2024

Research Report MultiRotorResearch

By:	Mohamed Dira Calvin Zhen Gökay Atalay Jordy Walraven
Group:	Group 2
Version:	3
Tutors:	Martijn Lamers Qin Zhao
Stakeholder:	Sieuwe Elferink
Project:	Multi Rotor Research
Date:	14-01-2025

Inhoudsopgave

Introduction

In a world where pollution is becoming increasingly important, technology is essential for a sustainable and innovative solution to this problem. This AI Trash Detection project is mainly focused on using artificial intelligence to tackle waste detection on a large scale. Extensive detection models and modern drones play an important role in identifying and locating all kinds of waste on various surfaces.

The reason for this project is mainly to respond to the increasing amount of litter in the Netherlands, which not only causes visual nuisance, but also causes significant damage to the environment/biodiversity. This AI Trash Detection project in collaboration with MRR can support municipal authorities, companies, and environmental groups in cleaning up waste in public areas faster and reducing the operating costs to do so. The benefits can be seen not only in terms of efficiency and cost reduction, but also in making a more sustainable contribution to a cleaner earth.

This document describes the research that has been made during the past 18 weeks. In this, the research questions that are processed in the Business Proposal are answered and a clear process is described about how we started, which paths we took and finally what conclusions we came to with the AI-Trash Detection project. Within this project, the following questions/chapters are answered:

1. The core operations of MultiRotorResearch (MRR)
2. The different systems within MRR and how they work together.
3. How data is being collected/annotated for AI-modelling and how this can be improved.
4. What the key patterns, distributions, and anomalies are in the dataset.
5. Which models are currently used and what their performance metrics are.
6. How the current AI-model can be improved in efficiency and accuracy.
7. What the critical steps/considerations for integrating the AI Trash Detection model are.
8. Other aspects where AI and drone technology can be implemented within MRR.

By answering these questions, both we and MRR get a clear picture of the past, the current situation, and the desired future situation. This is with the aim of combining technology and sustainability, making this project a step towards a future in which smart solutions contribute to tackling complex environmental problems.

1. What are the core operations of MRR, and how does the use of AI enhance value for its target group?

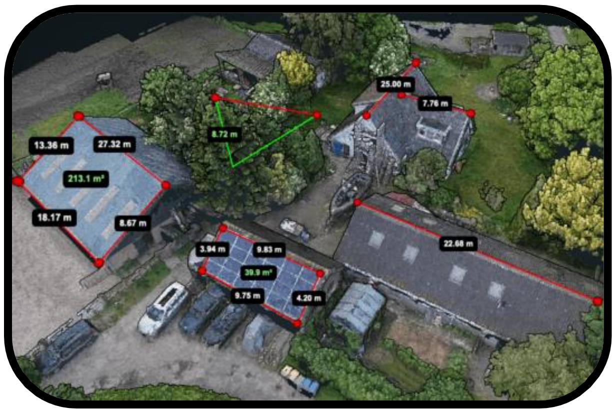
Who is MultiRotorResearch (MRR)?

MRR as an organization

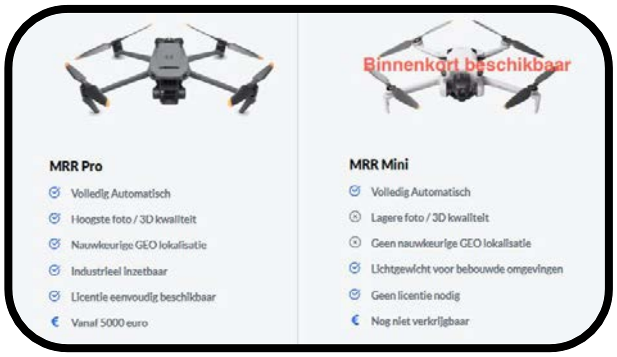
MultiRotorResearch (MRR) is a company specializing in providing services and products related to drone and AI technology. Founded in 2022, the company operates out of the Fontys building at Achtsteweg Zuid 151c, Eindhoven, with a workforce partly composed of students and staff. Students contribute to the company's operations through projects they undertake on behalf of Fontys. These projects may involve improving existing technologies or developing new technologies that stakeholder companies like MRR can implement in its operations.

