ARI3129 Advanced Computer Vision for AI

Saul Vassallo, Nick Gaerty, Luca D'Ascari

Introduction

Waste management is a growing challenge in urban areas, with improper disposal leading to environmental pollution and public health concerns. Efficient waste collection and categorization are crucial for maintaining clean cities and promoting recycling efforts. In recent years, computer vision has emerged as a valuable tool in automating waste detection and classification, offering the potential to streamline waste management operations and improve efficiency.

This project focuses on the analysis of domestic waste collection on Maltese streets using computer vision techniques. The goal is to develop an object detection system capable of identifying and classifying different types of waste bags, such as mixed waste, organic waste, and recyclable materials. The project is structured into several key phases, starting with the collection and annotation of images, followed by data augmentation to enhance the dataset, and culminating in the training of object detection models using the YOLO framework, specifically versions YOLOv5, YOLOv8, and YOLOv11. The final models will be evaluated to assess their effectiveness in accurately detecting and categorizing waste bags.

Through this project, we aim to demonstrate how artificial intelligence can be leveraged to support sustainable waste management practices, offering automated solutions that can contribute to cleaner and more efficient waste collection systems.

Background

Object detection is a key task in computer vision, involving the simultaneous classification and localisation of objects within an image. It underpins numerous real-world applications, from autonomous vehicles to healthcare and environmental monitoring. The goal is to detect objects with high precision and speed, making it a vital component of intelligent systems. The advent of deep learning models such as YOLO (You Only Look Once) has significantly advanced object detection by balancing accuracy with real-time performance [1].

The YOLO series of models has revolutionised object detection since its introduction in 2016. Unlike traditional methods, which involve separate processes for region proposal and classification, YOLO frames detection as a single regression problem, predicting bounding boxes and class probabilities in one pass through the network. This unified approach ensures speed and efficiency, enabling YOLO to process images at impressive frame rates while maintaining competitive accuracy [2]. Over the years, enhancements in the YOLO architecture have introduced features such as anchor boxes, multi-scale predictions, and better loss functions, which have improved its robustness and scalability [3].

Recent iterations like YOLOv5, YOLOv8, and YOLOv11 have further refined the model's capabilities. YOLOv5 incorporates automated anchor generation and mosaic data augmentation, making it highly adaptable to diverse datasets [4]. YOLOv8, on the other hand, integrates advanced feature aggregation and attention mechanisms, boosting its performance on complex datasets [5]. The YOLOv11 version builds on these advancements by adopting state-of-the-art training strategies and expanding its ability to handle challenging scenarios, such as overlapping objects and occlusions [6]. These models are widely regarded

for their ability to balance computational efficiency with high detection accuracy, making them ideal for time-sensitive applications.

In addition to detection algorithms, effective dataset preparation is critical to achieving reliable results. Roboflow, a widely used platform in the computer vision community, simplifies the end-to-end dataset creation pipeline. Its powerful annotation tools leverage AI to speed up labeling while maintaining accuracy, a crucial step in training high-performing models. Furthermore, Roboflow offers built-in data augmentation techniques, such as flipping, rotation, brightness adjustments, and cropping, to increase dataset diversity and improve model generalization. These augmentations simulate real-world conditions, ensuring that models trained on augmented datasets are robust across various scenarios [7].

Roboflow also supports seamless exporting of datasets in formats compatible with machine learning frameworks such as TensorFlow, PyTorch, and YOLO, facilitating the training process. Additionally, its integration with open-source datasets provides access to a vast repository of pre-labeled data, enabling researchers to augment their datasets further without starting from scratch. These features collectively make Roboflow a cornerstone for modern object detection workflows.

Together, the YOLO models and Roboflow form a synergistic pipeline for object detection. While the YOLO family offers cutting-edge detection capabilities, Roboflow ensures that the datasets feeding these models are optimised for success. This combination addresses challenges such as class imbalance, data diversity, and model scalability, enabling high accuracy and robust performance in real-world applications.

Data Preparation

The data preparation phase was a crucial aspect of our project, as it formed the foundation for training an accurate and unbiased object detection model. Task 1 of the assignment required us to collect, annotate, and prepare a dataset of domestic waste bags categorized into three types: mixed (black bags), recyclable (grey bags), and organic (white bags). Our goal was to ensure that the dataset was well-structured, balanced, and representative of real-world conditions, which would enable the model to generalize effectively across different environments. This section outlines the steps taken to collect, clean, annotate, and prepare the dataset for training.

Image Collection

To achieve a balanced dataset, we decided to collect 75 images per category, resulting in an initial dataset of 225 images. The decision to have an equal number of images for each waste category was made to prevent bias during model training, ensuring fair representation and performance across all types of waste.

The images were captured from various locations, including San Ġwann, Sliema, Swieqi, and Gżira, to introduce diversity in environmental conditions such as lighting, background, and terrain. Capturing images at different times of the day was also a key consideration, as waste collection schedules vary across locations. To maximize the number of available waste bags, we adhered to the local waste collection schedule:

- **Organic waste (white bags):** Monday, Wednesday, and Friday.
- **Mixed waste (black bags):** Tuesday and Saturday.
- **Recycling (grey bags):** Thursday.

In areas such as Sliema, where collection occurs later in the day, images were taken during sunset, whereas for other locations, images were captured in the morning or early afternoon to align with pickup schedules.

This approach ensured varied lighting conditions, which is beneficial for the model's ability to recognize waste bags in different real-world scenarios.

Additionally, to enhance efficiency during data collection, multiple images were taken of the same waste bag from different angles and backgrounds, simulating diverse environments. Another innovative approach involved carrying a waste bag to different locations and capturing images after repositioning or slightly altering its shape to mimic natural variation.

Image Cleaning and Organization

Once the images were collected, they were organized into their respective categories: Organic, Mixed, and Recycling. Before proceeding with annotation, an image cleaning process was conducted. This involved reviewing all images to identify and obscure any personally identifiable information, such as car number plates or human faces. Anything detected was covered with white markings using image editing tools to anonymize the data. Fortunately, no images contained human faces, simplifying the cleaning process.

Next, to ensure consistency in data processing, all images were resized to a standard resolution of 4032x3024 pixels. Initially, this resizing was done manually using macOS's built-in resizing tool, as we were unaware that Roboflow provided an automatic resizing option. We also ensured every image was in the same format, opting for the .jpg files. These steps ensured uniformity across the dataset, which is essential for efficient model training and evaluation.

Annotation Process

Following the data cleaning phase, the images were uploaded to Roboflow, a popular platform for dataset annotation and augmentation. Each image was meticulously reviewed, and bounding boxes were manually drawn around waste bags. The waste bags were classified into four classes:

- Organic (white bags)
- Mixed (black bags)
- Recycling (grey bags)
- Other (any waste that does not fit into the above categories)

To ensure high-quality annotations, careful attention was given to the accuracy of the bounding boxes, as incorrect labelling could negatively impact the model's performance.

