow and inefficient.
- Traffic enforcement, parking systems, and security cameras need automated detection.
- Computer vision offers a fast and reliable way to localize license plates.

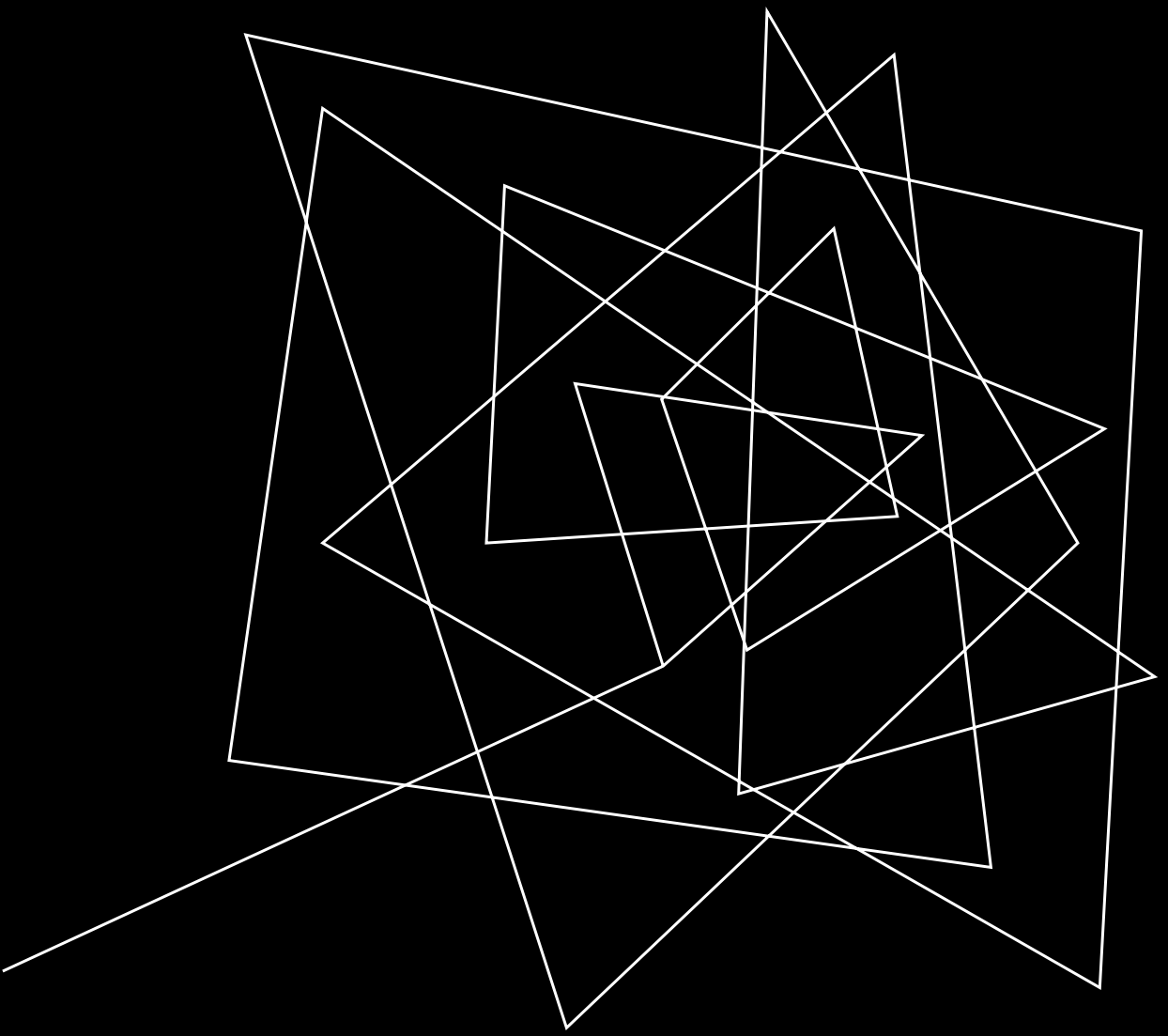
PROJECT GOAL

Train a YOLOv8 object detection model to accurately detect license plates in real-world images.
The model should be fast, lightweight, and able to run in Google Colab with limited compute.