Services and products offered by MRR.

MRR offers various services to its clients, primarily in the fields of AI and drone technology. First and foremost, MRR provides a range of use cases. These include applications of AI and drone technologies for inventory management, groundwork, distance measurement, building inspections, festivals, and, notably, AI Waste Detection. They also offer customized solutions, where they develop a tool tailored to the specific needs of the client.



In addition to these services, MRR provides drones, specifically two models equipped with MRR software, such as MRR-HUB. These drones are available for purchase or rental, depending on the consumer's preferences. In both cases, the drones undergo extensive testing, usage demonstrations, and training to ensure that clients can operate them independently with minimal support from MRR.

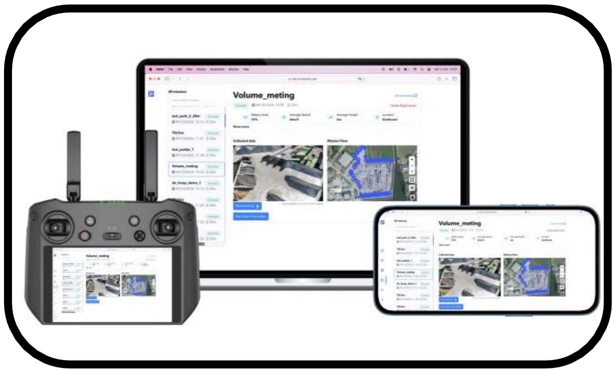


Binnenkort beschikbaar

MRR Pro	MRR Mini
✓ Volledig Automatisch	✓ Volledig Automatisch
✓ Hoogste foto / 3D kwaliteit	⊗ Lagere foto / 3D kwaliteit
✓ Nauwkeurige GEO lokalisatie	⊗ Geen nauwkeurige GEO lokalisatie
✓ Industrieel Inzetbaar	✓ Lichtgewicht voor bebouwde omgevingen
✓ Licentie eenvoudig beschikbaar	✓ Geen licentie nodig
€ Vanaf 5000 euro	€ Nog niet verkrijgbaar

The target group of MRR
MRR primarily focuses on the business segment, targeting companies and government agencies that wish to utilize MRR's applications for commercial purposes. The target audience for MRR's services and products aims to achieve greater efficiency through drone and AI applications. For example, this includes calculating the volume of raw materials or quickly detecting waste.

The Unique Selling Point of MRR
MRR's Unique Selling Point lies in its ability to cater to both individuals with extensive drone knowledge and those with little to no experience. Users who are well-versed in drone technology can rent or purchase a drone and operate it independently. Meanwhile, those unfamiliar with drones can benefit from the "Drone as a Service" model, which automates flights and requires minimal user input. By addressing a wide range of customer segments, MRR distinguishes itself from competitors.



How AI Enhances MRR's Business Values
AI enables MRR to continuously innovate, ensuring it is not solely reliant on drone technology. It allows the company to serve diverse audiences and segments through applications tailored to various use cases. This diversification creates multiple revenue streams, strengthening the company's financial stability. The variety of applications MRR offers enhances its specialization and professionalism, giving consumers a compelling reason to choose MRR over competitors.