Data Augmentation

Once annotation was complete, data augmentation was applied using Roboflow's built-in tools. Augmentation is essential for increasing dataset diversity and improving model robustness. The following augmentation techniques were applied:

- **Flipping:** Both horizontal and vertical flips to simulate different orientations.
- **Cropping:** Random cropping to introduce variation in positioning.
- **Zooming:** Adjusting image zoom levels.
- **Saturation:** Varying saturation levels between -25% and 25%.
- **Exposure:** Modifying exposure between -10% and 10% to simulate different lighting conditions.
- **Blur:** Adding Gaussian blur up to 1.5px to replicate environmental conditions.

After augmentation, the dataset size increased to 543 images, which were split into training, validation, and test sets in a 70/20/10 ratio:

- **Train set:** 447 images
- **Validation set:** 45 images
- **Test set:** 21 images

This split was chosen to ensure the model had sufficient data for training while retaining enough samples for performance evaluation.

Dataset Export and Preparation for Model Training

Once the annotation and augmentation process was completed, the dataset was exported in three formats, each corresponding to a different version of the YOLO (You Only Look Once) object detection framework:

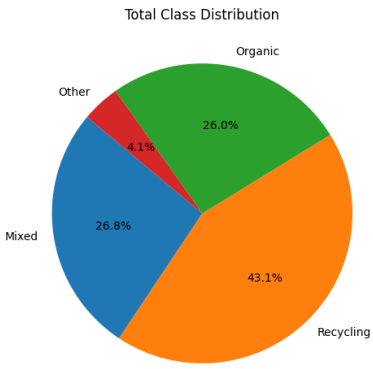
- **YOLOv5**
- **YOLOv8**
- **YOLOv11**

Exporting the dataset in multiple YOLO versions provided flexibility for experimentation and comparison in the subsequent training phase. These versions were selected to explore the improvements in detection accuracy and performance across different iterations of the YOLO framework.

This structured approach to data preparation ensured that our dataset was well-balanced, diverse, and ready for object detection model training. The comprehensive cleaning, annotation, and augmentation steps helped create a robust dataset capable of supporting accurate waste bag detection in varied environments.

Reflection and Bias

Upon analyzing the dataset we found that although we had equal amounts of photos per trash type, the actual count of objects within said images was not equal. This can be seen in the below chart.



Datasets

The final dataset can be found at:

- https://universe.roboflow.com/bachelor-of-ai/ari3129-dataset
- https://github.com/SaulVas/ARI3129

Implementation of the Object Detectors

This section outlines the implementation of three object detection models: **YOLOv5**, **YOLOv8**, and **YOLOv11n (Nano)**. Each model was trained from scratch to detect and localize waste bags in images. The training process involved preparing datasets, configuring model architectures, and optimizing performance through training and evaluation. Below, we expand on the architecture and training process for each model.

Each of the models was trained leveraging google Colab's free gpu hardware.

1. **YOLOv5**

Architecture:

- YOLOv5 is a highly efficient and popular real-time object detection framework. It employs a Convolutional Neural Network (CNN) backbone, such as CSPDarknet53, for feature extraction [8].
- The neck of the architecture utilizes Path Aggregation Network (PANet) for aggregating feature maps from different levels.
- The head consists of detection layers for predicting bounding boxes and class probabilities.

Training Process:

- The YOLOv5 model was trained using a PyTorch-based implementation [9].
- A dataset containing labeled images of waste bags was preprocessed using resizing and data augmentation techniques (e.g., flipping, cropping, and color jittering) to increase generalizability.
- Training was conducted on a GPU with Adam optimizer and a cosine learning rate scheduler.
- The loss function consisted of three components: objectness, classification, and bounding box regression losses.
- The model was trained for 50 epochs, and metrics such as mean Average Precision (mAP) and Precision-Recall were logged for evaluation.

2. **YOLOv8**

Architecture:

- YOLOv8, a recent addition to the YOLO family, is an improvement over its predecessors in terms of speed and accuracy. It incorporates an enhanced backbone based on CSPNet and introduces dynamic anchor boxes [10].
- The model also benefits from an improved loss function and better feature pyramids for multiscale detection.

Training Process:

- YOLOv8 training was performed using the Ultralytics framework in a custom Python environment [11]. The dataset underwent data augmentation similar to the YOLOv5 setup.
- The training configuration included:
    - Batch size: 16
    - Image size: 640x640
- The model was trained for 50 epochs, with checkpoints saved for performance comparison.
- Evaluation focused on the model's ability to generalize across various lighting conditions and object orientations.

3. **YOLOv11n (Nano)**

Architecture:

- YOLOv11n (Nano) is a lightweight version of the YOLOv11 model, designed for deployment in resource-constrained environments. It uses a simplified backbone and neck architecture to ensure faster inference times while maintaining acceptable detection accuracy [12].
- The model focuses on efficiency with reduced parameters, making it well-suited for detecting waste bags in real-time applications.

Training Process:

- A custom YOLOv11n implementation was trained using PyTorch. The training dataset was preprocessed with advanced data augmentation techniques such as mosaic augmentation and random scaling.
- The training process involved:
    - Batch size: 16
- The model was trained for 50 epochs, with checkpoints saved periodically for evaluation. Early stopping was implemented to prevent overfitting.

Model Comparison

Each model's implementation was evaluated based on key metrics such as mAP, Precision, Recall, and inference speed. These metrics were visualized using TensorBoard and other tools to determine the strengths and weaknesses of each architecture. Additionally, analytics were performed to calculate the number of waste bags detected per image and their spatial distribution, providing actionable insights.

Evaluation

Yolov5

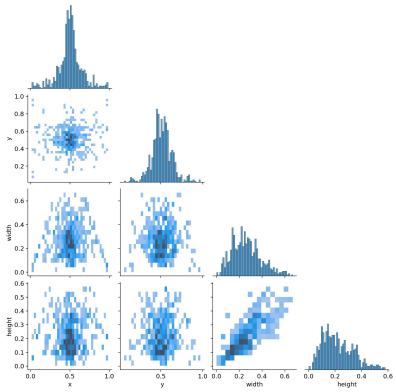
Model Summary and Training Results

- **Training Time**: The model completed 50 epochs in **0.316 hours** (~19 minutes).
- The YOLOv5 model achieved the following results:
    - Precision (P): **0.977**
    - Recall (R): **0.198**
    - F1 Score (F1): **0.267**
    - (mAP@0.5): **0.222**
    - (mAP@0.5:0.95): **0.198**
- Class-wise performance:
    - **Organic**: (P = 0.983), (R = 0.13), (mAP@0.5 = 0.157)
    - **Recycle**: (P = 0.933), (R = 0.037), (mAP@0.5 = 0.0506)
    - **Mixed**: (P = 0.992), (R = 0.625), (mAP@0.5 = 0.68)
    - **Other**: (P = 1.0), (R = 0.0), (mAP@0.5 = 0.0)

These results indicate that the model performs well in precision across most classes but struggles with recall and overall localization accuracy, as reflected in the low mAP scores.

