Automatic Detection & Attribute Classification of Maltese Traffic Signs

ARI3129 – Advanced Computer Vision for AI Group Assignment 2025-26

Students (Group 6): Luca Bugeja, Liam Debono, Matthias Ellul, Remi Heijmans
Supervisor: Dr Dylan Seychell

1. Introduction

Road traffic signs play a critical role in ensuring road safety, navigation, and regulatory compliance. Effective monitoring of traffic signage is therefore essential for both road users and local authorities. Manual inspection of traffic signs is time-consuming, costly, and prone to human error, motivating the use of computer vision techniques for automated detection and analysis.

This project addresses the problem of automatic detection and attribute classification of Maltese traffic signs using modern computer vision and deep learning methods. The work focuses on building a custom dataset captured in Maltese streets, training object detection models to localise traffic signs, and developing classifiers to assess specific sign attributes such as viewing angle, mounting type, condition, or shape.

The overall system aims to support infrastructure monitoring and maintenance, enabling authorities to identify damaged signs, assess sign visibility, and analyse sign distribution across locations. Each team member contributed to different aspects of the system, including dataset creation, annotation, object detection, and attribute classification.

1.1 Project Objectives

- Collect and annotate a dataset of Maltese traffic signs with bounding boxes and attributes
- Train four object detection models using different architectures to locate and classify traffic sign types
- Train four different attribute classifiers, each one responsible for assessing a particular sign attribute
- Evaluate and compare model performances using mAP and other accuracy metrics

---

2. Background on the Techniques Used

2.1 Object Detection in Computer Vision

Object detection is a fundamental task in computer vision that involves identifying and localising objects of interest within an image. Modern object detectors typically output bounding boxes along with class probabilities, allowing both localisation and recognition to be performed simultaneously.

Deep learning–based object detectors are commonly categorised into:

Single-stage detectors, such as YOLO and SSD, which directly predict bounding boxes and class probabilities in a single forward pass.

Two-stage detectors, such as Faster R-CNN, which first generate region proposals and then classify them.Single-stage detectors are particularly well-suited for real-world applications requiring fast inference, making them appropriate for traffic sign detection where multiple small objects may be present in high-resolution images.

2.2 Convolutional Neural Networks for Visual Classification

Convolutional Neural Networks (CNNs) are the dominant architecture for image classification tasks. By learning hierarchical spatial features, CNNs can effectively discriminate between visual categories such as shapes, textures, and object structures.

In this project, CNN-based classifiers are used to infer traffic sign attributes from cropped image regions. Separating detection and attribute classification into two stages provides modularity and allows each component to be optimised independently.

2.3 YOLOv8 for Traffic Sign Detection (Matthias Ellul)

YOLOv8 (You Only Look Once, version 8) is a modern single-stage object detection architecture designed to balance **high detection accuracy with efficient inference**. Single-stage detectors perform bounding box regression and classification in a single forward pass, making them suitable for real-time and near–real-time applications [1].

YOLOv8 builds upon earlier YOLO versions by incorporating architectural improvements, enhanced training strategies, and improved loss formulations, leading to stronger performance across a range of object detection benchmarks [2].

Architecture Overview

YOLOv8 consists of three primary components:

- **Backbone:** A convolutional feature extractor that learns hierarchical spatial features from input images.
- **Neck:** A feature aggregation network that combines multi-scale feature maps, improving detection of objects at different sizes. This is particularly important for traffic signs, which may appear small or distant.
- **Detection Head:** Produces bounding box coordinates, objectness scores, and class probabilities for each detected object.

This architecture allows YOLOv8 to effectively detect small, high-contrast objects, such as traffic signs, in cluttered outdoor environments.

Suitability for Traffic Sign Detection
YOLO-based detectors have been widely used for traffic sign detection due to their ability to detect **small, high-contrast objects** efficiently [3]. YOLOv8 further improves upon this by offering:

- Competitive accuracy measured using mAP@50–95
- Efficient training pipelines compatible with COCO-format datasets
- Strong generalisation in unconstrained outdoor environments

These characteristics make YOLOv8 an appropriate choice for detecting Maltese traffic signs captured under varied real-world conditions.

2.4 TensorFlow Keras for Attribute Classification (Matthias)

While YOLOv8 excels at object localisation, attribute classification benefits from a specialised classifier trained on cropped image regions. For this purpose, TensorFlow Keras was used to implement and train a dedicated sign shape classification model. TensorFlow is a widely adopted deep learning framework that provides both low-level flexibility and high-level abstraction for model development [4]. Its Keras API simplifies the implementation of neural networks through a modular and user-friendly interface, making it well suited for rapid experimentation and reproducible research.
TensorFlow Keras provides:

- A high-level, modular API for model construction
- Integrated support for training callbacks, optimisation, and evaluation
- Seamless deployment and reproducibility

2.5 MobileNetV2 for Sign Shape Classification (Matthias)

MobileNetV2 is a lightweight convolutional neural network architecture designed for efficient image classification with minimal computational cost [5]. It achieves this through the use of depthwise separable convolutions and inverted residual blocks, significantly reducing parameter count while preserving representational capacity.