2. How do the different systems (drones, cloud environment, app) work together and how is the AI model applied to this?

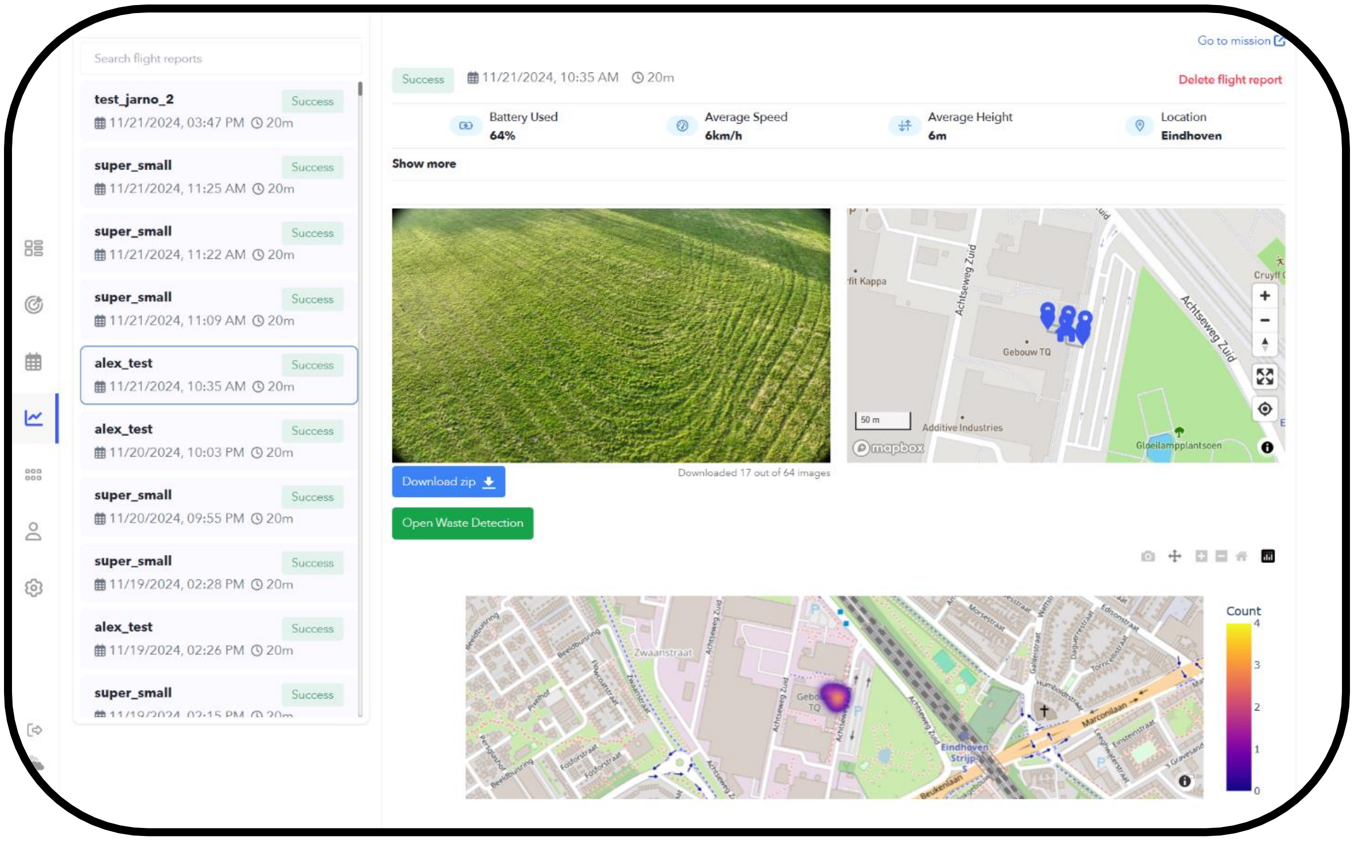
What different systems does MRR have?

MRR incorporates multiple interconnected systems to achieve all its functionality. Let's take a look at these systems:

1. Mission Control

Mission Control is a proprietary application developed by MRR for managing drones. It offers features such as:

- **Flight Planning**: Set up flight paths for drones.
- **Plugin Support**: Extend functionality with plugins, such as trash detection.



