Waste Object Detection Using Deep Learning

Authors

Muhammad Usama Javaid

Ahmed Khalil

Jean Lattard

Supervisor

Prof Joaquin Jorge Rodriguez

Master in Computer Vision & Robotics

Université de Bourgogne

Date: December 12, 2025

Abstract

This project focuses on developing a deep learning-based system for detecting waste objects in images. The goal is to support automated waste sorting by identifying four categories of recyclable materials: Aluminum Cans (AluCan), Glass bottles, HDPE Plastic (HDPEM), and PET Plastic bottles.

We developed a custom object detection model that combines a frozen pretrained YOLOv8s backbone with a custom Adaptive Bidirectional Feature Pyramid (ABFP) neck and decoupled attention-based detection heads. The model was trained on a dataset of 4,811 annotated images using standard preprocessing, several data augmentation techniques, and a combination of Focal Loss, Smooth L1 Loss, and Binary Cross-Entropy Loss.

The custom model achieved an mAP@0.5 of approximately 63.40%, with an overall recall of 72.69% and precision of 57.68%, meaning the model detects most objects but still produces some false positives. Performance varied across classes: Glass achieved the highest F1-score (71.21%), while PET was the most challenging class (60.95% F1-score).

These results demonstrate that while the custom architecture functions correctly and achieves moderate detection performance, it does not outperform a fully fine-tuned YOLOv8 model. The frozen backbone and limited dataset size restricted the model's ability to learn domain-specific waste features. Nonetheless, the work provides a foundation for future improvements, including backbone fine-tuning, larger datasets, and deeper ablation studies.

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Problem Definition . . . 1
- 1.3 Project Scope . . . 1
- 1.4 Objectives and Contributions . . . 1
- 1.5 Contributions . . . 2
- 1.6 Real-World Applications . . . 2
- 1.7 Structure of the report . . . 2

2 Dataset and Preprocessing **3**
- 2.1 Dataset Overview . . . 3
- 2.2 Annotation Format . . . 3
- 2.3 Class Distribution . . . 3
- 2.4 Sample Images . . . 4
- 2.5 Preprocessing Steps . . . 5
- 2.6 Summary . . . 6

3 Models and Methodology **7**
- 3.1 Overview . . . 7
- 3.2 YOLOv8s Backbone (Pretrained) . . . 7
- 3.3 Custom Model Architecture . . . 8
  - 3.3.1 Adaptive Bidirectional Feature Pyramid (ABFP) . . . 9
  - 3.3.2 Decoupled Attention Head . . . 9
- 3.4 Loss Functions . . . 9
- 3.5 Training Pipeline . . . 10
- 3.6 Summary . . . 10

4 Training Methodology **11**
- 4.1 Training Pipeline Overview . . . 11
- 4.2 Loss Functions . . . 11
  - 4.2.1 Bounding Box Regression Loss (Smooth L1 loss) . . . 11
  - 4.2.2 Classification Loss (Focal loss) . . . 11
  - 4.2.3 Objectness Loss (Binary Cross-Entropy) . . . 11
- 4.3 Optimisation Strategy . . . 11
  - 4.3.1 Optimizer . . . 11
  - 4.3.2 Learning Rate Scheduling . . . 11
- 4.4 Training Configuration . . . 12
- 4.5 Model Outputs During Training . . . 12
- 4.6 Summary . . . 12

5 Results and Evaluation **13**
- 5.1 Evaluation Metrics Overview . . . 13
- 5.2 Confidence-Based Curves . . . 13
- 5.3 Confusion Matrices . . . 14
  - 5.3.1 Custom ABFP Model . . . 15
  - 5.3.2 YOLOv8 (Reference Baseline) . . . 15
  - 5.3.3 ResNet50 Confusion Matrix . . . 16
- 5.4 Validation Batch Visualisations . . . 16
  - 5.4.1 Validation Batch 0 . . . 17
  - 5.4.2 Validation Batch 1 . . . 18
- 5.5 Predictions on Individual Images . . . 19

5.6 ResNet50 Precision–Recall Curve . . . 20
5.7 Qualitative Results (Custom ABFP Model) . . . 20
5.8 Summary . . . 22
6 Model Comparison **23**
6.1 Overall Metric Comparison . . . 23
6.2 Confusion Matrix Comparison . . . 23
6.2.1 Custom ABFP Model . . . 24
6.2.2 YOLOv8 Model . . . 25
6.2.3 ResNet50 Detector . . . 26
6.3 Precision–Recall Curve Comparison . . . 26
6.3.1 Custom ABFP Model PR Curve . . . 27
6.4 Qualitative Comparison of Model Predictions . . . 27
6.5 Discussion . . . 28
7 Conclusion **29**
References **30**

List of Figures

2.1 Distribution of waste categories in the dataset. . . . 4
2.2 Example images from the dataset showing different waste categories and environments. 5

3.1 General structure of the YOLOv8 backbone. . . . 8
3.2 High-level architecture of our custom model. . . . 9
3.3 Simplified training pipeline for the detection models (placeholder figure). . . . 10

5.1 Custom model training history. . . . 13
5.2 Confidence-based evaluation curves for Yolov8. . . . 14
5.3 Confusion matrix for the custom ABFP model. . . . 15
5.4 Confusion matrix for the pretrained YOLOv8 model (reference only). . . . 15
5.5 Confusion matrix for the ResNet50 detector. . . . 16
5.6 Ground-truth labels for validation batch 0 Yolov8. . . . 17
5.7 Predictions for validation batch 0 Yolov8. . . . 17
5.8 Ground-truth labels for validation batch 1. . . . 18
5.9 Predictions for validation batch 1. . . . 18
5.10 Example predictions on individual validation images Custom model. . . . 19
5.11 Precision–Recall curve for the ResNet50 detector. . . . 20
5.12 Qualitative predictions from the ABFP model (Set 1). . . . 21
5.13 Qualitative predictions from the ABFP model (Set 2). . . . 21

6.1 Confusion matrix for the custom ABFP model. . . . 24
6.2 Confusion matrix for the pretrained YOLOv8 model (reference). . . . 25
6.3 Confusion matrix for the ResNet50 detector. . . . 26
6.4 Precision–Recall curve comparison between the custom ABFP model and the ResNet50 detector. . . . 27
6.5 Qualitative comparison of predictions produced by the custom ABFP model and the ResNet50 detector. . . . 27

List of Tables

4.1 Training configuration used for the waste detection models. 12

6.1 Overall comparison of the three models. 23

Chapter 1
Introduction

1.1 Background and Motivation

Waste production is increasing every year due to population growth, high consumption, and urban living. As a result, modern waste management systems are under pressure to sort and recycle large amounts of materials quickly and correctly. Traditional sorting methods depend mostly on manual work, which can be slow, tiring, and not always accurate.

With the rise of computer vision and deep learning, it is now possible to teach machines to recognize different types of objects from images. These technologies are already used in areas like face recognition, medical imaging, and self-driving cars. In the same way, they can also help improve how waste is detected and sorted.

This project aims to explore how deep learning can be used to automatically detect different types of waste items in images. This is an important first step toward building smarter and more efficient recycling systems.

1.2 Problem Definition

The main goal of this project is to detect common recyclable waste items in images and identify their category. The system focuses on four main waste types:

- Aluminum cans (AluCan)
- Glass bottles
- HDPE milk containers (HDPEM)
- PET plastic bottles

The task is limited to **object detection**. This means the model should find the item in the image, draw a bounding box around it, and assign the correct label. We are **not** building a complete sorting or recycling system — only the detection step.

1.3 Project Scope

This semester project includes the following steps:

- Creating and preparing a dataset of annotated waste images.
- Training different deep learning models, including pretrained models and a custom model.
- Measuring performance using standard metrics such as precision, recall, F1-score, and mAP.
- Studying where the models perform well and where they struggle.
- Showing visual examples of correct and incorrect detections.

For now the project focuses only on the vision and detection part. No robotic arms, sorting machinery or automation are included.