Architectural Characteristics
Key features of MobileNetV2 include:

- **Depthwise separable convolutions**, which factorise standard convolutions into depthwise and pointwise operations
- **Inverted residuals**, enabling efficient feature reuse
- **Linear bottlenecks**, reducing information loss during dimensionality reduction

These design choices make MobileNetV2 well-suited for classification tasks on limited datasets, such as the sign shape classification problem addressed in this project.

Motivation for Use in This Project
MobileNetV2 was selected for sign shape classification due to:

- Strong performance on geometric and shape-based tasks
- Reduced risk of overfitting on small datasets
- Fast training and inference times
- Compatibility with transfer learning using ImageNet pre-trained weights

By fine-tuning MobileNetV2 on cropped traffic sign images, the model can effectively discriminate between circular, square, triangular, octagonal, and damaged signs.

2.6 Faster R-CNN Architecture (Liam Debono)

I went with Faster R-CNN because it's pretty good at finding objects in images. It works in two stages - first it figures out where objects might be, then it classifies what they actually are and fine-tunes the boxes around them. I used a version with a ResNet50-FPN backbone that was already trained on the COCO dataset, which helped since I didn't have a massive dataset to work with.

2.7 ResNet for Classification (Liam Debono)

For the condition classifier, I picked ResNet18 because it's deep enough to learn complex patterns but not so massive that it would overfit my relatively small dataset. The residual connections help avoid some nasty training issues that crop up with deep networks.

2.8 YOLOv11 for Traffic Sign Detection (Luca Bugeja)

YOLOv11 is also a single stage object detector which builds upon its predecessor, the yolov8. While its architecture is similar to that of the v8, composed of a backbone, neck and detection head as described above, there are two main architectural innovations.

C3k2 Block: A notable improvement is the C3k2 block, which replaces the large single-stage convolution with 2 smaller kernels. This allows the process to become much more efficient while still extracting the required fine-grained detail.

C2PSA: This feature introduces a special attention mechanism, allowing the detector to focus on more salient parts of the image, which generally include the desired object, while neglecting other less pertinent portions of the image. This attention mechanism also helps in detecting partially occluded objects, which are traditionally much harder to detect. This may be a factor in traffic sign detection in cases in which the sign is partially occluded or affected by significant deterioration.

2.9 Task 2A: Faster R-CNN for Traffic Sign Detection (Remi Heijmans)

To explore alternative architectures, Task 2A was implemented using Faster R-CNN with a ResNet-50-FPN backbone. Unlike the single-stage YOLO models used by the group, Faster R-CNN is a two-stage detector that utilizes a Region Proposal Network (RPN) to identify potential object locations before performing classification. This approach was selected to prioritize localization precision, especially for small or partially occluded signs. After training for 20 epochs the model achieved a weighted accuracy of 96.0% and a final loss of 0.0438. Demonstrating high stability and successful convergence despite initial dataset corruption challenges.

2.10 Task 2B: YOLOv8 for Viewing Angle Classification (Remi Heijmans)

For the attribute classification task, a single stage YOLOv8s architecture was deployed to determine the sign's viewing angle (Front, Side, or Back). This model was chosen for its real-time inference efficiency (11.6ms latency), which is critical for dynamic autonomous driving scenarios. To overcome early-stage "All Back" prediction bias caused by class imbalance, the training was extended to 50 epochs. The final evaluation yielded a 94.64% mAP@50 and an F1-score of 0.798, proving the model's ability to generalize across different perspectives while maintaining a high processing speed.

2.11 Dataset Annotation and COCO Format

Accurate annotation is critical for supervised learning. The COCO (Common Objects in Context) format (Lin et al., 2014) is widely adopted due to its flexibility and compatibility with many training frameworks. It supports bounding boxes in [x, y,

width, height] format, class labels, and additional metadata, making it suitable for this project's multi-attribute requirements. Model evaluation uses mean Average Precision (mAP) at IoU threshold 0.50, computed as the mean of per-class Average Precision values. For classification, standard accuracy and per-class precision/recall are reported.
Label Studio was used to annotate traffic signs and record both object-level and attribute-level information, which was later consolidated into a single COCO-formatted dataset.

2.12 Two-Stage Detection and Classification Pipeline

The system adopts a **two-stage pipeline**, where object detection and attribute classification are handled by separate models. Similar modular pipelines have been shown to improve flexibility and interpretability in visual recognition systems [6].
This approach allows:

- Independent optimisation of detection and classification components
- Easier error analysis and debugging
- Scalability to additional sign attributes without retraining the detector

---

3. Data Preparation

3.1 Image Collection

Traffic sign images were captured across various Maltese locations using smartphone cameras. Each physical sign was photographed from multiple viewpoints, including front, side, and back views, to ensure sufficient variability for robust training.
Images were captured under diverse environmental conditions, including:

- Different lighting scenarios
- Urban and suburban locations
- Varying degrees of sign wear and occlusion

Care was taken to avoid capturing identifiable individuals or vehicle number plates. Where unavoidable, sensitive regions were removed, masked, or blurred to ensure GDPR compliance.