Search flight reports

test_jarno_2 Success 11/21/2024, 03:47 PM 20m

super_small Success 11/21/2024, 11:25 AM 20m

super_small Success 11/21/2024, 11:22 AM 20m

super_small Success 11/21/2024, 11:09 AM 20m

alex_test Success 11/21/2024, 10:35 AM 20m

alex_test Success 11/20/2024, 10:03 PM 20m

super_small Success 11/20/2024, 09:55 PM 20m

super_small Success 11/19/2024, 02:28 PM 20m

alex_test Success 11/19/2024, 02:26 PM 20m

super_small Success

Success 11/21/2024, 10:35 AM 20m

Battery Used 64%

Average Speed 6km/h

Average Height 6m

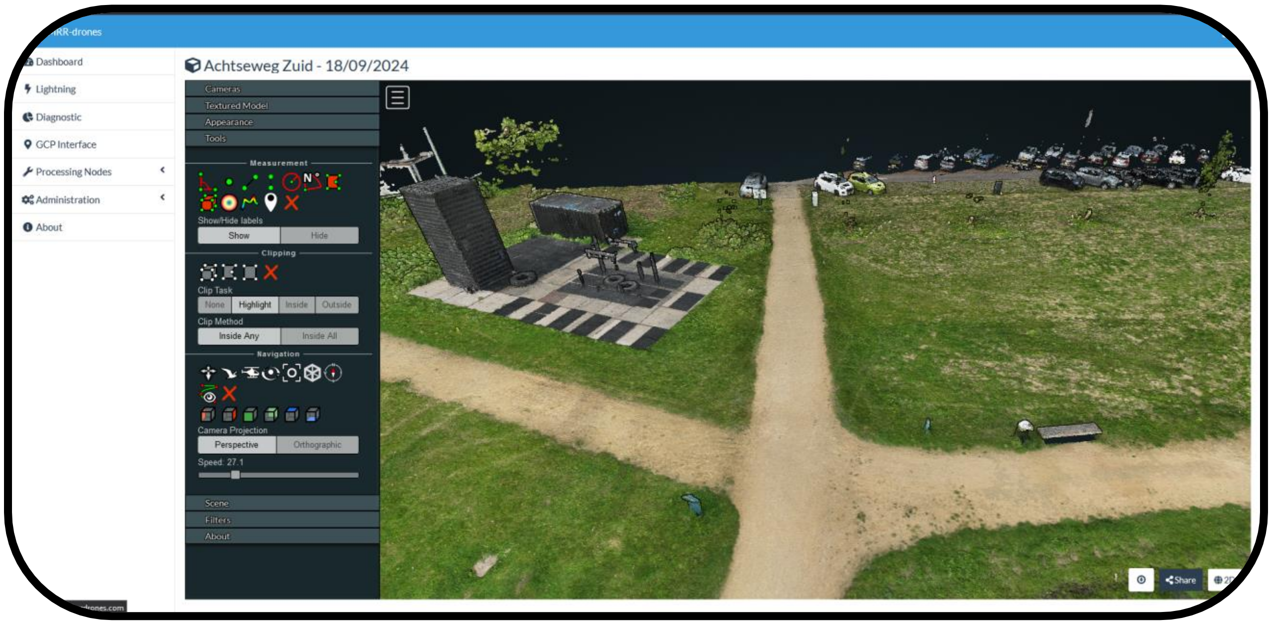
Location Eindhoven

Downloaded 17 out of 64 images

2. OpenDroneMap

OpenDroneMap is an open-source application utilized by MRR to process drone data. It enables:

- **Orthophoto Creation**: Generate detailed aerial maps from drone images.
- **Measurements**: Perform tasks like volume and distance measurements. OpenDroneMap can be run through Docker, making deployment and scaling efficient.



3. AWS S3

MRR uses multiple Amazon S3 buckets as part of its storage infrastructure. Examples of stored data include:

- Drone images.
- HTML files.
- Modified images.
  S3 serves as a reliable and scalable storage solution to manage large datasets efficiently

4. Drone

MRR's Drone Fleet

MRR operates multiple drones, with the **DJI Mavic 3** being their primary choice due to its advanced features and reliability.

