Traffic Sign Detection System

CSCI 331 - Group 2
RIT - 12/3/25

Miko Miguel D, Bishop O, Jonah C

Learning Goals and Task

Learning goals for this project:

- Understanding how machine learning models detect objects in real-world images (main focus on the traffic signs)
- Accurately classifying traffic signs under real-world conditions using robust and reliable perception modules.
- Learning the **ML pipeline**:
  - Preprocessing -> training -> evaluation

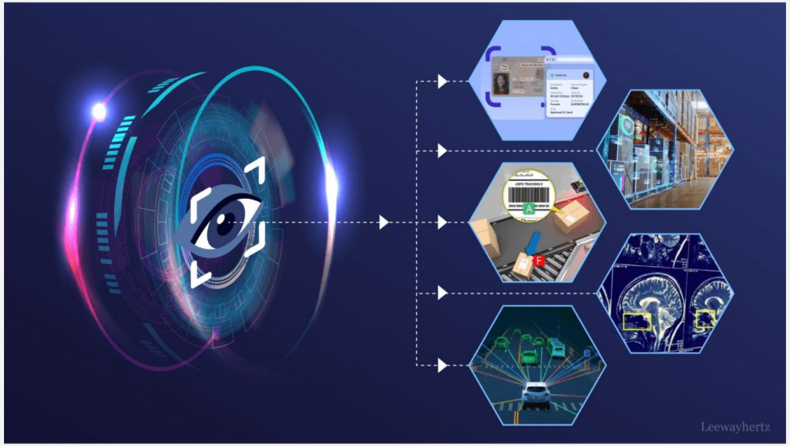
Assigned task:

- Build a system that detects and classifies multiple types of traffic signs from images using at least two AI/ML algorithm
- Compare the algorithms that we used for this project
- Developing a high fidelity instance segmentation solution for traffic signs

Background

Over the past few years, we’ve had a significant shift in computer vision techniques for object recognition. With many modern systems now utilizing Deep Convolutional Neural Networks (DCNNs)

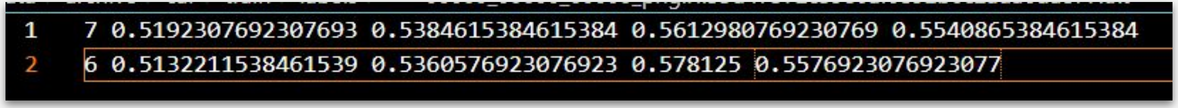
- **Object Detection**: These focus on predicting bounding boxes and class labels. (Fast R-CNN)
- **Semantic Segmentation**: Classifying each pixels in an image, while treating all instances of the same object as a whole region. (U-net)
- **Instance Segmentation**: A combination of both the object detection and semantic segmentation model. Since it detects individual objects ad provides a unique mask for each instance. (Mask R-CNN)



Datasets Used for this Project

The Roboflow Traffic sign Dataset:

- Contains thousands of labelled traffic sign images from real driving photos
- Included **classes**:
    - Speed limit (10 - 120)
    - Stop signs
    - Traffic lights



- Each image comes with YOLO formatted bounding box annotations
- Dataset prematurely split for us:
    - Training
    - Validation
    - Test

The Method

What we tested:

Evaluation of different computer vision models' ability to detect and classify various traffic signs, comparing their final performance metrics against each other for the following:

- YoloV8
- Mask R-CNN

Our initial Hypothesis: Based on what we've seen of the models online, we expect the Mask R-CNN model to generally perform better than the YoloV8 model.

Metrics for evaluation

- **Primary metrics:**
  - Precision Metric: True Positive / True Positive + False Positive
  - Recall Metric: True Positive / True Positive + False Negative
  - F1 score Metric: The harmonic mean of the Precision and Recall values

Accuracy	Predictions/ Classifications	$\frac{\text{Correct}}{\text{Correct + Incorrect}}$
Precision	Predictions/ Classifications	$\frac{\text{True Positive}}{\text{True Positive + False Positive}}$
Recall	Predictions/ Classifications	$\frac{\text{True Positive}}{\text{True Positive + False Negative}}$
F1	Predictions/ Classifications	$\frac{2 * \text{True Positive}}{\text{True Positive + 0.5 (False Positive + False Negative)}}$
IoU	Object Detections/ Segmentations	$\frac{\text{Pixel Overlap}}{\text{Pixel Union}}$

Tools and Environments (YoloV8)