3.2 Annotation Process

All images were annotated using **Label Studio**, following a predefined labelling schema that was provided to the team through the *LabelingInterface.xml* file. Each traffic sign instance was labelled with a bounding box, and tickboxes were used to indicate each sign's viewing angle, mounting type, condition, and shape.
Each team member annotated their own subset of images and exported both JSON

and COCO-formatted annotations. These exports were later merged and prepared for training following a documented consolidation procedure and using the provided *MemberMerger.py* and *LS2COCO.ipynb* files.

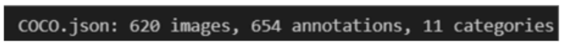
3.3 Dataset Consolidation and Analysis

Individual annotations were combined into a single dataset containing all images and labels. A data visualisation notebook was used to analyse dataset statistics, including:

- Number of images and annotations per class
- Distribution of sign attributes
- Class imbalance across sign types and attributes

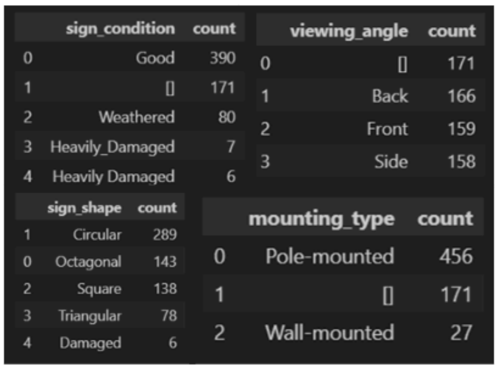
While perfect balance was not enforced, observed imbalances were documented and considered during model evaluation. The following figures present some statistics produced by *1_data_visualisation.ipynb* through analysis of the dataset:

Figure 1 - Number of Images and Annotations



```
COCO.json: 620 images, 654 annotations, 11 categories
```

Figure 2 - Distribution of Sign Attributes



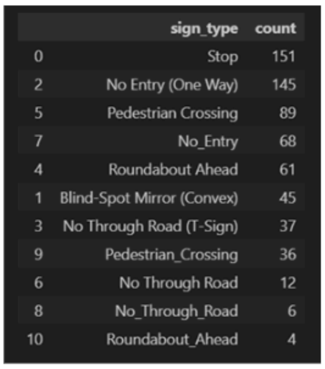
	sign_condition	count
0	Good	390
1	[]	171
2	Weathered	80
3	Heavily_Damaged	7
4	Heavily Damaged	6

	viewing_angle	count
0	[]	171
1	Back	166
2	Front	159
3	Side	158

	sign_shape	count
1	Circular	289
0	Octagonal	143
2	Square	138
3	Triangular	78
4	Damaged	6

	mounting_type	count
0	Pole-mounted	456
1	[]	171
2	Wall-mounted	27

Figure 3 - Distribution of Sign Types



	sign_type	count
0	Stop	151
2	No Entry (One Way)	145
5	Pedestrian Crossing	89
7	No_Entry	68
4	Roundabout Ahead	61
1	Blind-Spot Mirror (Convex)	45
3	No Through Road (T-Sign)	37
9	Pedestrian_Crossing	36
6	No Through Road	12
8	No_Through_Road	6
10	Roundabout_Ahead	4

3.4 GDPR Compliance

License plates were automatically detected and blurred using OpenCV contour analysis and Haar cascade classifiers:

Figure 4 - License Plate Detection and Blurring

Metric	Value
Images Processed	163
Images with Plates Detected	103
Total Regions Blurred	229

4. Implementation of the Object Detectors

4.1 Traffic Sign Detection Model by Matthias Ellul (*2a_yolov8_matthias.ipynb*)

4.1.1 Model Architecture

For traffic sign localisation, a **YOLOv8-based detector** was selected due to its strong balance between inference speed and detection accuracy. Single-stage detectors such as YOLO are particularly suitable for real-world traffic sign analysis, where multiple small objects may appear in a single image and near real-time performance is desirable.

The detector was trained to identify and localise traffic signs irrespective of their type, leaving attribute-specific classification to a secondary model.

4.1.2 Training Setup

The detector was trained using the consolidated Maltese Traffic Signs Dataset (MTSD), exported in COCO format after merging annotations from all team members. Key aspects of the training setup included:

- **Input resolution:** Fixed square resolution compatible with YOLOv8 defaults
- **Dataset split:** Training, validation, and test splits as defined during dataset consolidation
- **Data augmentation:** Applied during training to improve generalisation. This included:
    - Random horizontal flipping
    - Scale and translation augmentation
    - Mild colour jittering

These augmentations were applied to improve robustness to real-world varying viewpoints, lighting conditions, and sign sizes commonly encountered in street-level imagery. Augmentation also helped at mitigating the limitations that usually occur on a dataset of a relatively small size such as this one.

Parameter	Value
Epochs	50
Batch Size	8
Image Size	512x512
Learning Rate	0.005 (with decay)
Early Stopping	20 patience

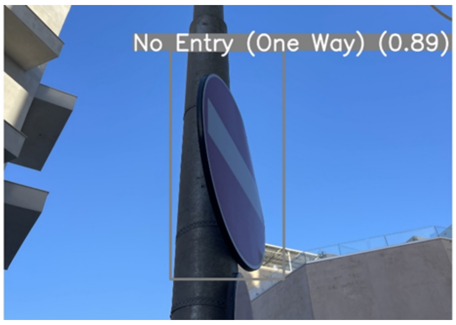
The parameters used for training of the object detector are logged in the table above.