DJI Mavic 3 Features

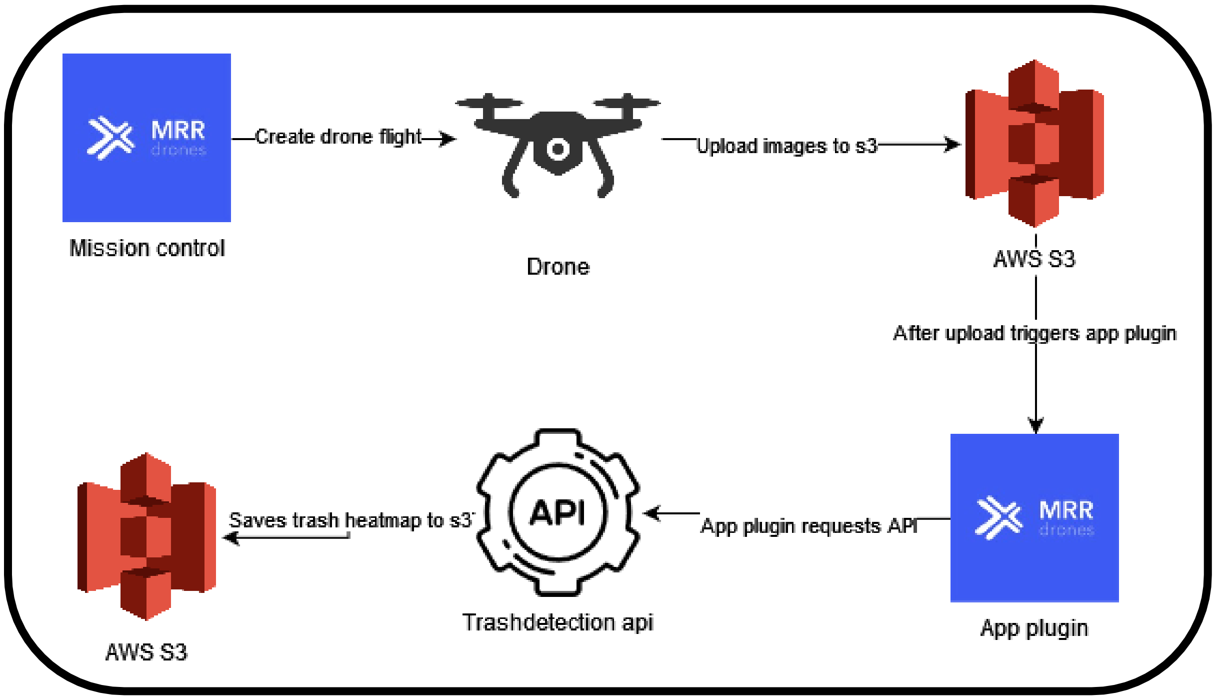
The DJI Mavic 3 offers cutting-edge capabilities that align with MRR's operational needs:

- **Autonomous Flight**: Fully compatible with MRR's Mission Control application, allowing pre-programmed flight paths and remote management.
- **High-Quality Imaging**: Equipped with a professional-grade camera, ideal for capturing detailed imagery for mapping and analysis.
- **Precision Navigation**: Incorporates advanced GPS and obstacle-avoidance systems for safe and accurate flight.
- **Long Battery Life**: Ensures extended flight times, maximizing efficiency during missions.

How do the services work together

MRR developed an autonomous drone system integrated with app plugins for seamless operation. The application workflow is as follows:

1. **Mission Initialization**: The process begins when the user starts a new mission within the application.
2. **Autonomous Flight**: The drone executes the mission, capturing images during its flight.
3. **Image Upload**: Upon completing the mission and landing, the drone automatically uploads the captured images to an Amazon S3 bucket.
4. **Plugin Trigger**: Once the images are successfully uploaded, the application triggers a designated app plugin.
5. **AI Trash Detection**: In this setup, the triggered plugin activates the AI-TrashDetection API, which processes the uploaded images to identify trash.
6. **Heatmap Generation**: After processing, the TrashDetection API generates a heatmap indicating trash locations and uploads it back to the S3 bucket.



In summary, MRR's ecosystem demonstrates a well-integrated workflow combining drones, applications, and cloud infrastructure to achieve its functionality. Each system plays a distinct yet interconnected role: Mission Control orchestrates drone operations and integrates plugins like AI-based trash detection; OpenDroneMap processes aerial data into actionable insights; AWS S3 provides scalable storage for drone images and processed outputs; and the DJI Mavic 3 drones serve as the primary hardware, capturing high-quality data with precision. The AI model seamlessly integrates into this system via plugins, enabling real-time analysis and automation. By leveraging these technologies, MRR achieves a streamlined and efficient approach to drone-based environmental monitoring and mapping.