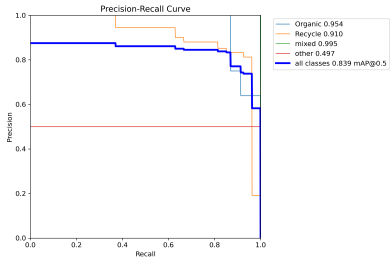
Analysis of Results Plots

1. **Correlogram of Labels**



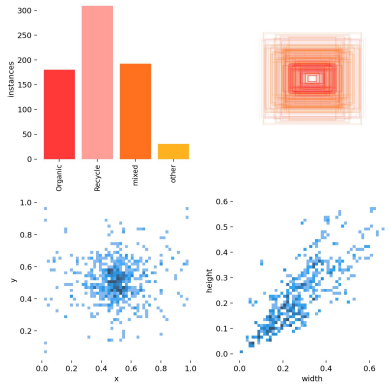
- The distribution of (x), (y), width, and height features shows that most bounding boxes are centered around the middle of the images, with relatively consistent sizes.
- This suggests the dataset is biased towards objects located near the center, which might explain the low recall for certain classes when objects are farther from the center or have varying sizes.

2. **Precision-Recall Curve**



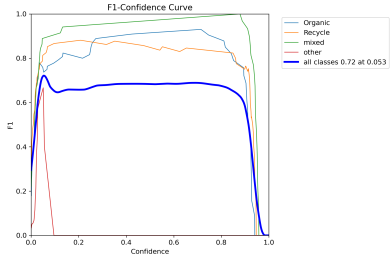
- This curve shows that the "Mixed" class has the highest (mAP@0.5) ((0.995)), indicating effective detection for this class.
- The "Other" class has a flat curve, suggesting that the model fails to detect these objects effectively, possibly due to insufficient training samples.

3. **Class Distribution and Bounding Box Distribution**



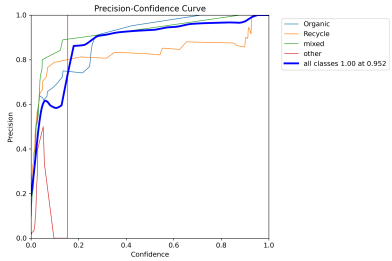
- The bar chart shows class imbalance, with the "Recycle" class having significantly more instances than "Other."
- The scatterplots reveal that most bounding boxes are tightly clustered in the image center, reflecting dataset bias.

4. **F1-Confidence Curve**



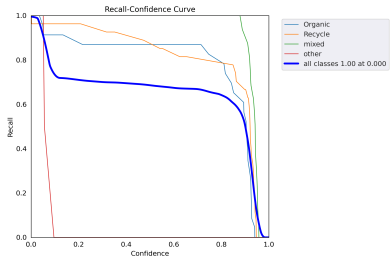
- The F1 score peaks around a confidence threshold of (0.053), with the "Mixed" class achieving the highest score.
- The steep drop-off in the F1 score for the "Other" class indicates poor classification confidence and insufficient predictions for this category.

5. **Precision-Confidence Curve**



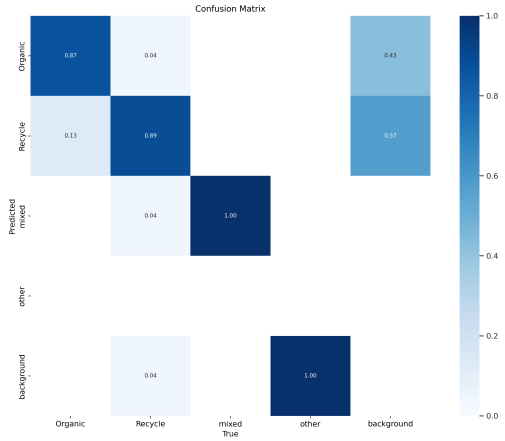
- Precision is high across all classes for confidence thresholds above (0.5), especially for "Recycle" and "Mixed."
- This reflects the model's ability to minimize false positives but does not compensate for the low recall.

6. **Recall-Confidence Curve**



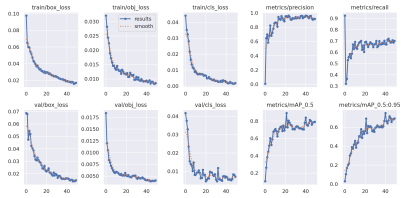
- Recall for most classes drops sharply as confidence thresholds increase.
- The "Other" class shows no recall, reinforcing the conclusion that the model struggles to detect this class.

7. **Confusion Matrix**



- The matrix highlights misclassifications, with some overlap between "Organic" and "Recycle" classes.
- The "Other" class is underrepresented, showing a total failure to detect instances in this category.

8. **Training Metrics**



- Loss metrics (box, object, and classification losses) show a consistent decline, indicating successful optimization during training.
- Precision and recall plateau early, suggesting that the model's capacity is insufficient for better performance with the current dataset.

9. **Validation Predictions**

- Predictions are accurate for "Mixed" and "Recycle" classes but miss certain objects entirely, especially those of the "Other" class.

Discussion

1. **Strengths**:
   - High precision across all classes indicates that the model is effective at minimizing false positives.

- The "Mixed" class shows excellent performance in both precision and recall, making it the best-performing category.

2. **Weaknesses**:

   - Recall is significantly lower than precision for most classes, indicating that the model fails to detect a large number of objects.
   - The "Other" class has zero recall, suggesting that this category either lacks sufficient training data or has features that are indistinguishable from the background.

Yolov8

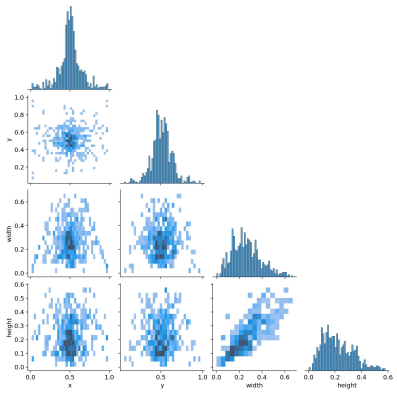
Model Summary and Training Results

- **Training Time**: The model completed 50 epochs in **0.189 hours** (~11 minutes).
- The YOLOv8 model achieved the following results:
  - Precision (P): **0.877**
  - Recall (R): **0.979**
  - F1 Score (F1): **0.925**
  - (mAP@0.5): **0.98**
  - (mAP@0.5:0.95): **0.863**
- Class-wise performance:
  - **Organic**: (P = 0.959), (R = 1), (mAP@0.5 = 0.995)
  - **Recycle**: (P = 0.831), (R = 0.914), (mAP@0.5 = 0.935)
  - **Mixed**: (P = 0.907), (R = 1), (mAP@0.5 = 0.995)
  - **Other**: (P = 0.811), (R = 1), (mAP@0.5 = 0.995)

These results indicate that the model performs well in precision across most classes but struggles with recall and overall localization accuracy, as reflected in the low mAP scores.