1.4 Objectives and Contributions

This work has several main objectives:

- To understand how well deep learning models can detect waste items in real-world images.
- To compare pretrained models (YOLOv8s, ResNet50-FPN) with a smaller custom-built model.
- To design a training setup that is stable and easy to reproduce.
- To analyze the results and extract useful insights for future improvements.

1.5 Contributions

The main contributions of this project are:

- We developed a custom waste detection model combining a frozen YOLOv8s backbone with a newly designed Adaptive Bidirectional Feature Pyramid (ABFP) neck and decoupled attention-based detection heads.
- We trained the custom model on a 4-class waste detection dataset and evaluated it using mAP, precision, recall, per-class metrics, confusion matrices, and qualitative detection results.
- We documented the complete custom training pipeline, including data augmentation, loss functions, training configuration, and architectural design.
- We provide an honest performance analysis, showing that while the model reaches high recall, its overall precision and mAP remain moderate and do not surpass expectations for a fully fine-tuned YOLOv8 baseline.
- We identify key limitations—such as the frozen backbone and dataset size—that restricted model performance, and we outline clear directions for future work.

1.6 Real-World Applications

Although the project only handles detection, this step is important for many real-world systems. Examples include:

- Smart bins that recognize what users throw away.
- Robots that help sort waste in recycling centers.
- Tools for cleaning beaches or parks by detecting litter.
- Mobile apps that help people understand which items can be recycled.

Reliable detection is the foundation for all of these systems.

1.7 Structure of the report

This report is divided into several chapters:

- **Chapter 1** introduces the project and explains why the work is important.
- **Chapter 2** describes the dataset and how it was prepared.
- **Chapter 3** explains the models used in the project.
- **Chapter 4** presents the training setup and evaluation methods.
- **Chapter 5** shows the results and discusses what they mean.
- **Chapter 6** concludes the work and suggests possible future improvements.

This structure helps guide the reader through the problem, the method, and the final outcomes.

Chapter 2
Dataset and Preprocessing

2.1 Dataset Overview

The dataset used in this project contains a total of 4,811 images, each showing one or more waste items. Every image has an associated label file following the YOLO annotation format. The dataset includes four main categories of recyclable waste:

- **AluCan** — aluminum drink cans
- **Glass** — glass bottles and containers
- **HDPEM** — high-density polyethylene milk containers
- **PET** — plastic water and soda bottles

These items appear in different lighting conditions, backgrounds, and orientations, which helps the model learn more general features. The dataset was split into training and validation sets as follows:

- **Training set:** 80% (3,848 images)
- **Validation set:** 20% (963 images)

All images were resized to **640×640** pixels to match the input size required by the detection models.

2.2 Annotation Format

Each image has a corresponding text file containing bounding box coordinates in YOLO format. Each line in the label file represents one object using the structure:

```
class_id   x_center   y_center   width   height
```

All values are normalized between 0 and 1. For example:

```
0 0.5498 0.2437 0.2949 0.1771
```

This format makes the dataset compatible with YOLO-based models and many other object detection frameworks.

2.3 Class Distribution

The dataset is not perfectly balanced. Some classes, such as PET and HDPEM, appear more frequently than others. This imbalance can affect training and is later discussed in the results and challenges sections.

Figure 2.1 shows the distribution of images across the four categories.

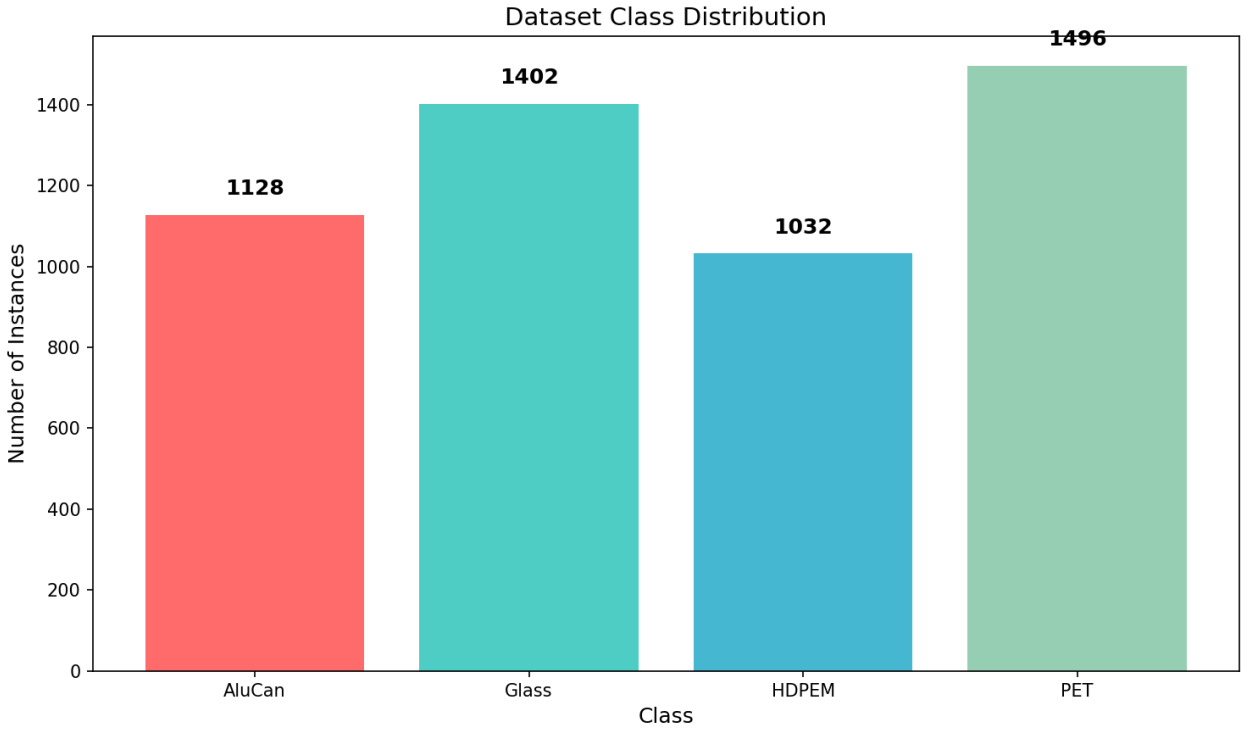


Figure 2.1: Distribution of waste categories in the dataset.

2.4 Sample Images

To give a better idea of what the dataset looks like, Figure 2.2 shows ten sample images taken from different parts of the dataset. These examples include objects of different sizes, shapes, and colours, placed in a variety of environments such as floors, tables, carpets, and outdoor areas. Some images contain single objects, while others have multiple items together.

The purpose of showing these samples is to highlight the diversity that the model needs to learn from. The changes in lighting, perspective, background texture, and object orientation make the detection task more challenging, but also help the model become more general and robust during training.

Figure 2.2: Example images from the dataset showing different waste categories and environments.

2.5 Preprocessing Steps

Before training, each image passes through several preprocessing steps to ensure consistent input and improve the model's ability to learn:

- **Resizing:** All images resized to 640×640 pixels.
- **Normalization:** Pixel values normalized using standard ImageNet mean and standard deviation.
- **Data augmentation (training only):**
  - Horizontal flipping
  - Random rotation (up to ±15°)
  - Color jitter (brightness and contrast changes)
  - Random scaling (80%–120%)

These steps help reduce overfitting and make the model more robust to changes in lighting, perspective, and background.

2.6 Summary

This chapter presented the dataset used in the project, the annotation format, and the preprocessing steps applied during training. The dataset contains a diverse collection of waste images, making it suitable for building a practical and generalizable waste detection model. The next chapter describes the models and methods used in the project.

Chapter 3
Models and Methodology

3.1 Overview

This chapter describes the deep learning models used in our waste detection project and explains the main ideas behind their design. We worked with both pretrained models and a custom-built architecture to understand how different approaches perform on our dataset. The goal was not only to compare accuracy, but also to study model size, speed, and behaviour in real images.