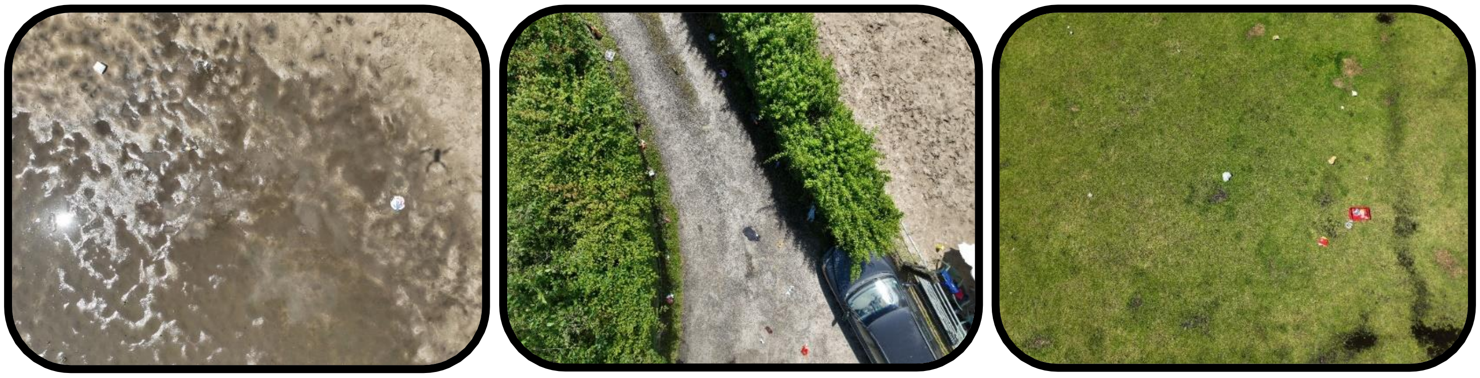
3. How is data being collected and how is it annotated for AI modeling? How can this be improved?

Current data

From researching MRR's environment and what our stakeholder Sieuwe told us during our first stakeholder meeting about data collection and storage, their data is acquired by drones that take pictures during their flights. As explained by Sieuwe, the drone first gets a pre-determined path assigned to it and takes a stream of pictures during the flight. These get uploaded to their Amazon AWS environment, after which the images can get processed by the AI models. Besides the AWS environment, MRR also has a Google Drive where they have a copy of their data stored. The current dataset used for trash detection by MRR consists of 169 unannotated images and MRR does not have any methods in place to annotate the images they have collected.

The images from MRR used in the context of trash detection are all taken at a single location with multiple ground types, including sand, grass, mud and asphalt during daylight.

For the trash in these images, MRR and the previous AI project group have gathered a set random object used as a depiction of trash that can be found outside (e.g. empty cans or bottles, plastic bags etc.)



Example images in MRR's dataset.

These images provided by MRR do have a downside, namely that the data is monotone, as they contain recurring ground types and trash objects. This causes the model to overfit on the ground types and objects present in the dataset and may not perform as well in other situations. While only containing images taken in daylight may also cause the same effect, Sieuwe mentioned that drone images are only taken during daylight hours on dry days, as taking images in the dark isn't as effective and bad weather can affect the drones.

Repeating objects in multiple images. DJI_20240617125704_0001_V.JPG, DJI_20240617125707_0002_V.JPG, DJI_20240617125742_0008_V.JPG

Within the previous group's work, they used a dataset of which they didn't state any source. It was later found that the dataset originated from the UAVVaste GitHub repository (PUTvision, 2023). The repository contains 772 images, which have a total of 3718 annotations of trash within them in COCO format.

The images are more varied compared to the MRR images, containing, but not limited to gravel, forests and tile ground types. These images are specific in the context of it being drone imagery as stated in the research document by (Kraft, Piechocki, Ptak, & Walas, 2021), which differs from datasets like COCO where its images are taken from a pedestrian's point of view. This UAVVaste dataset is made available to the public with the intention of it to be expanded with more data with accommodating annotations, though it seems that this hasn't happened since November 14th, 2020, as of writing this document.

Data annotations

The UAVVaste dataset comes pre-annotated with a CSV-file within the GitHub repository. These annotations are done in the COCO format containing both segmentation data and bounding box data.