DATASET OVERVIEW

Dataset: License Plate Recognition (Roboflow Universe)

- ~10,125 labeled images
- Label: license_plate
- Mixture of cars, angles, lighting conditions
- Augmentations included in Roboflow version



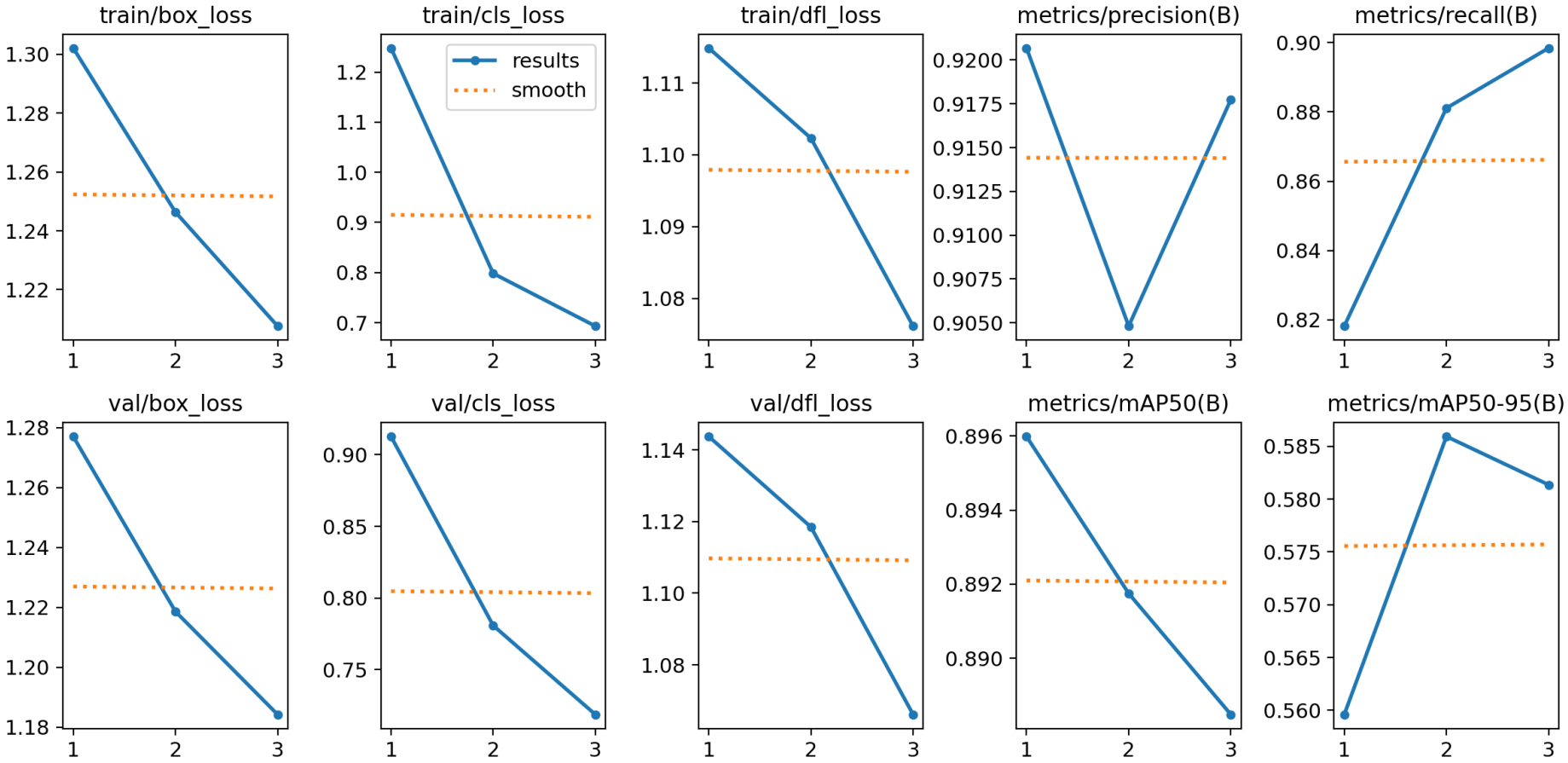
TECHNICAL APPROACH

- Technique: Object Detection
- Model: YOLOv8n (nano version for faster training)
- Framework: PyTorch (via Ultralytics)
- Why YOLOv8n:
  - Fastest YOLO version
  - Lightweight
  - Works within Colab limits
  - Great for Tier 1 projects