The three main models used in this project are:

- A pretrained **YOLOv8s** model (used mainly as a strong baseline),
- A pretrained **ResNet50** also trained for experiment,
- A **custom model** combining YOLOv8's backbone with our own ABFP neck and decoupled head.

Each model has different strengths. YOLOv8s offers strong performance out of the box, while our custom model gives more flexibility to adjust training, feature fusion, and loss weighting.

3.2 YOLOv8s Backbone (Pretrained)

YOLOv8s is a modern and efficient object detection model trained on the COCO dataset. We use its backbone as a fixed feature extractor. This means that all of its layers remain unchanged during training. The idea is to make use of the general visual features it has already learned on thousands of images of real-world objects.

The YOLOv8 backbone produces feature maps at three different scales:

- P3: high-resolution features for small objects,
- P4: medium-resolution features,
- P5: low-resolution features for larger objects.

These multi-scale outputs allow the model to detect items of different sizes, which is useful in our dataset because waste items appear in many shapes and positions.

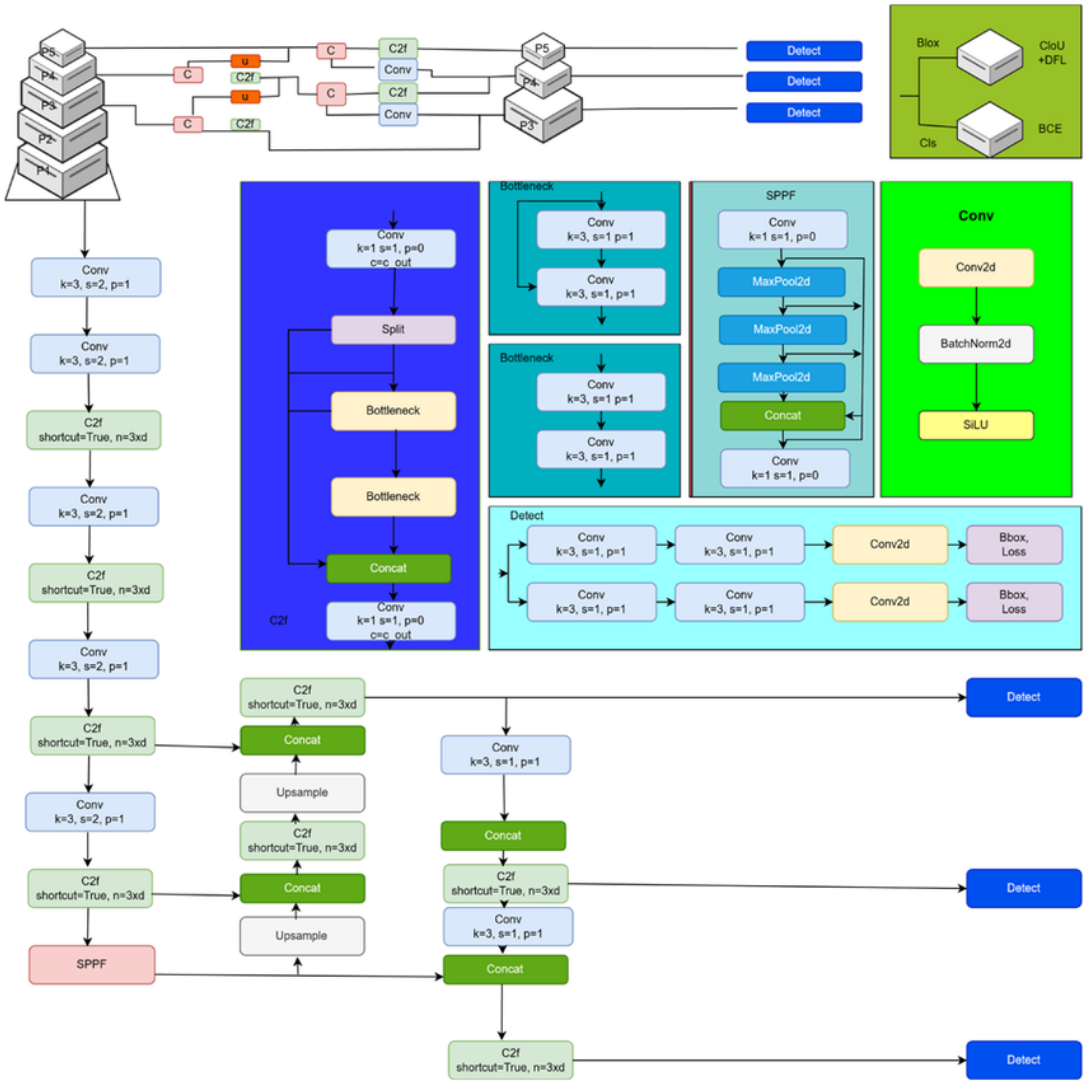


Figure 3.1: General structure of the YOLOv8 backbone.

3.3 Custom Model Architecture

The main contribution of our work is a custom object detection model. It uses the YOLOv8s backbone but introduces two key components that we designed ourselves:

- An **Adaptive Bidirectional Feature Pyramid (ABFP)** as the neck,
- A **Decoupled Attention Head** for final predictions.

A high-level view of our model is shown in Figure 3.2.

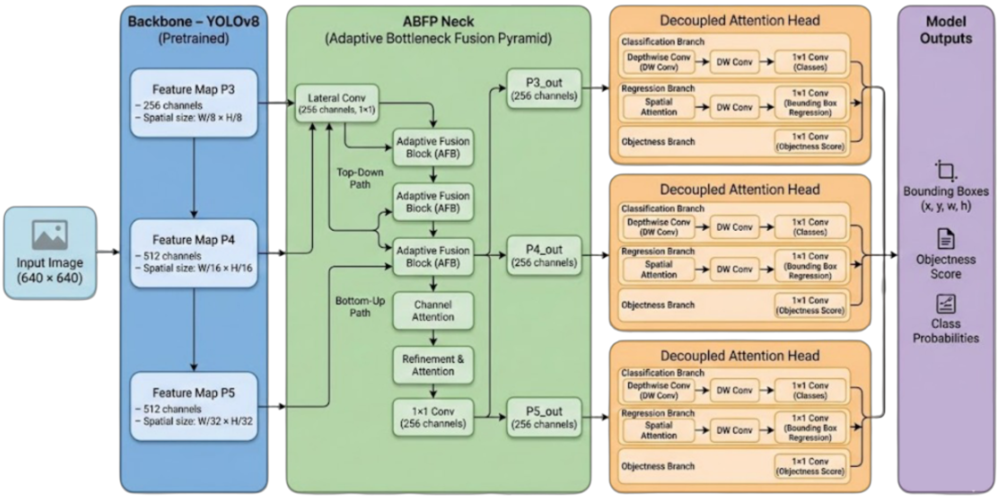


Figure 3.2: High-level architecture of our custom model.

3.3.1 Adaptive Bidirectional Feature Pyramid (ABFP)

The neck of an object detection model combines features from different scales. Instead of simply adding feature maps together, our ABFP neck learns how much weight to assign to each feature map using learnable fusion weights. This makes the feature combination process more flexible.

The ABFP includes:

- **Learnable fusion weights** — The model learns how important each scale is.
- **Channel attention** — Highlights important channels while reducing noise.
- **Spatial attention** — Helps the model focus on useful regions in the image.
- **Depthwise separable convolutions** — Reduces computation while keeping accuracy high.

These design choices make the neck more lightweight and adaptable than a traditional FPN.

3.3.2 Decoupled Attention Head

In standard YOLO-style architectures, classification and bounding box regression share the same layers. In our model, we separate these two tasks so each branch can specialise:

- The **classification branch** learns semantic features,
- The **regression branch** learns precise spatial features.

Both branches include attention layers that make predictions more stable and focused.

This structure improves feature quality and makes the model easier to debug during training.