The dataset received from MRR does not come with annotations for its images. Currently, it has been decided to opt for the use of the annotation tool: CVAT, which can be run locally. This tool has the option to export the annotations in multiple different formats, including the COCO format used by the UAVVaste dataset. The trash within the MRR dataset was annotated in segmentations and then exported into the COCO format, which automatically converts the segmentations into bounding boxes.

How to improve data collection and annotations?

As stated before, the current data used is received from MRR and enriched with data from the UAVVaste GitHub repository. The data received from MRR are not annotated and as it stands, the images from MRR will not be annotated within their current environment. Both datasets are also currently stored in different locations.

For the collection of the data, it can be improved by storing all the used data into a centralized (cloud) storage to streamline the process of accessing the data.

For the annotation of new data from MRR or other unannotated images, a sufficiently trained CNN model can be used for self-training, where the model makes predictions on unlabeled data and feeds the new data as new training data. In the earlier stages, human involvement (Human-in-the-Loop) will be necessary to give feedback and correct the predictions made by the model. As the model improves, this involvement can be lessened but should not be left to the model completely.

4. What are the key patterns, distributions, and anomalies in the dataset that may impact the performance of the AI trash detection model?

For our project we will be using multiple datasets, the datasets we use consist of images recorded by MRR, and a dataset on GitHub named UAVVaste. Since the use-case of our project is very specific, meaning to only detect pieces of trash, the images we need are pretty specific.

Dataset Characteristics

UAVVaste
The UAVVaste dataset contains many images from a top-down view, which kind of resembles a drone-view. It becomes obvious that not all of the images are taken using a drone, because of the height at which the images are taken. In some of them the images are taken relatively low and at a skewed angle.

The terrain above which these images are taken consists of grass fields and concrete floors. Although, these fields are mostly empty with just some pieces of trash and not much else going on. The quantity of the images is rather large, 772 images.

MRR
This dataset is more consistent as it's not open source like the UAVVaste dataset. This dataset is provided by our stakeholder, which he himself has helped in creating. They flew a drone over different fields like grass fields, concrete and mud. These images are taken at a higher altitude, which helps the model recognize the surrounding environment.

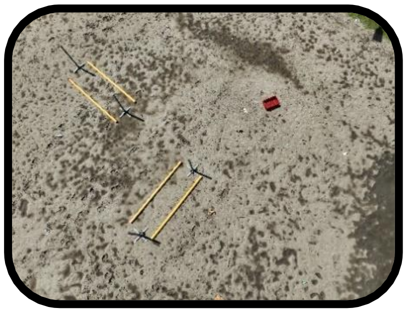
These images are taken mostly of empty fields, like UAVVaste, but this dataset has at least some images in which other objects are visible, like cars and buildings. This dataset contains 169 images, which is much less than the previous dataset.

Patterns in the dataset

Environment

There are many different environments in the images, in which the images are taken. In the MRR images it's mostly grass fields and sand. When looking at the UAVVaste images there are also images with concrete floors. This will help the model work correctly with different terrains.

Grass


Sand

Concrete

Trash distribution

Since clusters of trash in the world varies in density, it's important to have a dataset which has loose single pieces of trash as well as multiple pieces clustered together. This will paint a more realistic picture of what trash can look like in the real world.

Trash clustered together

Large distance between trash

Object size

Trash can differ when it comes to its size and scale. Smaller objects that appear in the dataset frequently are cans of drinks, larger objects present in the data are plastic bags containing other items. The model should know that these pieces can have varying sizes and shapes, so that the model doesn't just recognize items of a certain size/shape.