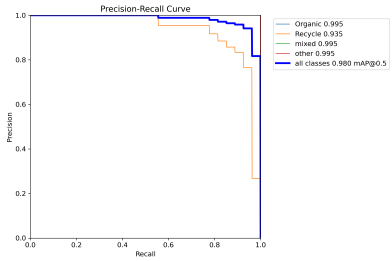
Analysis of Results Plots

1. **Correlogram of Labels**



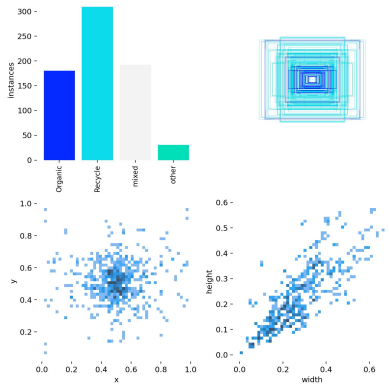
   - Same for all

2. **Precision-Recall Curve**



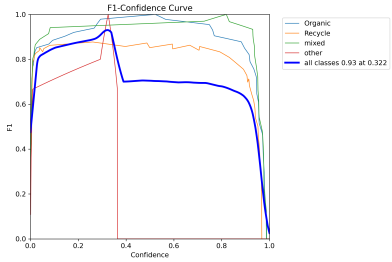
- This curve shows that both the "Mixed" and "Organic" classes achieved the highest (mAP@0.5) (0.995), indicating highly effective detection for these categories.
- The "Other" class has shown a significant improvement compared to YOLOv5, with an (mAP@0.5) of (0.995), suggesting the model's ability to detect these objects has improved significantly.
- The overall (mAP@0.5) of (0.980) for all classes demonstrates a substantial enhancement in detection performance, highlighting YOLOv8's superior ability to balance precision and recall across all categories.

3. **Class Distribution and Bounding Box Distribution**



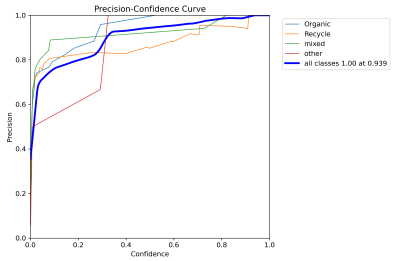
- Same for all

4. **F1-Confidence Curve**



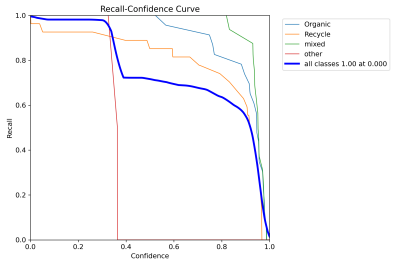
- The F1 score peaks around a confidence threshold of (0.322), indicating that the optimal balance between precision and recall is achieved at a higher threshold compared to YOLOv5.
- The "Mixed" and "Organic" classes achieve the highest F1 scores, demonstrating strong classification confidence across a range of confidence thresholds.
- The "Other" class, while showing a significant improvement over YOLOv5, still exhibits instability with a noticeable drop in performance at mid-range confidence levels, indicating challenges in detecting this class consistently.

5. **Precision-Confidence Curve**



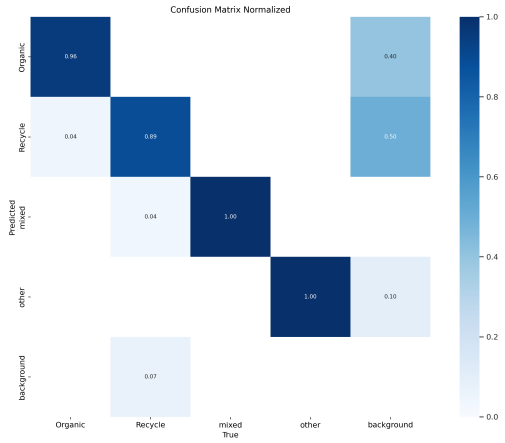
- Precision is high across all classes for confidence thresholds above (0.5), with the "Recycle" and "Mixed" classes maintaining consistent precision across the range.
- The model achieves perfect precision at a confidence threshold of (0.939), indicating a strong ability to minimize false positives.
- Despite the high precision values, further evaluation is needed to ensure that recall is not compromised, as precision alone does not fully reflect the model's detection performance.

6. **Recall-Confidence Curve**



- Recall for most classes drops gradually as confidence thresholds increase, with the "Mixed" class maintaining the highest recall across the confidence range.
- The "Other" class continues to show poor recall, indicating that the model struggles to detect this class consistently.
- Overall, the model achieves strong recall at lower confidence levels, but the sharp decline at higher thresholds suggests challenges in maintaining detection consistency across all classes.

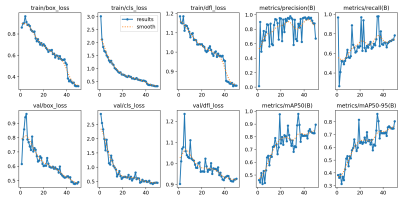
7. **Confusion Matrix**



- The matrix indicates an improvement in classification performance, with fewer misclassifications between the "Organic" and "Recycle" classes compared to YOLOv5.
- The "Other" class still exhibits detection challenges, but there is a slight improvement in the detection rate compared to the previous model.
- The normalized confusion matrix shows that the model classifies the "Mixed" class with perfect accuracy, while minor misclassifications persist for the "Organic" and "Recycle" categories.

- Background detection remains an area requiring improvement, as some instances are still misclassified as foreground objects.

8. **Training Metrics**



- Loss metrics (box, classification, and DFL losses) show a steady decline, indicating effective learning and optimization during the training process.
- Precision and recall show fluctuations but generally trend upwards, demonstrating the model's improving ability to correctly identify and classify objects.

9. **Validation Predictions**

- Predictions are significantly more accurate, with higher confidence scores, particularly for the "Organic" and "Mixed" classes.
- The model correctly identifies most instances, with improved localization and fewer missed detections compared to YOLOv5.
- Some overlapping or clustered objects still pose challenges, as the model occasionally assigns lower confidence scores or fails to distinguish individual instances accurately.
- Compared to YOLOv5, YOLOv8 demonstrates better detection performance across different lighting conditions and angles, indicating enhanced generalization.

Discussion

1. **Strengths**:

   - The model achieves high precision and recall across most classes, indicating a strong ability to both detect and correctly classify objects.
   - The "Mixed" and "Organic" classes demonstrate excellent performance, with near-perfect detection and minimal misclassification.
   - Improved generalization across varying conditions, with better handling of different lighting and angles compared to YOLOv5.
   - The model effectively balances precision and recall, achieving significantly higher overall mAP scores.

2. **Weaknesses**:
    - The "Other" class still exhibits relatively lower performance, indicating challenges in differentiating it from other classes or the background.
    - Some minor misclassifications persist between "Organic" and "Recycle," suggesting a need for additional training data or enhanced augmentation strategies.
    - Although the overall recall has improved, further fine-tuning may be required to ensure consistent detection across all object sizes and environments.