3.4 Loss Functions

Object detection requires balancing several tasks at the same time. Our model uses three loss components:

- **Classification Loss** — Focal Loss Helps the model pay more attention to difficult examples.
- **Regression Loss** — Smooth L1 Loss Encourages stable and accurate bounding box predictions.
- **Objectness Loss** — Binary Cross-Entropy Determines whether a predicted box contains an object.

The final loss is a weighted sum:

$$\mathcal{L}_{total} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{reg}\mathcal{L}_{reg} + \lambda_{obj}\mathcal{L}_{obj} \tag{3.1}$$

The weights were adjusted during development to correct the imbalance between high recall and low precision.

3.5 Training Pipeline

Figure 3.3 shows a simplified version of our training pipeline. It includes data loading, preprocessing, augmentation, forward pass, and loss computation.

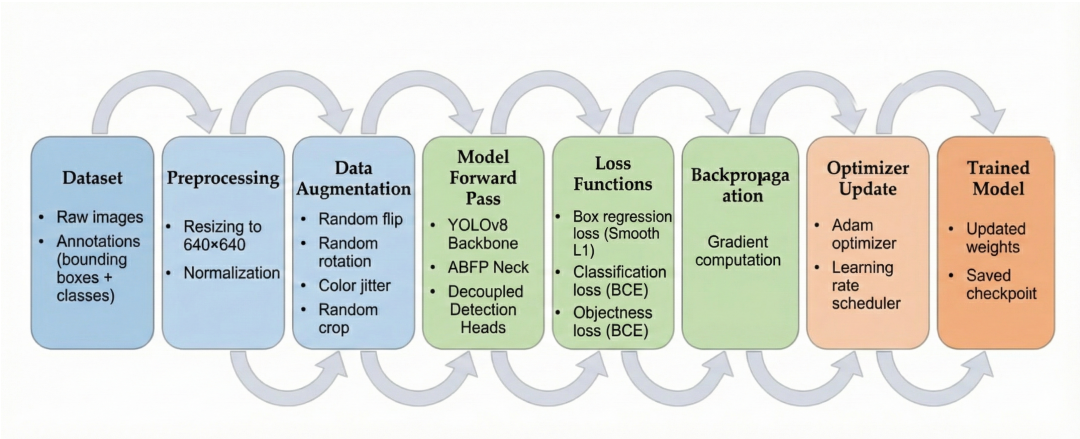


Figure 3.3: Simplified training pipeline for the detection models (placeholder figure).

An outline of the workflow is shown below:

Algorithm 1: Training Pipeline Overview

Input: Training images, annotations, model parameters
Output: Trained model weights
for *each epoch* **do**
  for *each batch* **do**
    Load and preprocess images
    Apply augmentations
    Run forward pass through the model
    Compute classification, regression, and objectness losses
    Sum weighted losses
    Backpropagate gradients and update model weights
Save best-performing model based on validation loss

3.6 Summary

In this chapter, we presented the pretrained models used in the project and described the design of our custom architecture. The combination of YOLOv8 features, adaptive feature fusion, and a decoupled attention head allows the model to be both lightweight and flexible. The next chapter explains the training setup and evaluation metrics in more detail.

Chapter 4
Training Methodology

This chapter explains how the waste detection models were trained. While Chapter 3 focused on the model architectures themselves, this chapter describes the full training workflow, the loss functions used, the optimisation strategy, and the practical steps taken during training.

4.1 Training Pipeline Overview

Figure 3.3 shows the complete training workflow used in this project. The diagram highlights the main stages the data passes through during training.

The process begins with the annotated dataset. Each image contains one or more waste objects, along with bounding boxes and class labels. Before feeding the images into the model, they are resized to 640×640 and normalised. To help the model generalise better, several data augmentation techniques are applied, including random flips, rotations, colour jitter, and random crops.

Once an image has been preprocessed and augmented, it is passed through the model in a forward pass. The model outputs predicted bounding boxes, objectness scores, and class probabilities. These predictions are then compared with the ground truth annotations to compute the training loss.

Finally, backpropagation is used to calculate gradients, and the optimiser updates the model weights. This process repeats for each batch and continues for multiple epochs until the model converges.

4.2 Loss Functions

To train the detection models effectively, three types of loss functions were used. Each loss plays a separate role in helping the model learn correct bounding box positions, objectness, and class predictions.

4.2.1 Bounding Box Regression Loss (Smooth L1 loss)

We used the L1 loss to measure how well the predicted bounding boxes match the ground truth boxes. This makes it more stable and accurate than simpler losses such as IoU or GIoU.

4.2.2 Classification Loss (Focal loss)

For class prediction, Focal loss was used. It compares the predicted class probabilities with the true labels. Since our dataset contains multiple waste categories, BCE helps the model learn which class each object belongs to.

4.2.3 Objectness Loss (Binary Cross-Entropy)

Objectness loss is used to determine whether a predicted box actually contains an object or not. This helps reduce false positives and encourages the model to focus on meaningful regions in the image.

4.3 Optimisation Strategy

4.3.1 Optimizer

We used the Adam w optimiser during training. Adam adapts the learning rate for each parameter, which makes training smoother and more stable, especially in early epochs.

4.3.2 Learning Rate Scheduling

A learning rate scheduler was used to gradually reduce the learning rate over time. This prevents the model from overshooting the optimal solution and helps it converge more effectively.

The training curves in Figure **??** show how both training and validation losses decreased steadily over the epochs, indicating stable learning.

4.4 Training Configuration

Table 4.1 summarises the main training settings. These values were chosen based on common practices for object detection and by observing how the model behaved during early experiments.

Parameter	**Value**
Input Image Size	640×640
Batch Size	16
Number of Epochs	50
Optimizer	Adam
Initial Learning Rate	0.001
Learning Rate Scheduler	Cosine decay
Loss Functions, Focal loss (classification), BCE (objectness)	
Data Augmentation	Flip, rotation, jitter, crop

Table 4.1: Training configuration used for the waste detection models.

4.5 Model Outputs During Training

Throughout the training process, the model produces three outputs for each predicted bounding box:

- **Bounding box coordinates** (x, y, w, h),
- **Objectness score**, which indicates whether a box contains an object,
- **Class probabilities**, one for each waste category.

These values are compared with the ground truth to compute the losses described earlier. Improvements in these outputs from epoch to epoch show that the model is learning meaningful patterns in the data.

4.6 Summary

This chapter described the complete training process used for the waste detection models. The workflow included preprocessing, data augmentation, forward passes, loss computation, and weight updates using the Adam optimiser. With 50 epochs of training and effective loss functions, the models were able to learn to detect and classify waste objects in a variety of environments. The next chapter presents the evaluation results and discusses the performance of each model.

Chapter 5
Results and Evaluation

This chapter presents the evaluation of the models used in this project. We report quantitative metrics such as mean Average Precision (mAP), precision, recall, and F1-score, along with confidence curves, confusion matrices, and qualitative prediction examples. The goal is to understand how well the custom model performs and how its behaviour compares to ResNet50 and YOLOv8.

5.1 Evaluation Metrics Overview

The custom ABFP model achieved improved performance after training for 50 epochs. The main evaluation metrics on the validation set are summarised below:

- **mAP@0.5:** 63.40%
- **Precision:** 57.68%
- **Recall:** 72.69%
- **Overall F1-score:** (varies per class)

These results indicate that the model successfully detects most objects (high recall) but produces a number of false positives (lower precision).