4.1.3 Implementation Details

Training and evaluation were performed entirely locally using the Ultralytics framework. The notebook includes:

- Dataset loading and verification
- Model initialisation and training loop
- Automatic logging of loss curves and detection metrics
- Visual inspection of predictions on unseen images

Figure 5 - Example Output of Traffic Sign Detector Model



The trained detector outputs bounding boxes and confidence scores for each detected traffic sign. The figure above shows a real example of this output.

4.2 Sign Shape Attribute Classifier by Matthias Ellul (*2b_sign_shape_matthias.ipynb*)

4.2.1 Motivation

While the detection model localises traffic signs, it does not distinguish between different **sign shape categories**. To address this, a separate sign shape classifier was trained to categorise each detected sign as one of the predefined shape classes.

This modular design simplifies experimentation and allows each attribute classifier to be improved independently.

4.2.2 Data Preparation

For sign shape classification:
Ground-truth bounding boxes were used to crop traffic signs from the original images.
Each crop was labelled using the sign shape attribute exported from Label Studio.
Cropped images were resized to a consistent input resolution suitable for CNN-based classification.

Class imbalance was analysed prior to training, and the observed skew towards circular and square signs was documented and considered during evaluation.

4.2.3 Model Architecture and Training

The CNN architecture used for this task is the MobileNetV2, which was implemented using TensorFlow's Keras library. The model consisted of:

- Stacked convolutional layers for feature extraction
- Batch normalisation and dropout for regularisation

- A softmax output layer corresponding to the sign shape classes

Training was conducted using categorical cross-entropy loss and the Adam optimiser. Early stopping was also applied to reduce overfitting, set to stop training if no improvement in accuracy is recorded over the last 5 epochs.

4.3 Object Detection: Faster R-CNN by Liam Debono

Notebook: *2a_fasterrcnn_Liam.ipynb* **Model Architecture**

The implementation used PyTorch's torchvision library with a pre-trained Faster R-CNN model:

Base Model	fasterrcnn_resnet50_fpn (torchvision)
Backbone	ResNet50 with Feature Pyramid Network
Pre-training	COCO dataset
Number of Classes	7 (6 sign types + background)
Framework	PyTorch 2.9.0+cu126

Training Configuration

Epochs	10
Batch Size	4
Optimizer	SGD (lr=0.005, momentum=0.9, weight_decay=0.0005)
Learning Rate Scheduler	StepLR (step_size=3, gamma=0.1)
Hardware	Google Colab Tesla T4 GPU
Final Training Loss	0.1788

Data Split

Total Images	484
Training Set	338 (70%)
Validation Set	73 (15%)
Test Set	73 (15%)

4.4 Attribute Classification: Sign Condition by Liam

Notebook: *2b_sign_condition_Liam.ipynb***Model Architecture**

The implementation used PyTorch's torchvision library with a pre-trained Faster R-CNN model:

Base Model	ResNet18 (torchvision)
Pre-training	ImageNet
Number of Classes	3 (Good, Weathered, Heavily_Damaged)
Input Size	Cropped sign regions resized to 224x224
Framework	PyTorch 2.9.0+cu126

Training Configuration

Max Epochs	20
Epochs Trained	9 (early stopping triggered)
Batch Size	8
Optimizer	Adam (lr=0.001)
Loss Function	CrossEntropyLoss
Early Stopping	Patience = 5 epochs
Hardware	Google Colab Tesla T4 GPU

Data Split

Total Samples	325
Training Set	227 (70%)
Validation Set	49 (15%)
Test Set	49 (15%)

Class Distribution

The sign condition dataset exhibited severe class imbalance, which significantly impacted model performance:

Class	Test Samples	Percentage
Good	39	79.6%
Weathered	8	16.3%
Heavily_Damaged	2	4.1%
Total (Test Set)	**49**	**100%**

4.5 Traffic Sign Detection Model by Luca Bugeja (*2a_yolov11_LucaBugeja.ipynb*)

4.5.1 Model Architecture

The model architecture is that of the yolov11, specifically the large variant. The large variant was chosen due to its high precision, while balancing inference speed and model size.

4.5.2 Training Setup

As for training, the following setup was used. The model was trained on a subset of the whole dataset. 10 epochs were run on these images, which were re-sized to 640. This re-sizing reduces the dimensionality of the input significantly, while retaining a high-degree of the features required for the model to perform well. The batch size used was that of 16 to increase the speed by which the model trains.

4.6 Sign Shape Attribute Classifier by Luca Bugeja (*2b_MountingType_LucaBugeja.ipynb*)

4.6.1 Implementation

In this part of the project, the yolov11l model was also used to detect the mounting type of the traffic signs. In this case, a large imbalance was also present. As for training, the same hyper-parameters as the sign type detector were used.