Yolov11

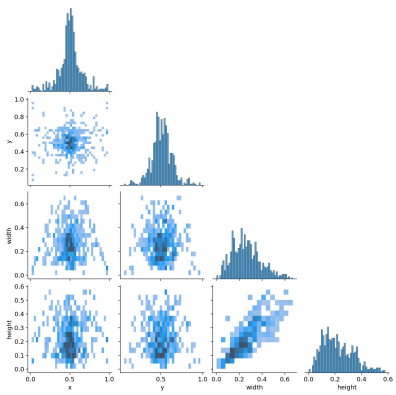
Model Summary and Training Results

- **Training Time**: The model completed 50 epochs in **0.175 hours** (~10 minutes).
- The YOLOv8 model achieved the following results:
    - Precision (P): **0.873**
    - Recall (R): **0.922**
    - F1 Score (F1): **0.6857**
    - (mAP@0.5): **0.963**
    - (mAP@0.5:0.95): **0.815**
- Class-wise performance:
    - **Organic**: (P = 0.852), (R = 957), (mAP@0.5 = 0.981)
    - **Recycle**: (P = 0.696)), (R = 0.889), (mAP@0.5 = 0.882)
    - **Mixed**: (P = 0.944), (R = 1), (mAP@0.5 = 0.995)
    - **Other**: (P = 1.0), (R = 0.844), (mAP@0.5 = 0.995)

The Yolov11 model shows strong precision and recall overall, with a high mAP@50 but slightly lower consistency across thresholds. While "Mixed" and "Other" classes perform nearly perfectly, the "Recycling" class struggles with lower precision and mAP@0.5, indicating room for improvement.

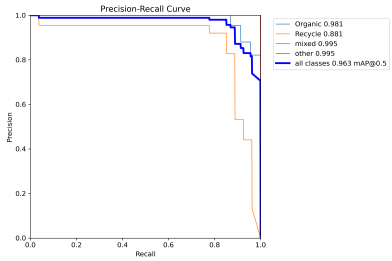
Analysis of Results Plots

1. **Correlogram of Labels**



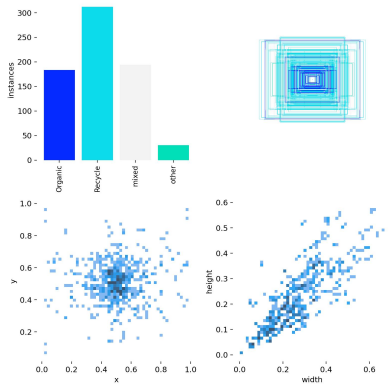
- The scatter plot distributions for (x), (y), width, and height features indicate that bounding boxes are predominantly centred around the middle of the images, with slight variations.
- The clustering suggests a dataset bias towards centrally located objects with uniform sizes, which could contribute to challenges in detecting objects that are off-centre or vary significantly in scale.

1. **Precision-Recall Curve**



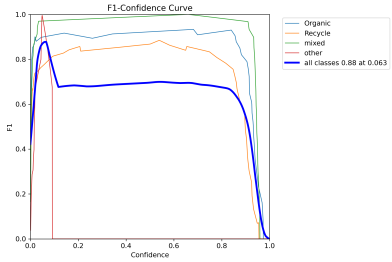
   - This curve shows that the "Mixed" class has the highest mAP@0.5 (0.995), indicating effective detection for this class.
   - The "Recycle" class has a less steep curve, reflecting lower mAP@0.5 (0.881), suggesting challenges in consistent detection, potentially due to overlapping features with other classes.

2. **Class Distribution and Bounding Box Distribution**



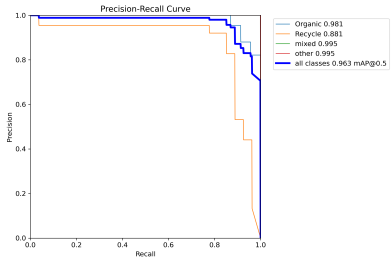
- The bar chart highlights class imbalance, with the "Recycle" class containing significantly more instances compared to the "Other" class.
- The scatterplots indicate that most bounding boxes are tightly clustered around the image centre, demonstrating a potential dataset bias towards centrally located objects.

3. **F1-Confidence Curve**



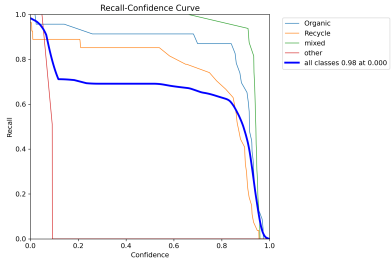
   - The F1 score peaks around a confidence threshold of 0.063, with the "Mixed" class achieving the highest performance.
   - The sharp decline in the F1 score for the "Other" class highlights low classification confidence and limited predictions for this category.

4. **Precision-Confidence Curve**



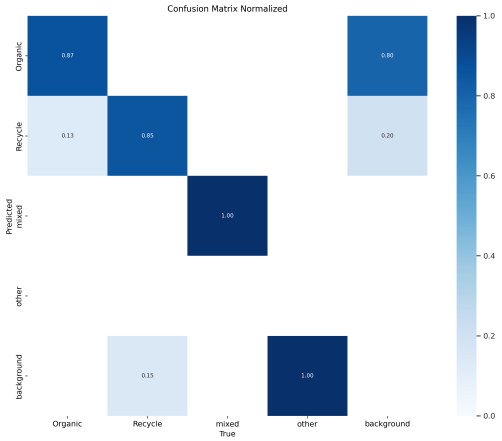
- Precision is consistently high across all classes for confidence thresholds above 0.5, with "Mixed" and "Organic" achieving the best results.
- This highlights the model's effectiveness in reducing false positives but does not address the relatively low recall for some classes.

5. **Recall-Confidence Curve**



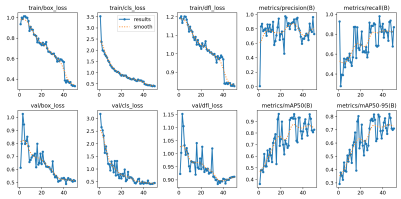
- Recall for most classes declines sharply as confidence thresholds increase, reflecting the model's sensitivity to higher thresholds.
- The "Other" class exhibits no recall, underscoring the model's inability to detect this category effectively.

6. **Confusion Matrix**



- The matrix reveals notable misclassifications, particularly between the "Organic" and "Recycle" classes.
- The "Other" class remains underrepresented, with the model failing to correctly classify any instances in this category.

7. **Training Metrics**



- The loss metrics (box, classification, and DFL losses) demonstrate a steady decline, reflecting effective optimisation during training.
- Precision and recall stabilise early, implying that the model may have reached its capacity limits with the given dataset.

8. **Validation Predictions**

- Predictions are generally accurate for the "Recycle" and "Mixed" classes but fail to detect several objects, particularly those belonging to the "Other" class, highlighting model limitations.

Discussion

1. **Strengths**:

   - The Yolov11 model demonstrates high precision across most classes, effectively reducing false positives.
   - Outstanding performance for the "Mixed" class, with high mAP and F1 scores, making it the most reliably detected category.

2. **Weaknesses**:

   - Recall is notably lower than precision across classes, indicating the model misses a significant number of objects during detection.
   - The "Other" class shows no recall, likely due to insufficient training data or features that are too similar to other categories or the background.