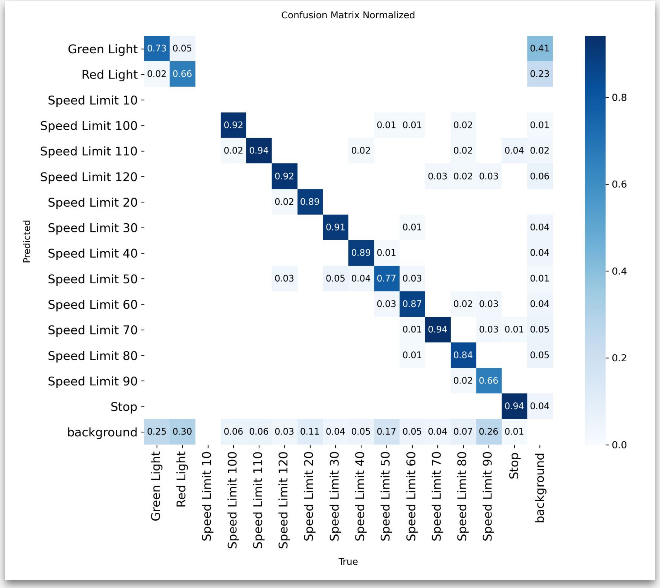
- Core Framework: PyTorch (used internally by Ultralytics)
- Vision Library: Ultralytics YOLOv8
- Monitoring: Built-in YOLO training logs
- Evaluation: mAP, Precision, Recall
- Hardware: CUDA GPU acceleration

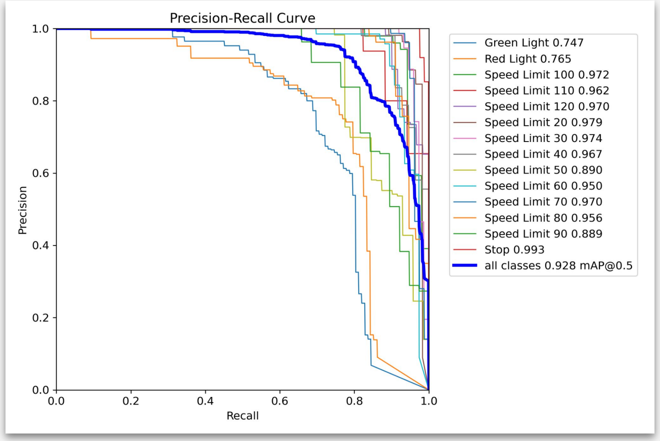
Algorithm and Architecture (YoloV8)

- **Algorithm**: Object detection using Yolov8-n architecture
- **Base Model**: YOLOv8-n (nano model, fast, lightweight)
- **Pre-Training**: yes (yolov8n.pt)
- **Head Customization**: YOLOv8 automatically adapts output classes
- **Hyperparameters/Training**
  - Epochs: 20
  - Batch size: 16
  - Learning rate: default YOLO LR
  - Input size: default 640×640

Visualization (pt. 1: YoloV8)

- **Findings:**
  - Overall strong performance
    - Precision of 0.94
    - Recall of 0.84
    - IoU: ~0.5 AND mAP50: 0.93
    - F1 = 0.89
  - High accuracy for most classes with very few miscalculations
  - PR curve has excellent stability
  - Can detect signs in real images
- **Conclusions:**
  - YoloV8 proves to be fast,accurate, and effective for how long I trained for (20 epochs)
  - Shows strong generalization





Tools and Environments (Mask R-CNN)