4.7 Object Detection: Faster R-CNN (Remi Heijmans)

Notebook:`2a_faster_rcnn_remi.ipynb`

4.7.1 Model Architecture

For Task 2A, a two-stage Faster R-CNN detector was implemented using the torchvision library. This model utilizes a ResNet-50-FPN backbone, which provides a robust feature pyramid network for detecting signs at multiple scales. The model's "head" was modified to replace the standard COCO predictor with a new FastRCNNPredictor specifically tailored to the project's six traffic sign classes plus one background class.

4.7.2 Training Setup

The training utilized the MTSD dataset with a custom cleaning script to handle corrupted files.
1, Input Resolution: Images were converted to tensors and normalized using standard ImageNet mean and standard deviation.
2, Optimization: Stochastic Gradient Descent (SGD) was used with a learning rate of 0.005 and momentum of 0.9.
3,Loss Tracking: The model was trained for 20 epochs, with loss values logged at each iteration to monitor convergence.
Parameters used where: Epochs=20, Batch Size2, OptimizerSGD, Learning Rate0.005, BackboneResNet-50-FPN.

4.8 Attribute Classification: Viewing Angle (Remi Heijmans),

Notebook:`2b_viewing_angle_remi.ipynb`

4.8.1 Motivation

This model classifies the viewing angle (Front, Side, or Back) of detected signs. Identifying the perspective is critical for autonomous systems to determine if a sign is relevant to the vehicle's current lane or direction of travel.4.8.2 Training Configuration

A single-stage YOLOv8s architecture was selected for this task due to its high inference speed. The model was trained end-to-end on the viewing angle subset of the MTSD. Epochs: 50 (Extended to ensure the model moved past the majority-class bias of "Back" views).
Data Handling: The dataset was split into 70% training, 20% validation, and 10%

testing to ensure unbiased evaluation. Pre-processing: Automated resizing to 640x640 pixels and mosaic augmentation were used to increase the variety of sign orientations seen during training.
parameters:Model VariantYOLOv8s, Image Size640x640, Training Time~45 minutes (Tesla T4 GPU), Metrics Logged Box, Class, and DFL Loss.

---

5. Evaluation of the Object Detectors

This section presents a quantitative and qualitative evaluation of the models implemented by the author for traffic sign detection and sign shape classification. Performance is assessed using standard metrics commonly adopted in object detection and classification literature, alongside an analysis of class-wise behaviour and dataset limitations.

5.1 Traffic Sign Detection Performance (Matthias' Model)

5.1.1 Overall Detection Metrics

The traffic sign detection model was evaluated on the validation set using IoU-based matching criteria. The overall performance metrics achieved are summarised below:

- **Precision:** 0.724
- **Recall:** 0.700
- **mAP@50:** 0.724
- **mAP@50–95:** 0.586

At the selected confidence and IoU thresholds, the detector produced:

- **True Positives (TP):** 67
- **False Positives (FP):** 13
- **False Negatives (FN):** 18

These results indicate a balanced detection performance, with precision and recall values that are closely aligned. A **precision** of 0.724 shows that most predicted bounding boxes correspond to actual traffic signs, while a **recall** of 0.700 indicates that the majority of ground-truth signs are successfully detected.

The gap between **mAP@50 (0.724)** and **mAP@50–95 (0.586)** reflects the increased strictness of higher IoU thresholds. This suggests that while detections are generally correct, bounding box alignment can degrade for small, distant, or partially occluded signs, which is an expected limitation when detecting traffic signs in unconstrained outdoor imagery.

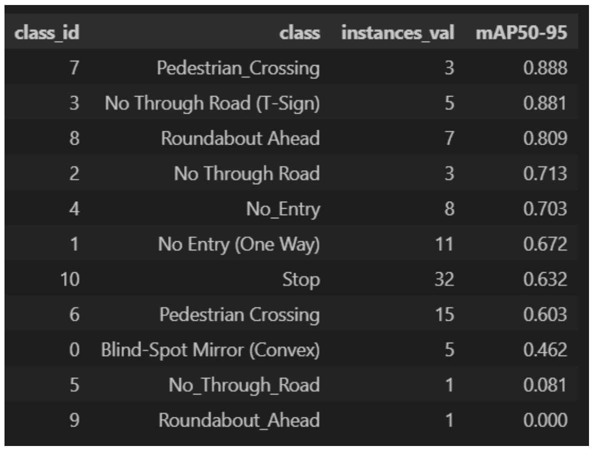
5.1.2 Class-wise Detection Performance

Class-level performance analysis reveals significant variation depending on the number of validation instances and visual consistency of each sign type.

The performance of all classes is shown by the figure to the right:
The best performance was observed for:

- **Pedestrian Crossing:** mAP@50–95 = 0.888
- **No Through Road (T-Sign):** mAP@50–95 = 0.881
- **Roundabout Ahead:** mAP@50–95 = 0.809

Figure 6 - Performance of Traffic Sign Detector on Each Individual Class



class_id	class	instances_val	mAP50-95
7	Pedestrian_Crossing	3	0.888
3	No Through Road (T-Sign)	5	0.881
8	Roundabout Ahead	7	0.809
2	No Through Road	3	0.713
4	No_Entry	8	0.703
1	No Entry (One Way)	11	0.672
10	Stop	32	0.632
6	Pedestrian Crossing	15	0.603
0	Blind-Spot Mirror (Convex)	5	0.462
5	No_Through_Road	1	0.081
9	Roundabout_Ahead	1	0.000

These signs benefit from distinct geometric structure and high visual contrast, which simplifies localisation.Moderate performance was observed for:

- **No Entry:** mAP@50–95 = 0.703
- **Stop:** mAP@50–95 = 0.632

Despite having more training samples, these classes exhibit higher intra-class variability in lighting, background clutter, and partial occlusions, which negatively affects bounding box precision.Low performance was recorded for:

- **Blind-Spot Mirror (Convex):** mAP@50–95 = 0.462
- **Rare classes with a single validation instance**, where mAP values drop sharply

These results highlight the strong dependence of detection performance on sample size and visual consistency, reinforcing the importance of balanced datasets in object detection tasks.