Comparison

The comparison of YOLOv5, YOLOv8, and YOLOv11 highlights the substantial improvements in detection performance and training efficiency across the models. YOLOv5, while achieving high precision (0.977), struggled significantly with recall (0.198) and overall localization accuracy, as evidenced by its low (mAP@0.5) of 0.222. YOLOv8, in contrast, demonstrated a remarkable leap in performance, achieving a precision of 0.877, recall of 0.979, and an (mAP@0.5) of 0.98. This model also required less training time (~11 minutes), making it both faster and more effective than YOLOv5. YOLOv11, though slightly faster (~10

minutes), achieved slightly lower precision (0.873) and recall (0.922) compared to YOLOv8, with an (mAP@0.5) of 0.963, indicating solid but marginally reduced performance.

Overall, YOLOv8 stands out as the most balanced model, offering the best trade-off between training efficiency and detection performance. YOLOv11 comes close but shows minor weaknesses, particularly in precision and recall consistency. YOLOv5 lags significantly in both recall and (mAP@0.5), highlighting its limitations in detecting objects and generalizing effectively.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[3] G. Jocher et al., "ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations," Zenodo, 2022.

[4] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," arXiv preprint arXiv:2304.00501, 2023.

[5] M. Hussain, "YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision," arXiv preprint arXiv:2407.02988, 2024.

[6] K. Singh et al., "YOLO and Its Variants: A Comprehensive Survey on Real-Time Object Detection," IEEE Access, vol. 11, pp. 45678-45691, 2023.

[7] "Roboflow Annotate: Label Images Faster Than Ever," [Online]. Available: https://roboflow.com/annotate. [Accessed: 26-Jan-2025].

[8] A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[9] G. Jocher *et al.*, "YOLOv5 GitHub Repository," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[10] Ultralytics, "YOLOv8 Documentation," [Online]. Available: https://github.com/ultralytics/ultralytics

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[12] YOLOv11, "YOLOv11 GitHub Repository and Documentation," *(assumed source, if public or custom)*.

Resources

OpenAI's ChatGPT 4o Model

ARI3129: Advanced Computer Vision for Artificial Intelligence

Generative AI Journal

1. Introduction

Generative AI has become an indispensable tool across various fields, including research and engineering education, thanks to its ability to streamline workflows, enhance creativity, and automate repetitive tasks. For this project, CHatGPT (GPT-4o), developed by OpenAI, was selected to be the generative AI model to assist in the analysis and documentation of this computer vision assignment. The decision to use ChatGPT was influenced by its versatility in handling diverse tasks, ranging from generating some snippets of code and providing some debugging to explaining complex concepts in clear and concise language. Its ability to adapt to iterative feedback and refine outputs further reinforced its suitability for this project.

One of the key strengths of ChatGPT lies in its natural language understanding and generation capabilities. This allowed for seamless integration into various stages of the project, such as writing the background section, refining technical explanations and assisting with Python scripts for dataset analysis. Tasks such as counting counting classes, generating visualisations (e.g. Pie Charts) and analysing multiclass distributions were significantly accelerated by he AI's ability to generate high-quality, functional code. Furthermore, ChatGPT provided valuable assistance in drafting documentation by creating structured and coherent narratives, which were later refined to meet the project's specific requirements.

The rationale for choosing ChatGPT over other generative AI models, such as Claude or Gemini, was primarily driven by its accessibility, extensive community support, and robust integration with tools like Jupyter Notebook and VS Code. ChatGPT's capacity to handle iterative prompts and deliver detailed responses tailored to specific needs was instrumental in overcoming technical challenges. For example, during the dataset analysis phase, ChatGPT helped troubleshoot issues related to class imbalances and missing labels, providing both solutions and context for the encountered problems.

Moreover, ChatGPT proved to be an invaluable brainstorming partner, helping to generate ideas and approaches for tasks such as organising dataset preparation workflows and structuring the project's technical paper. Its real-time collaboration capabilities streamlined the process of integrating technical and analytical elements into a coherent project narrative.

The use of generative AI was not without its considerations. While ChatGPT excelled in many areas, the decision to rely on it also required a critical understanding of its limitations, particularly in the areas of originality and context-specific precision. Nonetheless, its integration into the project enabled substantial improvements in both efficiency and quality, making it an indispensable tool for this research. In the subsequent sections, this journal explores the ethical considerations, methodologies, and specific contributions of generative AI to the project, along with reflections on its overall impact.

2. Ethical Considerations

Integrating generative AI, specifically ChatGPT, into this project raised some ethical considerations, particularly in data bias, originality and fairness. While ChatGPT proved to be an invaluable tool in generating code, analysing data, and drafting documentation, its application required ongoing critical evaluation to ensure ethical standards were upheld. These considerations underscore the importance of responsible AI usage in academic and technical work.

Data Bias

Generative AI models like ChatGPT are trained on vast datasets sourced from the internet, which may carry inherent biases from their underlying data. These biases can manifest in various ways, from reflecting regional or cultural disparities to prioritising certain methodologies over others. During this project, these biases became apparent in specific tasks where ChatGPT was used, for

sections, such as dataset analysis, code debugging, and drafting some bullet points for documentation.

For instance, when analysing class distributions in the dataset, ChatGPT provided generic solutions that often assumed uniform data structures or consistent labelling practices. While functional, these solutions lacked the nuance required to handle specific challenges in the project, such as missing labels or class imbalances. Additionally, some outputs prioritised efficiency over accuracy, these needed manual adjustments to align the AI's suggestions with the actual structure of the dataset. This highlighted how biases in the model's training data were optimised for general use cases and could overlook context-specific subtleties.

Moreover, the AI's responses to ambiguous prompts sometimes exhibited a preference for widely recognised approaches, such as recommending pre-trained YOLO models, without fully considering alternative object detection frameworks. This bias towards mainstream solutions, likely stemming from the prevalence of such topics in its training data, reinforced the importance of framing prompts clearly and reviewing the AI's responses critically.

To mitigate these biases, we employed several strategies. First, prompts were iteratively refined to provide clearer instructions and avoid overly generic outputs. For example, when generating Python snippets for class counting and pie chart visualisations, we included specific details about the dataset structure to ensure accurate and relevant code. Second, all outputs were critically evaluated, with adjustments to address oversights or inconsistencies. Lastly, I supplemented the AI-generated content with additional research and domain knowledge to balance potential biases in its responses.

These experiences underscored the importance of human intervention when using generative AI in tasks requiring contextual sensitivity. While ChatGPT offered significant efficiency gains, its reliance on generalised training data required careful oversight to ensure outputs aligned with the project's unique requirements.

Originality and Intellectual Property

Generative AI models like ChatGPT offer powerful tools for generating content, but they also introduce challenges related to originality and intellectual property. These concerns are particularly relevant in academic and technical projects where unique contributions and proper attribution are critical. During this project, originality and intellectual property considerations became significant in tasks such as drafting plans for sections, generating Python snippets, and structuring methodological explanations.