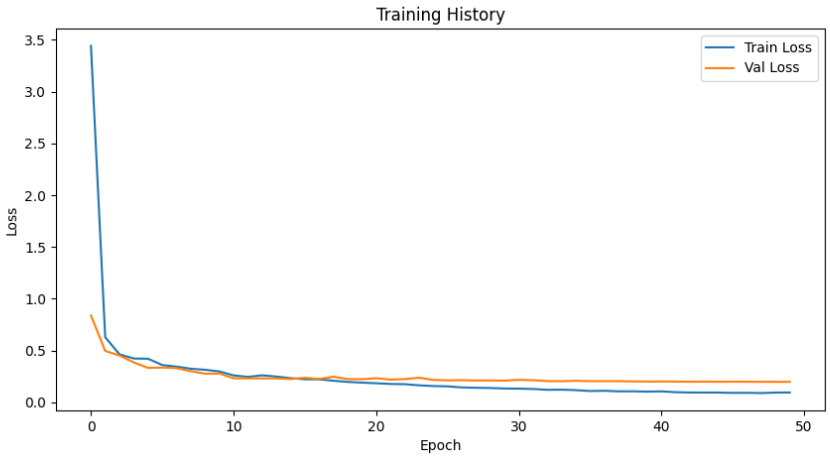
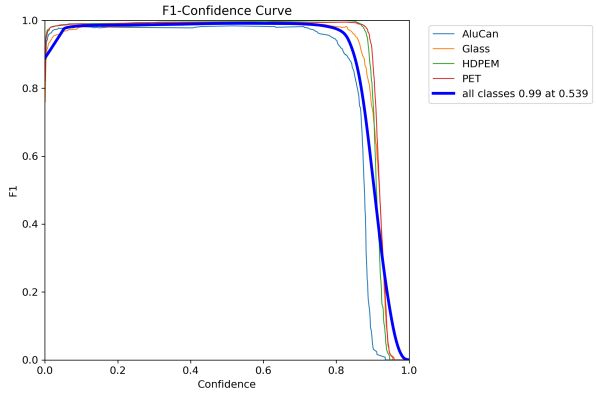


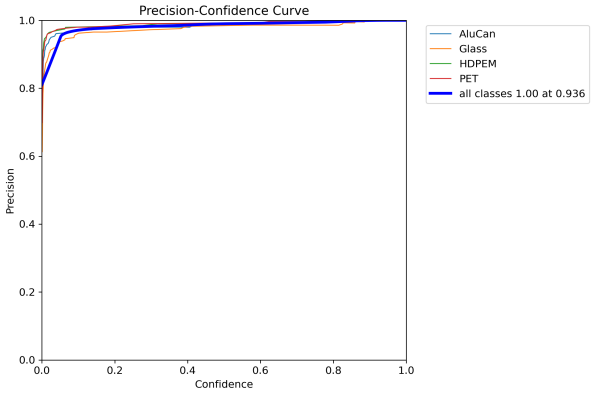
Figure 5.1: Custom model training history.

5.2 Confidence-Based Curves

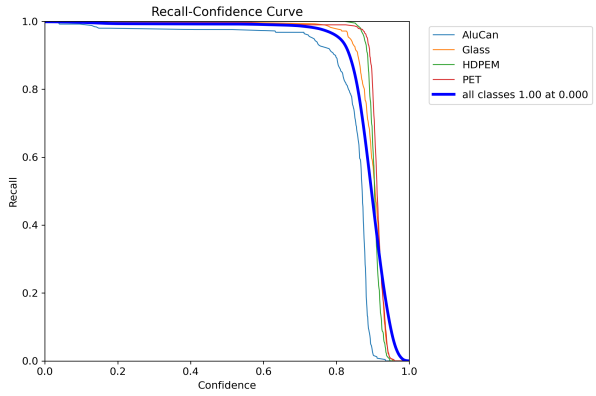
Confidence-based curves illustrate how the model behaves under different decision thresholds. Figures 5.2a–5.2d present the F1-score, precision, recall, and precision–recall curves.



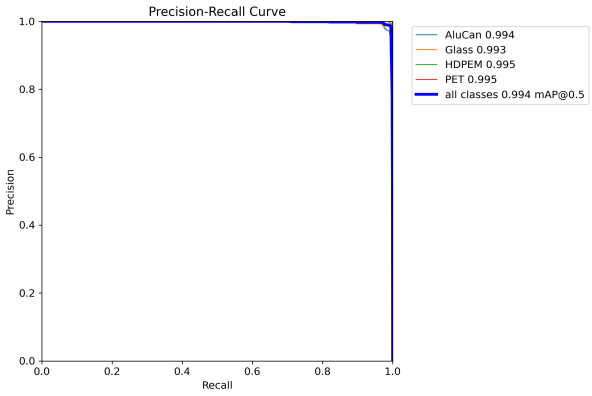
(a) F1-score vs. confidence threshold.



(b) Precision vs. confidence threshold.



(c) Recall vs. confidence threshold.



(d) Precision–Recall curve.

Figure 5.2: Confidence-based evaluation curves for Yolov8.

5.3 Confusion Matrices

Confusion matrices reveal how frequently the model predicts each class correctly and where misclassifications occur.

5.3.1 Custom ABFP Model

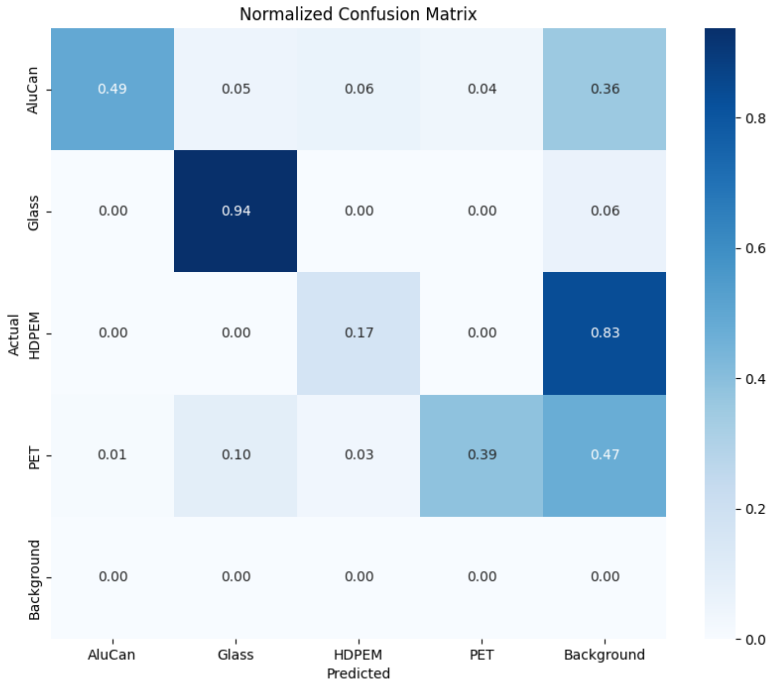


Figure 5.3: Confusion matrix for the custom ABFP model.

5.3.2 YOLOv8 (Reference Baseline)

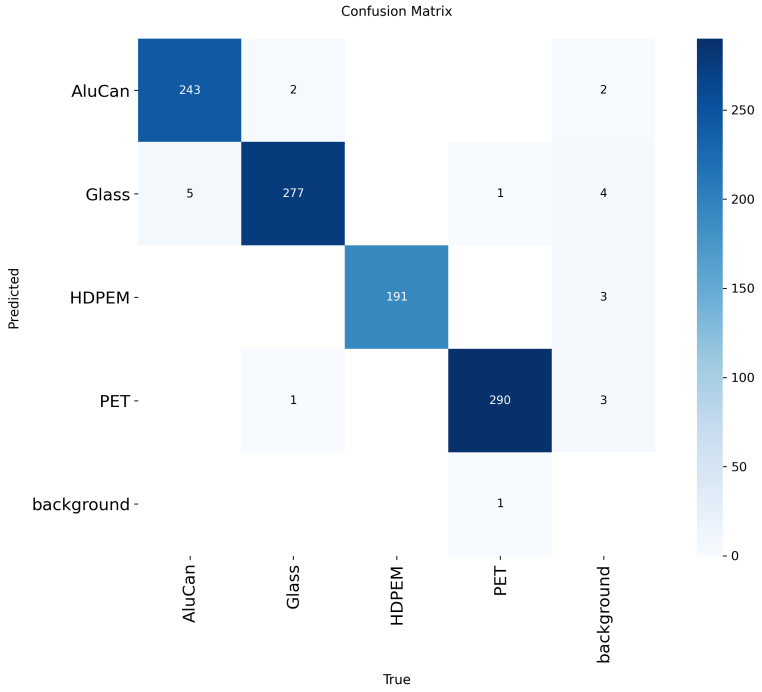


Figure 5.4: Confusion matrix for the pretrained YOLOv8 model (reference only).