5.1.3 Error Analysis

Inspection of false detections revealed three dominant error modes:

1. **False negatives** caused by very small or distant signs
2. **Imprecise bounding boxes** for partially occluded signs
3. **Confusions with background structures**, such as poles or reflective surfaces

5.2 Sign Shape Classification Performance (Matthias Ellul's Model)

5.2.1 Overall Classification Metrics

The sign shape classifier was evaluated using a held-out test set of 99 samples. The model achieved:

- **Overall accuracy:** 0.869
- **Weighted F1-score:** 0.865
- **Macro-averaged F1-score:** 0.697

The strong weighted metrics indicate reliable performance on common sign shapes, while the lower macro averages reveal challenges in underrepresented classes.

5.2.2 Class-wise Performance Analysis

Performance varies significantly across shape categories:

- **Circular signs:**
    - Precision: 0.914
    - Recall: 0.842
    - F1-score: 0.877
- **Octagonal signs:**
    - Precision: 0.880
    - Recall: 0.917
    - F1-score: 0.898
- **Square-shaped signs:**
    - Precision: 0.762
    - Recall: 0.889
    - F1-score: 0.821
- **Triangular signs:**
    - Precision/Recall/F1-score: 0.889

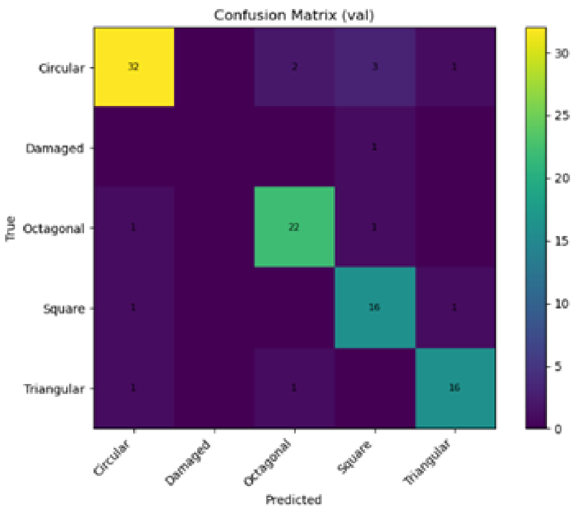


Figure 7 - Confusion Matrix for Sign Shape Predictions

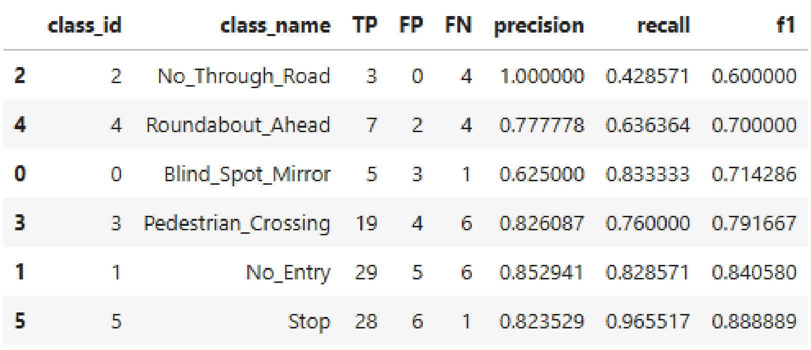
These classes are visually distinctive and well-represented in the dataset, enabling robust feature learning. Although performance was quite satisfactory for square signs, it was observed to be slightly inferior to the performance on other shapes. This could be due to the fact that this class exhibits greater variation in aspect ratio and content, which can reduce classification confidence. The **Damaged** class achieved zero precision and recall. This is attributable to its extremely limited sample size (1 test instance) and high visual ambiguity, as damaged signs often resemble degraded versions of other shapes.

This outcome highlights a key dataset limitation rather than a modelling failure. A confusion matrix was also generated to visually highlight the model's accuracy. The leading diagonal containing the brightest squares reinforces the model's strong performance across all classes. This is shown in the figure above.