TRAINING SETUP

- Image Size: 416
- Epochs: 3 (fast training for Colab)
- Batch Size: 8
- Pretrained weights: yolov8n.pt
- Optimizer + augmentations handled internally by Ultralytics

TRAINING CURVE



MODEL EVALUATION

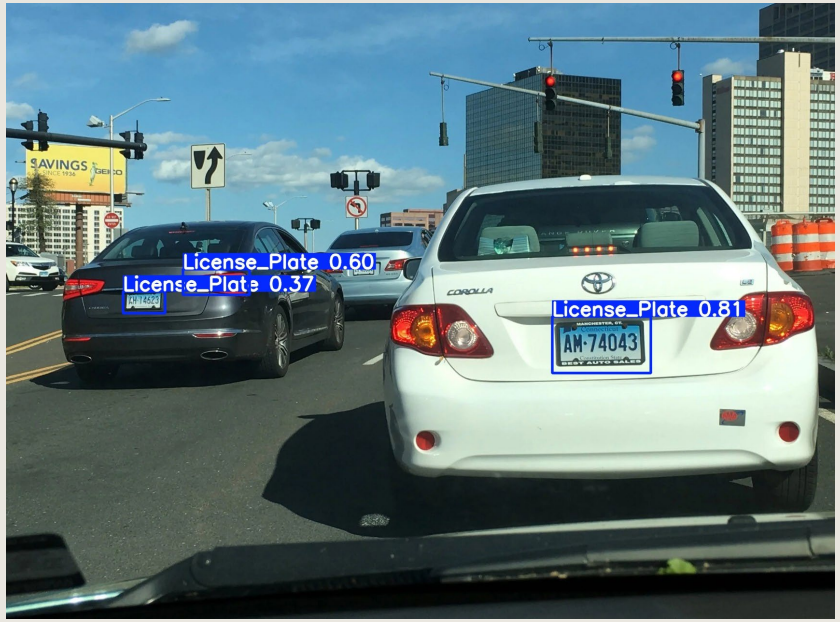
- Precision: 0.90
- Recall:0.88
- mAP50: 0.89

```
metrics = model.val()
metrics_dict = metrics.results_dict
metrics_dict
```

```
Ultralytics 8.3.235 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1222.4±638.4 MB/s, size: 29.1 KB)
val: Scanning /content/License-Plate-Recognition-1/valid/labels.cache... 2048 images, 3 backgrounds, 0 corrupt: 100% ━━━━━━━━━━━━ 2048/2048 3.5Mit/s 0.0s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 128/128 9.6it/s 13.3s
                   all       2048       2195      0.905      0.881      0.892      0.586
Speed: 0.4ms preprocess, 1.4ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to /content/runs/detect/val2
{'metrics/precision(B)': 0.9045897078541533,
 'metrics/recall(B)': 0.8811545163782943,
 'metrics/mAP50(B)': 0.8917370472152344,
 'metrics/mAP50-95(B)': 0.5859516847033878,
 'fitness': 0.5859516847033878}
```

SAMPLE PREDICTIONS

- Model successfully detected plates
- Works on different vehicle types and lighting conditions



DEMO WORKFLOW

Steps shown in the video:

- Load YOLOv8 model
- Download dataset using Roboflow
- Train model for a few epochs
- Evaluate metrics
- Upload a new image
- Run inference & show results

LIMITATIONS

- Only detects plates, does not read characters
- Trained with few epochs
- Performance depends on lighting and image quality
- Single-class detection only

FUTURE IMPROVEMENTS

- Train for more epochs to increase accuracy
- Add OCR using Tesseract or EasyOCR
- Deploy model to a web app (Flask or Streamlit)
- Create a mobile-friendly version