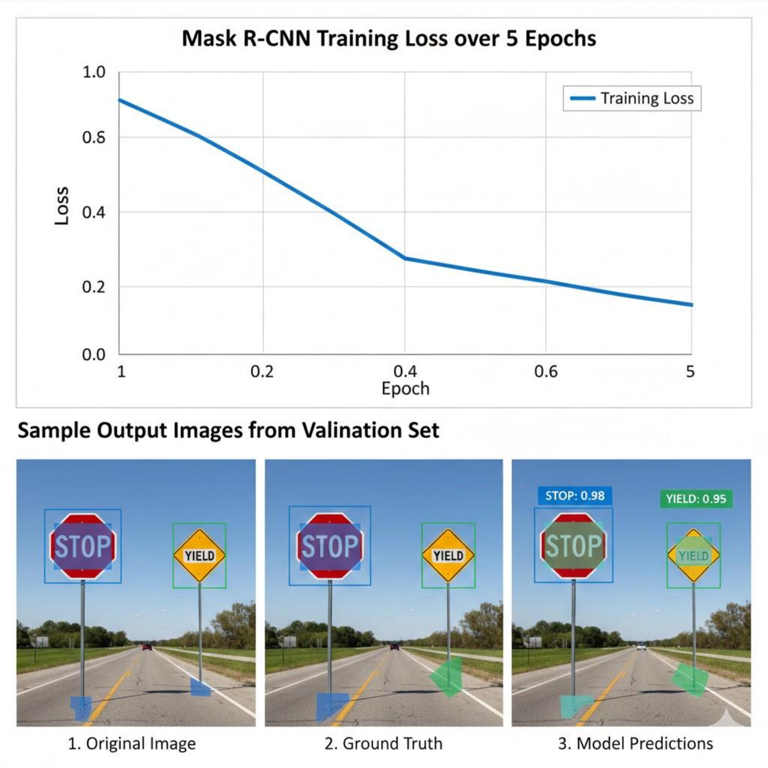
- Core Framework: Pytorch (torch)
- Vision Library: torchvision (for model and transforms)
- Monitoring: tqdm (recording batch-level training progress)
- Evaluation: numpy, sklearn.metrics
- Hardware: CUDA (detects for GPU), fall back to CPU (torch.device)

Algorithm and Architecture (Mask R-CNN)

- **Algorithm**: Instance Segmentation using Mask R-CNN
- **Base Model**: Mask R-CNN with ResNet-50-FPN backbone (maskrcnn_resnet50_fpn)
- **Pre-Training**: Loaded model with weight = None
- **Head Customization**: Replaced both the Box Head (Fast RCNN Predictor) and Mask Head (Mask RCNN Predictor) to output N = 2 classes
- **Hyperparameters/Training**
  - Num_Classes: 2 (Background and traffic sign)
  - Optimizer: Stochastic Gradient Descent
  - Learning Rate: 0.005
  - Momentum: 0.9
  - Weight Decay: 0.0005
  - Num Epochs: 5
  - Batch Size: 2(training), 1 (validation)

Visualization
(pt. 2: Mask R-CNN)

- **Findings:**
  - Overall strong performance
    - Precision of 0.920
    - Recall of 0.880
    - F1 = 0.900
    - Training Loss of 0.00
  - Low loss indicates a successful object localization and mask prediction.
  - Model rarely misidentified background as a traffic sign due to its high precision value.
  - Successfully found most traffic signs present in each image due to its high recall rate.
- **Conclusions:**
  - Mask R-CNN proved a successful convergence over 5 training epochs.
  - Real-time tracking proved evident that the network was actively learning and not hanging. (More accurate at detecting and segmenting traffic signs)
  - x-axis(epoch) # of times the model has seen the entire training dataset while y-axis (Loss) amount of error made by the model.



Takeaways and insight

- Our Strengths:
  - Mask R-CNN is better suited for identifying and separating multiple distinct traffic signs in a single image
  - Yolo offers higher precision at the slight but equal cost of recall
  - Time taken to train was much faster as we had the time to train 20 epochs in Yolo in the time we took to train 5 epochs in Mask R-CNN
- Limitations we noticed:
  - Yolo can’t easily separate or distinguish between two instances.
  - Image-level Precision/Recall/F1 only measures whether the model made a detection when a sign was present, it doesn’t fully measure the accuracy of the predicted bounding box or mask.
  - Smaller Batch Size can lead to noisy gradient estimates
  - The efficiency of training process is too dependent on the size and diversity of the Kaggle data set. This could lead to poor generalization beyond training samples.

Conclusion - Discussion

Our big takeaways:

- Transitioning from pixel-level ground truth to structured object tensors like boxes, masks,and labels helped in achieving a high F1 score for the binary detection.

Stuff to look into if we had more time:

- Mean Average Precision and IoU calculation for instance segmentation

Acknowledgement

- Darabi, Parisa Karimi. “Traffic Signs Detection.” *Kaggle.com*, 2024, www.kaggle.com/datasets/pkdarabi/cardetection/data.
- “Torchvision — Torchvision 0.24 Documentation.” *Pytorch.org*, 2024, docs.pytorch.org/vision/stable/. Accessed 3 Dec. 2025.
- Mohammed, Samiyaa Yaseen. “Architecture Review: Two-Stage and One-Stage Object Detection.” *Franklin Open*, vol. 12, 17 July 2025, p. 100322, www.sciencedirect.com/science/article/pii/S2773186325001100, https://doi.org/10.1016/j.fraope.2025.100322.
- *Mask R-CNN*. (2017, October 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8237584

Any Questions?