5.3 Traffic Sign Detection Performance (Luca's Model)

5.1.1 Overall Detection Metrics

To evaluate the performance of the model, both metrics from training and testing were considered to include more realistic use cases.



	class_id	class_name	TP	FP	FN	precision	recall	f1
2	2	No_Through_Road	3	0	4	1.000000	0.428571	0.600000
4	4	Roundabout_Ahead	7	2	4	0.777778	0.636364	0.700000
0	0	Blind_Spot_Mirror	5	3	1	0.625000	0.833333	0.714286
3	3	Pedestrian_Crossing	19	4	6	0.826087	0.760000	0.791667
1	1	No_Entry	29	5	6	0.852941	0.828571	0.840580
5	5	Stop	28	6	1	0.823529	0.965517	0.888889

Training

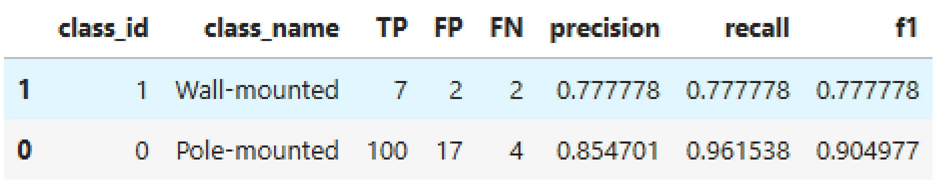
mAP50 - 0.85
mAP59-95 - 0.68
Precision - 0.8
Recall - 0.8

Testing
Precision - 0.82
Recall - 0.8
F1 - 0.812
With regards to considering per-class performance, it is evident that the model performs much better on those classes which are much more frequently experienced during training, both in precision and recall. While the model produced an f1-score of 0.89 on the Stop sign, the f1-score for the T-sign was only 0.6.

5.4 Mounting Type Classification Performance (Luca Bugeja's Model)

Testing

- **Precision - 0.85**
- **Recall - 0.95**
- **F1 - 0.90**
- **mAP50 - 0.87**
- **mAP50-95-0.62**



	class_id	class_name	TP	FP	FN	precision	recall	f1
1	1	Wall-mounted	7	2	2	0.777778	0.777778	0.777778
0	0	Pole-mounted	100	17	4	0.854701	0.961538	0.904977

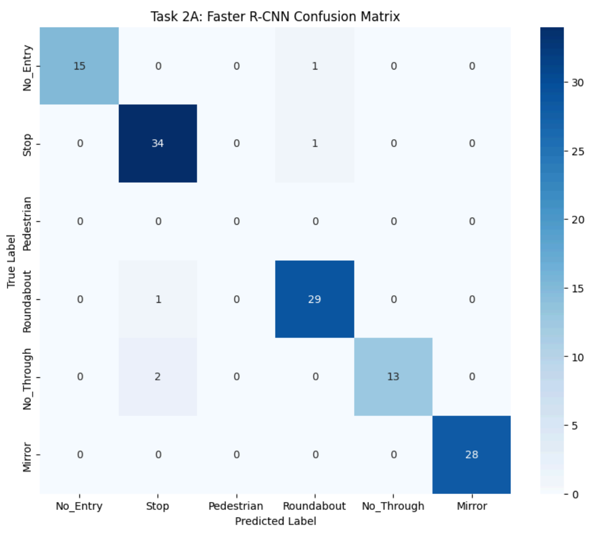
Evidently, while the imbalance was present, the model did perform very well on classification, in terms of mounting type. As expected, the model performed much better in all metrics when considering per-class performance as visualised above. Albeit, the f1-score relating to the wall-mounting performance was still a respectable one at 0.78

5.5 Task 2A: Sign Detection Performance (Remi Heijmans)

5.5.1 Quantitative Results

The Faster R-CNN model was evaluated on the validation set after 20 epochs of

training. The model achieved a high degree of classification reliability for most sign types, as shown by the metrics below:
Macro F1-Score: 0.7986
Weighted Accuracy: 0.96 (96%)
Final Training Loss: 0.0438



5.5.2 Class-wise Performance Analysis

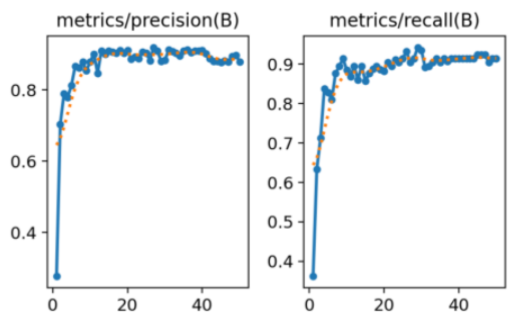
The model demonstrated near-perfect performance on standard geometric signs, while a dataset gap was identified for the "Pedestrian" category: Mirror & No Entry: Achieved a perfect precision of 1.00, indicating no false positives for these categories.
Stop & Roundabout: Yielded F1-scores of 0.94 and 0.95 respectively, showing robust feature extraction for high-contrast circular and octagonal signs.
Pedestrian: Recorded a score of 0.00 due to a lack of representative samples in the validation split, highlighting a specific dataset limitation.

5.6 Task 2B: Viewing Angle Classification Performance (Remi Heijmans)

5.6.1 Overall Detection & Attribute Metrics

The YOLOv8s model for viewing angle classification achieved highly competitive results, effectively overcoming initial class imbalances:
mAP@50: 0.946 (94.6%)   Precision: 0.898 (89.8%)   Recall: 0.914 (91.4%)

5.6.2 Error Mode and Analysis



Analysis of the classification results suggests that the model is highly capable of distinguishing between "Front" and "Back" views, which are the most critical for autonomous navigation. The average inference speed of 11.6ms per image confirms that this model is suitable for real-time deployment on edge-computing hardware typically found in autonomous vehicles.