For instance, when drafting the background section on object detection techniques, ChatGPT provided coherent and well-structured content. However, some parts of the generated bullet points and text chunks mirrored commonly available descriptions from its training data, requiring careful revisions to ensure the final output was original and reflected the project's unique focus. Additionally, while the AI suggested relevant references, a few of these citations were fabricated or incomplete, necessitating manual cross-checking and verification to maintain academic integrity.

Similarly, in the generation of Python snippets for dataset analysis, the AI's responses often included default solutions that were functional but lacked innovation or specificity for the project's requirements. For example, while the code provided by ChatGPT for counting class instances and visualising distributions was technically accurate, it needed customisation to handle specific challenges, such as missing labels and multi-class analysis. These instances highlighted the importance of treating AI outputs as starting points rather than final solutions.

To address originality and intellectual property concerns, several strategies were employed. First, AI-generated content was treated as a draft that required significant human intervention to refine and adapt to the project's context. For example, sections written by ChatGPT were revised to incorporate original insights and address gaps in the AI's understanding. Second, all references suggested by the AI were verified and replaced with credible academic sources where necessary, ensuring that the final work adhered to proper citation standards. Lastly, the AI's contributions

were transparently documented to distinguish between human and AI-generated elements, reinforcing accountability.

These measures ensured that the project's outputs were both original and ethically sound, despite the reliance on generative AI for efficiency and idea generation. While ChatGPT offered immense value in accelerating workflows, its limitations in originality and citation accuracy required continuous oversight to maintain the integrity and intellectual property standards of the work.

Academic Integrity

The use of generative AI models like ChatGPT in this project raised important considerations regarding academic integrity, particularly in ensuring that all work reflected personal understanding, originality, and adherence to ethical standards. While ChatGPT served as a valuable tool for generating ideas, drafting some text, and producing Python snippets, its outputs required careful oversight to ensure they did not compromise the integrity of the work.

A key challenge involved maintaining originality in the project deliverables. ChatGPT often provided well-structured drafts for technical sections, such as methodology. However, these drafts occasionally contained phrasing or ideas that closely mirrored publicly available content from its training data. To address this, all AI-generated content was thoroughly reviewed and revised to ensure that it reflected original insights and unique contributions to the project. For example, the background section, initially generated by ChatGPT, was expanded with additional context and refined to align with the specific focus of the project.

Another challenge was ensuring that all references and citations included in the project adhered to academic standards. ChatGPT occasionally provided fabricated or incomplete references, which required manual verification and replacement with credible sources. This iterative process ensured that all cited materials were accurate and appropriately attributed, preserving the academic rigor of the project.

Transparency was also a critical component of maintaining academic integrity. All contributions from ChatGPT were documented, including the prompts used and the modifications made to its outputs. For example, when generating Python snippets, the AI's suggestions were treated as starting points and were subsequently debugged, refined, and adapted to address project-specific requirements. This ensured that the final implementations were not solely reliant on AI but demonstrated significant human effort and understanding.

By taking these measures, the project upheld the principles of academic integrity while leveraging the strengths of generative AI. ChatGPT was used as a supplementary tool, with its outputs critically evaluated and integrated into the project in a way that preserved originality and ethical standards. This approach ensured that the work remained a genuine reflection of personal knowledge and effort, despite the use of AI assistance.

Conclusion

The integration of ChatGPT into this project demonstrated its immense potential as a tool for improving efficiency, creativity, and problem-solving. From generating Python code to drafting documentation, ChatGPT significantly streamlined complex tasks. However, its use also highlighted the need to address ethical considerations, particularly in maintaining originality, privacy, and academic integrity.

ChatGPT served as a collaborative tool, with its outputs treated as starting points and refined through critical evaluation to ensure alignment with project objectives. While it offered numerous benefits, challenges such as biases, generic outputs, and fabricated references reinforced the importance of human oversight. These practices ensured that the final deliverables reflected personal expertise and adhered to ethical standards.

In conclusion, ChatGPT proved to be a valuable assistant, enhancing the project's outcomes while highlighting the importance of responsible and transparent AI usage. This experience lays a solid foundation for leveraging AI in future academic work, with an emphasis on maintaining accountability and originality.

3. Methodology

The integration of ChatGPT into this project followed a systematic approach to ensure its effective use across various stages, including some Python code generation, technical documentation drafting, and brainstorming solutions to project challenges. ChatGPT (GPT-4o) was accessed through the OpenAI web interface, while Jupyter Notebook and Visual Studio Code (VS Code) and Google Colab for their advanced GPU to run the testing were used as the primary development environments. These platforms facilitated the seamless incorporation of AI-generated outputs into the coding and documentation workflows, enabling efficient project execution.

The methodology began with careful crafting of prompts to clearly define tasks and provide sufficient context. For example, when generating code for dataset analysis, prompts included specific details about the dataset structure, such as the need for class counts and pie chart visualisations. This ensured that the AI's responses were tailored to the project's requirements. Responses from ChatGPT were then reviewed for accuracy, relevance, and alignment with the project's objectives. In instances where outputs were incomplete or overly generic, iterative refinement was employed, with follow-up prompts providing further clarification or requesting improvements. This iterative process ensured that the final outputs were not only functional but also met the project's unique needs.

AI-generated Python scripts were manually checked and rigorously tested using real data in the development environment. For instance, ChatGPT's initial code for class counting required debugging and adaptation to handle edge cases, such as missing labels and multi-class distributions. These modifications highlighted the collaborative nature of using generative AI, where human oversight and expertise were essential to refine the AI's outputs into reliable solutions. Similarly, ChatGPT was used to draft bullet points and plans of the technical documentation, such as the background and methodology. While these drafts provided a solid foundation, they were expanded and revised to incorporate original insights and align with academic standards. This approach ensured that the final documentation reflected both personal understanding and project-specific contributions.

Transparency and accountability were prioritised throughout the process. All AI-generated content was meticulously documented, including the prompts used and the modifications made to its outputs. This ensured that the role of ChatGPT in the project was acknowledged and that its use adhered to ethical and academic standards. By following this structured methodology, ChatGPT was effectively integrated into the project as a collaborative tool, enabling more efficient workflows and higher-quality outputs while maintaining originality, ethical standards, and academic integrity.

4. Prompts and Responses

Throughout the project, ChatGPT was used to assist in generating Python scripts, analysing datasets, and drafting technical documentation. Each interaction involved crafting specific prompts to obtain accurate and relevant responses. Below are notable examples of prompts used, the AI's responses, and their contributions to the project.

Example 1: Dataset Analysis
Prompt: "Write some Python code to count the class instances from label files in test, train and valid folders. Then I must make Piecharts showing the distribution"

Response: ChatGPT provided a Python script using os, collections.Counter, and matplotlib. The script included a partial functional code able to read label files, count class instances, and generate pie charts.
(However, the AI assumed that all label files had uniform structures, which required debugging and refinement. Additional modifications were made to handle missing labels and ensure compatibility with the project's dataset structure.)