5.3.3 ResNet50 Confusion Matrix

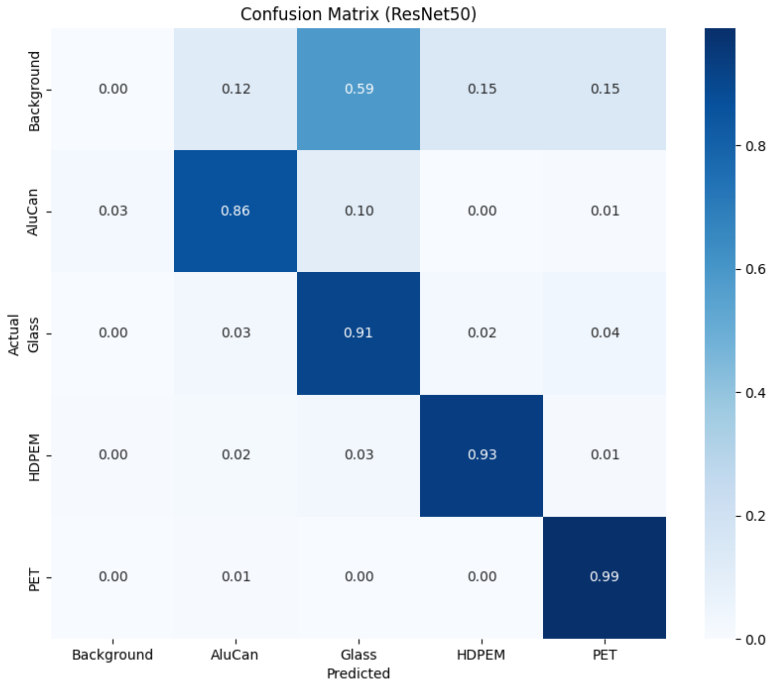


Figure 5.5: Confusion matrix for the ResNet50 detector.

5.4 Validation Batch Visualisations

To further understand the behaviour of the models, labelled images and prediction results are shown below.

5.4.1 Validation Batch 0

Figure 5.6: Ground-truth labels for validation batch 0 Yolov8.

Figure 5.7: Predictions for validation batch 0 Yolov8.

5.4.2 Validation Batch 1

Figure 5.8: Ground-truth labels for validation batch 1.

Figure 5.9: Predictions for validation batch 1.

5.5 Predictions on Individual Images

(a) PET image prediction.

(b) Glass image prediction.

(c) AluCan image prediction.

(d) HDPEM image prediction.

Figure 5.10: Example predictions on individual validation images Custom model.

5.6 ResNet50 Precision–Recall Curve

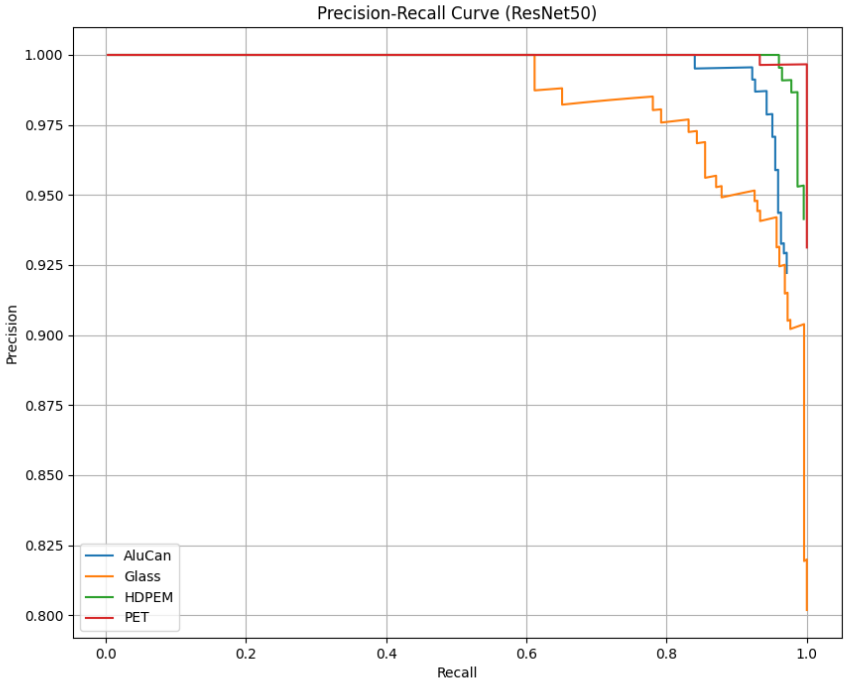
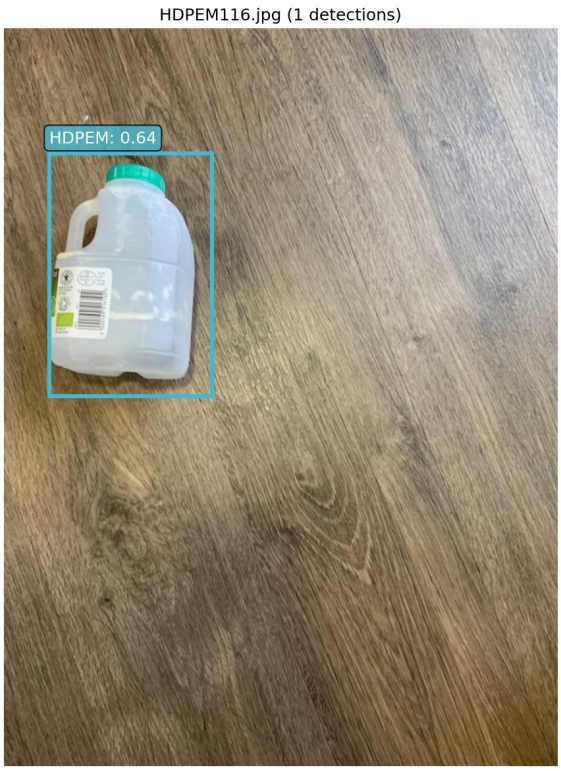


Figure 5.11: Precision–Recall curve for the ResNet50 detector.

5.7 Qualitative Results (Custom ABFP Model)

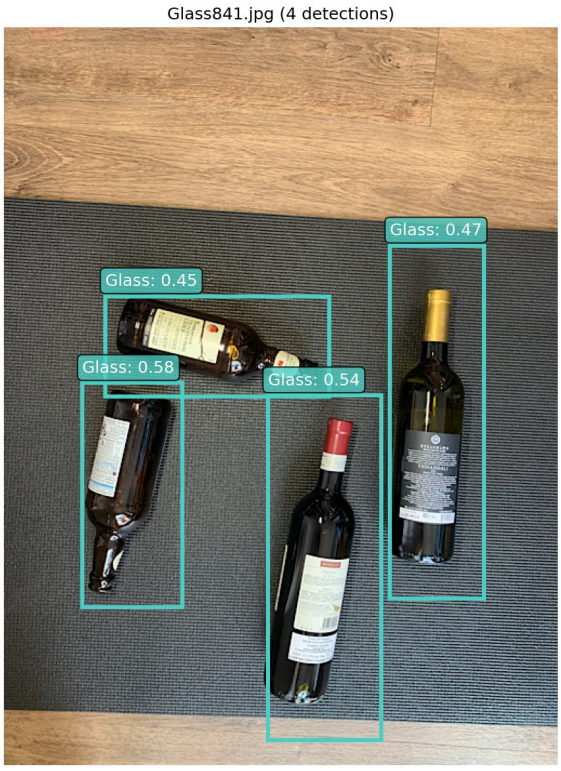
To better understand how the custom ABFP model behaves on real images, we present several qualitative detection examples. Figures 5.12–5.13 show predictions across different object types such as HDPEM, Glass, PET, and AluCan.