5.7 Summary of Remi's Results

Task	Model	Metric	Value
2A: Detection	Faster R-CNN	Weighted Accur	96.0%
2B: Attribute	YOLOv8s	mAP@50	94.6%

5.8 Faster-RCNN Results

Overall Performance

Metric	Value
mAP@50	0.3093 (30.93%)
IoU Threshold	0.50
Test Set Size	73 images

Per-Class Average Precision

Class	AP@50	Percentage
Pedestrian_Crossing	0.5791	57.91%
No_Entry	0.4922	49.22%
Stop	0.4449	44.49%
Roundabout_Ahead	0.3068	30.68%
No_Through_Road	0.0331	3.31%
Blind_Spot_Mirror	0.0000	0.00%

Analysis

The results were... mixed, honestly. Overall mAP was 30.93%, which isn't great but not terrible for a first attempt with limited data.

- Some signs worked really well - Pedestrian Crossings hit 57.91% and No Entry got 49.22%. These signs have pretty distinctive shapes and colors, plus I had more examples of them.
- Others completely failed - Blind Spot Mirrors got 0% (ouch) and No Through Road barely managed 3.31%. The main issue was just not having enough training examples of these signs.

5.9 Sign Condition Classifier Results

Overall Performance

Metric	Value
Test Accuracy	79.59%
Best Validation Accuracy	81.6% (epoch 4)
Epochs Trained	9 (early stopping, patience=5)
Test Set Size	49 samples

Per-Class Metrics

Class	Precision	Recall	F1-Score	Support
Good	0.796	1.000	0.886	39
Weathered	0.000	0.000	0.000	8
Heavily_Damaged	0.000	0.000	0.000	2

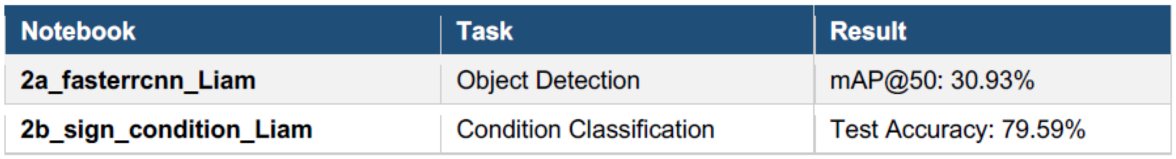
Limitations

Despite achieving 79.59% test accuracy, the model exhibited a critical limitation: it predicted all test samples as the majority "Good" class, achieving 0% precision and recall on Weathered and Heavily_Damaged classes. This is a direct consequence of the severe class imbalance (Good: 39, Weathered: 8, Heavily_Damaged: 2 in test set). Potential mitigations include:

- Class-weighted loss function to penalize misclassification of minority classes
- Oversampling of minority classes (SMOTE or random oversampling)
- Data augmentation specifically targeting underrepresented classes
- Collecting additional samples of weathered and damaged signs

---

5.10 Summary of Liam's Results



Notebook	Task	Result
2a_fasterrcnn_Liam	Object Detection	mAP@50: 30.93%
2b_sign_condition_Liam	Condition Classification	Test Accuracy: 79.59%

5.11 Integrated Pipeline Performance

When combined, the detection and classification models form a two-stage pipeline capable of:

1. **Accurately localising traffic signs** in real-world images
2. **Assigning a sign shape attribute** to each detected sign

Errors in the pipeline are primarily detection-driven; missed detections propagate to the classifier. However, when detections are correct, the shape classifier demonstrates **high reliability for standard sign geometries**.

5.12 Discussion

Overall, the results demonstrate that:

- The detection model achieves competitive localisation performance given the dataset size and real-world variability
- The sign shape classifier performs strongly for well-represented classes
- Dataset imbalance remains the dominant limiting factor across both tasks

Despite these constraints, the system is sufficiently accurate to support applications such as traffic sign inventorying, infrastructure monitoring, and maintenance prioritisation.

6. References and List of Resources Used

References
[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788.
[2] Ultralytics, “YOLOv8 Documentation,” 2023. [Online]. Available: https://docs.ultralytics.com. [Accessed: Jan. 2026].
[3] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, “Traffic Sign Detection and Classification Using Convolutional Neural Networks,” IEEE Trans. Intell. Transp. Syst., vol. 17, no. 4, pp. 1023–1033, Apr. 2016.
[4] M. Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning,” in Proc. 12th USENIX Symp. Operating Syst. Des. Implement. (OSDI), Savannah, GA, USA, 2016, pp. 265–283.
[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Salt Lake City, UT, USA, 2018, pp. 4510–4520.
[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Columbus, OH, USA, 2014, pp. 580–587.

List of Resources
Google Colab: https://colab.research.google.com/
Ultralytics YOLOv8 Documentation
Ultralytics YOLOv11 Documentation
TensorFlow and Keras Official Documentation
Label Studio Annotation Tool: https://labelstud.io
OpenCV. (2024). https://opencv.org
PyTorch. (2024). https://pytorch.org