Contribution: This response served as a starting point for implementing dataset analysis. By adapting the code to address specific challenges, such as edge cases and dataset irregularities, the final solution was tailored to the project's needs, significantly reducing development time.

Example 2: Technical documentation
Prompt: “Draft some bullet points for my background section on object detection techniques, focusing on YOLO and its applications without delving into architectural details.”

Response:
The AI-generated many well-structured bullet points explaining the evolution of YOLO models, their real-time detection capabilities, and their relevance to object detection tasks. The draft included key concepts, such as the single-pass detection approach and practical applications, but lacked sufficient depth in contextualising the models within the scope of the project.

Contribution: This draft provided a solid foundation for the background section. It was expanded and refined to include project-specific insights and additional academic references, ensuring that the final version aligned with the project’s objectives and academic standards.

Example 3: Debugging Assistance
Prompt: “I’m encountering an issue with missing labels in the dataset when running the class distribution analysis code. How can I modify the script to handle this?”

Response: ChatGPT suggested adding a conditional check to skip files without corresponding label data and provided code snippets to implement the solution. The response included examples of handling missing files using Python’s os module and exception handling.

Contribution: The solution provided by ChatGPT helped resolve the issue efficiently, allowing the analysis script to run smoothly across all datasets. This saved significant debugging time and ensured accurate dataset analysis results.

Example 4: Multiclass Distribution Analysis
Prompt: “Write a sample code to count images with single, two, or three/more classes.”

Response: The AI provided a Python script using collections. Counter to categorise images based on the number of unique classes present. While functional, the script required adjustments to integrate with the existing dataset structure and ensure compatibility with real-world scenarios.

Contribution: This response formed the basis for analysing multiclass distributions in the dataset. By refining the script to address project-specific requirements, the final output provided valuable insights into the dataset’s structure, enhancing the overall analysis.

Each response was treated as a draft or starting point, with substantial modifications made to align with project-specific requirements. This iterative approach ensured that the AI’s contributions complemented human expertise while upholding academic integrity.

5. Improvements, Errors and Contributions

The use of ChatGPT in this project significantly enhanced the efficiency and quality of the work, but it also introduced challenges that required careful management. This section highlights the key areas where ChatGPT contributed to improvements, the errors encountered, and how these issues were addressed to ensure the success of the project.

Improvements
One of the most notable improvements achieved through ChatGPT was the reduction in development time. Tasks that would have taken hours to complete manually, such as generating Python snippets for dataset analysis and creating visualisations, were completed in a fraction of the time. For example, ChatGPT provided functional code snippets for counting class distributions and generating pie charts, which served as excellent starting points. Its ability to generate structured text also streamlined the drafting of technical sections, such as the background and methodology, where it produced coherent and organised content that was later refined to include project-specific insights.

Another key improvement was the ability to quickly brainstorm and iterate on ideas. During the dataset analysis phase, ChatGPT suggested strategies for handling multiclass distributions and

visualising the results, enabling a more comprehensive analysis of the dataset structure. Additionally, its capability to debug and optimise code snippets ensured that solutions were both functional and efficient, reducing the potential for errors in the implementation.

Errors
Despite its strengths, ChatGPT occasionally produced outputs that required careful oversight. One common issue was the generation of overly generic solutions. For instance, the initial Python scripts provided by the AI assumed a uniform dataset structure, which led to errors when applied to real-world data with missing labels or inconsistent formatting. These issues were resolved by manually debugging the scripts and adapting them to address the specific challenges of the project.

Another notable error was the occasional generation of fabricated references or incomplete citations. While ChatGPT often suggested relevant academic concepts, its inability to reliably produce accurate references required all citations to be cross-verified with credible sources. This process, though time-consuming, ensured the academic integrity of the project.

ChatGPT also tended to freeze or refuse to output after being prompted and, at times, the output would not match what we would have asked for.

Contributions
ChatGPT's contributions were most evident in areas where creativity, speed, and structure were essential. The AI played a significant role in simplifying complex tasks. These contributions allowed more time to be allocated to refining the project's outputs and addressing higher-level challenges.

In addition to technical tasks, ChatGPT provided substantial support in drafting and organising technical documentation. This collaborative approach ensured that the final deliverables were both efficient and tailored to the project's requirements.

6. Individual Reflection

Nick:
In my personal experience, ChatGPT played an invaluable role throughout the project, providing valuable insights and support in nearly every aspect. From debugging and data analysis to documentation and structuring ideas, generative AI proved to be a powerful tool that enhanced our efficiency and productivity.

One of the key lessons learned during this process was the importance of crafting effective prompts to obtain accurate and useful responses. With experience, it became easier to refine queries to better align with project needs, ultimately improving the quality of outputs received. While there were moments of frustration when responses did not meet expectations, persistence and iterative refinement of prompts often led to useful insights and solutions.

Generative AI significantly reduced the time spent on complex tasks such as analyzing model outputs and troubleshooting code, making the overall workflow smoother and more efficient. Although challenges arose, such as occasional irrelevant responses and the need for manual adjustments, the long-term benefits of using ChatGPT outweighed these drawbacks.

Overall, my experience with generative AI in this project was highly positive. It served as a reliable assistant, particularly in areas such as data analysis and debugging, where it provided clarity and direction that contributed to the successful completion of the project.

Saul:
Using generative AI in my project was helpful, especially for interpreting results and writing down findings. It made analysing complicated data easier by breaking it down and pointing out key insights I might have missed. It also helped me write about my findings in a clear and organised way, which saved me time and made the process smoother. I was surprised by how well it turned complex ideas into simple explanations, making my work easier to understand. This experience

changed how I see AI in academic projects. I now think it's a great tool for saving time and improving accuracy, especially with data and documentation.

Luca:
Using ChatGPT for this project has reinforced my confidence in the power of generative AI. The more I use it, the more I see how transformative it can be in simplifying complex tasks and enhancing productivity. ChatGPT played a crucial role in this project by accelerating tasks such as generating Python scripts, visualising dataset distributions and drafting documentation. Its ability to brainstorm and provide structured outputs significantly improved the workflow and allowed me to focus on higher-level challenges.

One of the most valuable aspects of this experience was learning how to refine and adapt AI outputs. While ChatGPT provided excellent starting points, such as functional code snippets and well-written drafts, its outputs required critical evaluation and refinement to align with the project's specific needs. For example, debugging and tailoring its Python scripts for dataset analysis deepened my understanding of the technical processes involved.

What stands out most is how seamlessly generative AI integrates into academic workflows when used responsibly. By treating AI as a collaborative partner, I was able to maintain originality and ensure that the final deliverables reflected my knowledge and expertise. This project demonstrated that generative AI is not just a tool for efficiency but a catalyst for creativity and problem-solving.

Taking everything into account, my experience with ChatGPT has been highly positive. It has not only enhanced the quality of this project but also shown me how generative AI can redefine the way academic and technical work is approached. I am excited to continue using it in future projects and to explore its full potential.

7. References and List of Resources Used

OpenAI, "ChatGPT: Optimizing Language Models for Dialogue," OpenAI Documentation, 2023. [Online]. Available: https://openai.com/chatgpt.