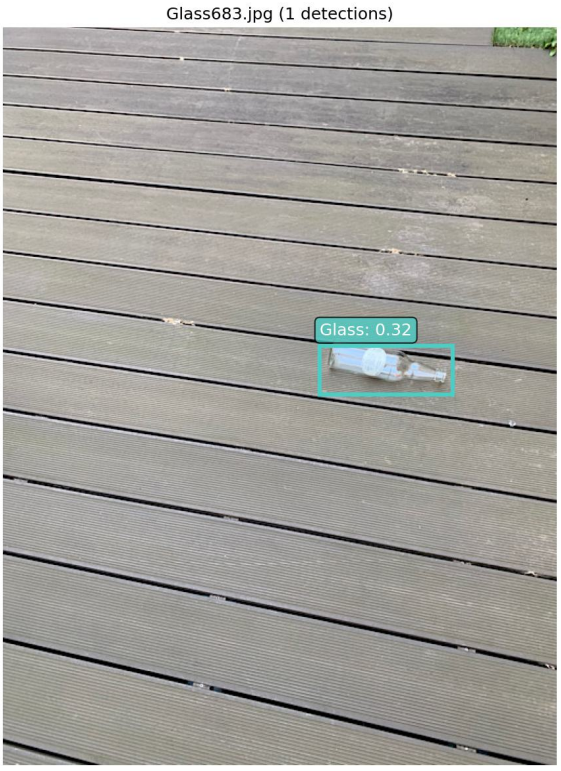
(a) HDPEM detection

(b) Glass detection

Figure 5.12: Qualitative predictions from the ABFP model (Set 1).



(c) AluCan detection



(d) Additional prediction example

Figure 5.13: Qualitative predictions from the ABFP model (Set 2).

5.8 Summary

The custom ABFP model demonstrates moderate performance. It detects a large proportion of objects (high recall) but produces many false positives (lower precision). Confidence curves and confusion matrices highlight these strengths and weaknesses. Qualitative samples show that the model performs reliably in simple scenes but struggles in complex backgrounds.

These insights help identify improvement directions such as unfreezing the backbone, adjusting loss terms, or expanding the dataset.

Chapter 6
Model Comparison

This chapter compares the performance of three models used in this project:

- the **Custom ABFP Model** (our main model),
- the **ResNet50 Detector**, and
- the **pretrained YOLOv8 Model**.

The comparison is based on evaluation metrics, confusion matrices, precision–recall behaviour, and qualitative predictions. Since YOLOv8 was not fine-tuned on our dataset, its results serve only as a reference.

6.1 Overall Metric Comparison

Table 6.1 summarises the main performance metrics for all three models. The YOLOv8 model achieves the highest scores across all metrics, followed by the ResNet50 detector. The custom ABFP model achieves much lower precision, showing that it often predicts extra bounding boxes.

Model	**mAP@0.5**	**Precision**	**Recall**
Custom ABFP Model	0.63	0.58	0.73
ResNet50 Detector	High (0.90+)	0.90–1.00	0.85–1.00
YOLOv8 (Pretrained)	0.994	0.991	0.992

Table 6.1: Overall comparison of the three models.

The large difference between the custom model and the two pretrained models shows how important backbone fine-tuning and large datasets are for obtaining high accuracy in object detection.

6.2 Confusion Matrix Comparison

Confusion matrices provide insights into how well each model separates the four classes. Figures 6.1–6.3 display the confusion matrices for the three models.

6.2.1 Custom ABFP Model

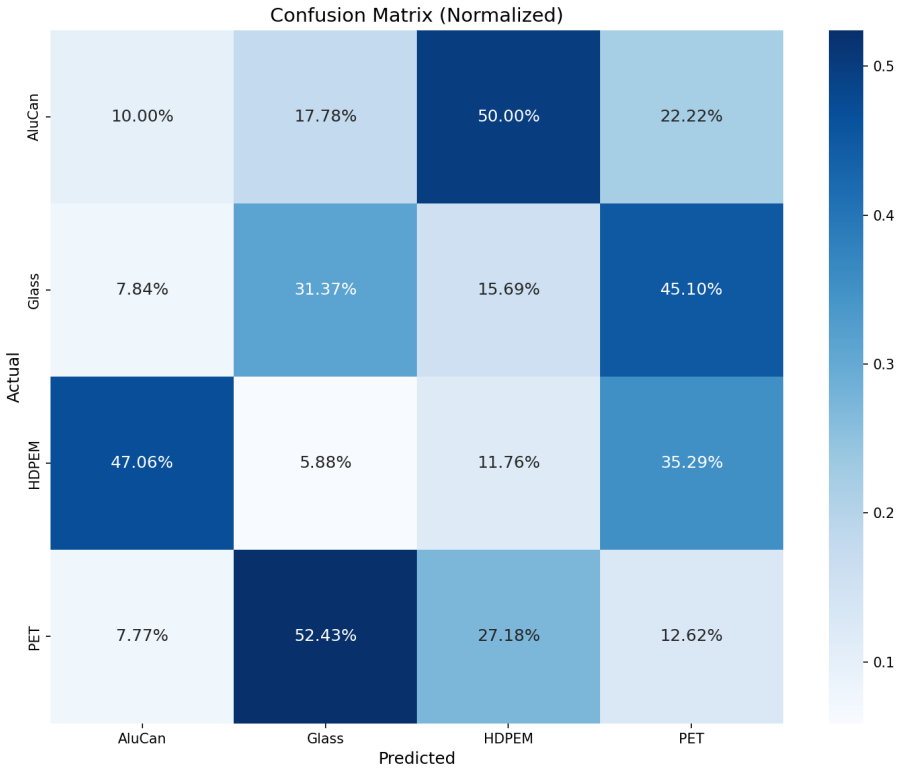


Figure 6.1: Confusion matrix for the custom ABFP model.

The ABFP model shows strong recall but weaker precision. PET and AluCan are the classes with the most confusion, and the model often predicts extra bounding boxes for these categories.

6.2.2 YOLOv8 Model

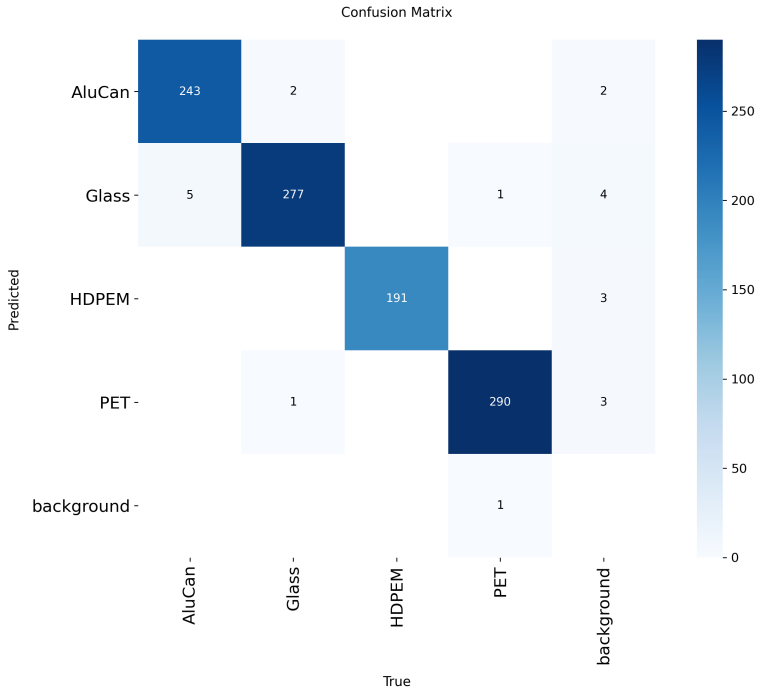


Figure 6.2: Confusion matrix for the pretrained YOLOv8 model (reference).

The YOLOv8 confusion matrix is almost perfectly diagonal, showing very accurate class predictions. This level of accuracy is expected from YOLOv8 due to its COCO pretraining.

6.2.3 ResNet50 Detector

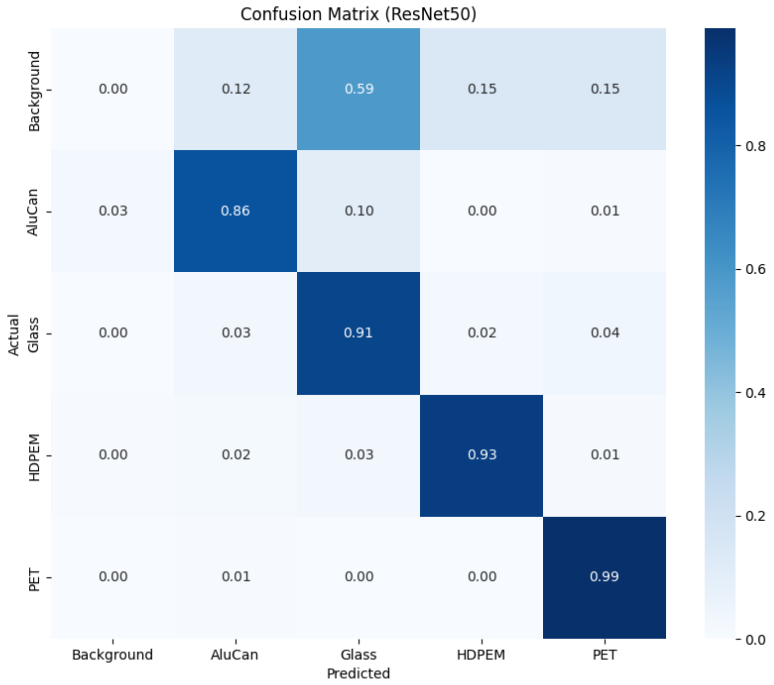


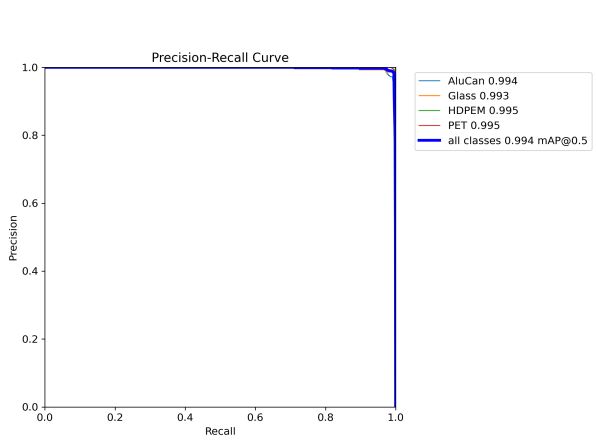
Figure 6.3: Confusion matrix for the ResNet50 detector.

ResNet50 performs very well, correctly separating all four classes with minimal confusion. Its accuracy is significantly higher than the custom ABFP model and closer to the YOLOv8 results.

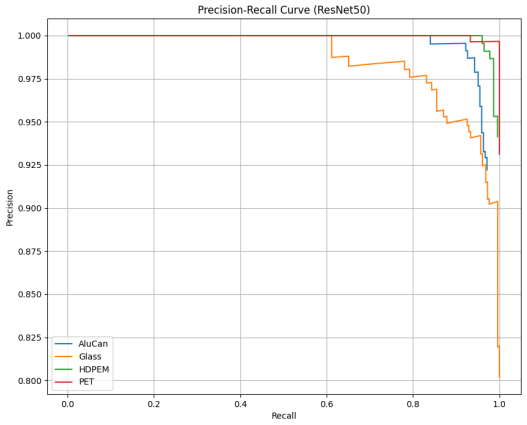
6.3 Precision–Recall Curve Comparison

The precision–recall behaviour further highlights the differences between the models. Figures 6.4a and 6.4b show the PR curves for the ABFP model and ResNet50.

6.3.1 Custom ABFP Model PR Curve



(a) Precision–Recall curve for the custom ABFP model.



(b) Precision–Recall curve for the ResNet50 detector.

Figure 6.4: Precision–Recall curve comparison between the custom ABFP model and the ResNet50 detector.

ResNet50 maintains much higher precision across recall values, showing that it is more reliable at detecting objects without producing extra bounding boxes.

6.4 Qualitative Comparison of Model Predictions

In addition to numerical metrics, it is useful to visually compare model predictions. Figures 6.5a–6.5b show examples from each model.

(a) Example prediction from the custom ABFP model.

(b) Example prediction from the ResNet50 detector.

Figure 6.5: Qualitative comparison of predictions produced by the custom ABFP model and the ResNet50 detector.

ResNet50 generally produces cleaner, more accurate bounding boxes compared to the custom model.

6.5 Discussion

The comparison highlights several important points:

- **YOLOv8 performs the best**: Its results are extremely strong even without fine-tuning, due to the benefits of large-scale COCO pretraining.
- **ResNet50 is a strong detector**: It performs much better than the custom ABFP model, achieving higher precision and better class separation.
- **Custom ABFP Model needs improvement**: While it achieves good recall, the low precision shows that it predicts many additional bounding boxes. This behaviour likely results from freezing the backbone and training with a relatively small dataset.

Overall, the comparison shows that the custom ABFP model works but is not as accurate as established architectures. The results highlight the importance of fine-tuning the backbone, increasing dataset size, and performing ablation studies in future work.

Chapter 7
Conclusion

This project focused on developing a deep learning-based system for detecting waste objects in images. The work involved preparing a labelled dataset, designing a custom detection architecture, training the model, and evaluating its performance using standard object detection metrics. The goal was to explore whether a custom lightweight design, built on top of a frozen pretrained YOLOv8 backbone, could serve as a feasible approach for automated waste detection.

The experiments showed that deep learning models can detect waste objects with reasonable accuracy, even with a relatively small dataset. The custom model, which combined an Adaptive Bidirectional Feature Pyramid (ABFP) neck and decoupled attention-based detection heads, achieved moderate performance. After 50 epochs of training, the model reached an mAP@0.5 of approximately 63.40%, with an overall recall of 72.69% and precision of 57.68%. These results indicate that the model is often able to locate the correct objects, but it also produces some false positives, which reduces overall accuracy.

Because a fully fine-tuned YOLOv8 baseline model was not trained for direct comparison, we cannot conclude that the custom architecture offers improvements over standard YOLOv8 detectors. In practice, YOLOv8 typically performs strongly on small- to medium-sized datasets, and it is likely that a fine-tuned baseline would outperform the custom model. The frozen backbone and the limited dataset size also restricted the model's ability to learn waste-specific visual patterns.

Despite these challenges, the project successfully demonstrates the feasibility of building and experimenting with custom detection components on top of a pretrained model. It also highlights the importance of dataset size, class balance, appropriate loss functions, and training configurations in achieving strong detection performance.

The work serves as a foundation for future research in this area. Potential extensions include collecting a larger and more diverse dataset, unfreezing and fine-tuning the backbone, conducting ablation studies to measure the effect of the ABFP and attention modules, and deploying the model in real-world waste-sorting environments.

Deep learning remains a powerful approach to visual recognition problems, and this project shows how it can be applied to environmental and sustainability applications, even though further work is needed to reach production-level performance.

Bibliography

[1] G. Jocher, A. Chaurasia, and J. Qiu, "YOLOv8: The Next Generation of YOLO," 2023. [Online]. Available: https://docs.ultralytics.com

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *CVPR*, 2016.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.

[4] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934*, 2020.

[5] G. Jocher, "YOLOv5 by Ultralytics," GitHub Repository, 2022. Available: https://github.com/ultralytics/yolov5

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[7] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *CVPR*, 2017.

[8] Z. Zheng et al., "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," in *AAAI*, 2020.

[9] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2014.

[10] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[12] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019.

[13] C. R. Harris et al., "Array Programming with NumPy," *Nature*, 2020.

[14] S. van der Walt et al., "scikit-image: Image Processing in Python," *PeerJ*, vol. 2, p. e453, 2014.

[15] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.