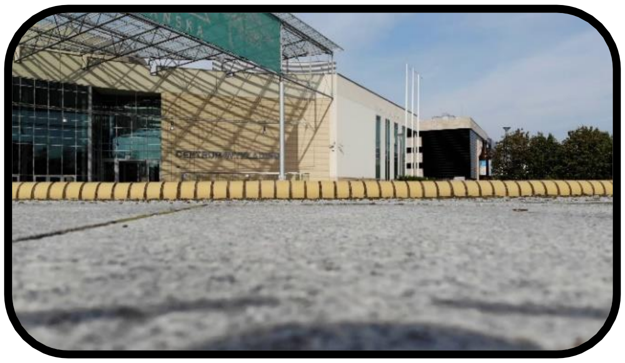
Larger pieces

Smaller pieces

Potential Anomalies

A potential anomaly that immediately comes into mind is a misrepresentative dataset of what the actual data will look like. This can include multiple things, for example the type of terrain the drone will fly over. The drone might get confused if it flies over a type of terrain that is not represented in the dataset.

Angle
Since we are talking about drone images, they should be looking directly from above. Images taken from a phone looking across a field is an example of what we do not need. This can mess up the model as the backdrop completely changes from the floor that the drone is flying over.



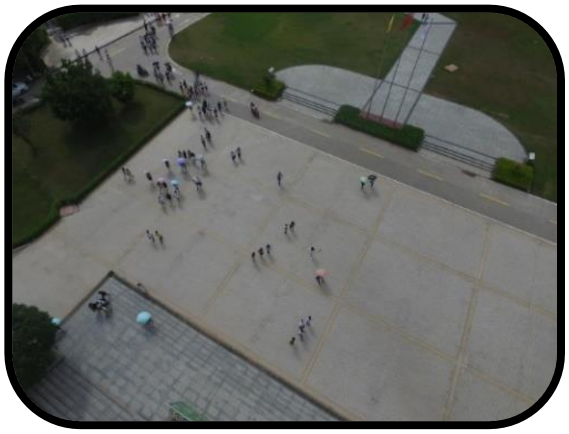
❌ Horizontal

✓ Top-down view

Obstacles

As the drone will be flying over many kinds of terrain, chances are that other objects may come into view, which are not trash. This can be anything from cars, to people, to parts of buildings and many more. If these objects are not present in the training data, the model may label these as trash, because it's not used to these objects. To counteract this, we need to have images with a lot of cars, people, extra obstacles that may not be trash necessarily.

❌ Horizontal


✓ Top-down view

5. Which AI models (e.g., convolutional neural networks, YOLO) are currently used for trash detection in drone footage, and what are their performance metrics?

Previous group

Going through the GitHub repo from the previous group, there are branches with multiple different models being tested. These models are:

- YOLOv8n
- InceptionV3
- Masked RCNN

From the notebooks that were handed over to us via the GitHub repo none of them were able to run and with a lack of documentation, we were unable to replicate their work within this repo.

On the project PC from MRR, there are notebooks present where they have only run the YOLOv8 model. The previous work has run a total of 14 training runs on the model, out of which 12 were done on bounding box data and 2 were done on segmentation data, which is the data type we are using. From the metrics from the train runs, the best results found in their model are:

Precision:	0.70715
Recall:	0.49891
mAP50:	0.52637
mAP50-95:	0.20922

Current work

Results October 10th, 2024 (first iteration)
As of October 10th, 2024, we have trained 2 models as of now, namely YOLOv11 and RT-DETR. Both models were run with 2 variations of the dataset, one with the entire dataset and the other excluding the GOPR images within the UAVVaste dataset due to orientation issues we had at the time. The metrics of these models are as follows:

Black = *baseline/equal*
Green = *better (compared to baseline)*
Blue = *best*
Orange = *worse (compared to baseline)*
Red = *worst*

Model\Metric	**Precision**	**Recall**	**mAP50**	**mAP50-95**	**cls_loss**
YOLOv11 (with GOPR)	0.81732	0.65723	0.70476	0.39708	0.88714
YOLOv11 (without GOPR)	0.85408	0.70618	0.76048	0.43869	0.88714
RT-DETR (with GOPR)	0.87298	0.8894	0.80489	0.49087	0.45454
RT-DETR (without GOPR)	0.87861	0.84868	0.86812	0.51838	0.46685

For a detailed report for this part, see Appendix A.

Results January 7th, 2025 (further iterations)
At this point, we have adopted the Detectron2 model as one of the models we experimented with. This model along with the RT-DETR model were both improved through data enrichment and hyperparameter tuning. The YOLO model was dropped, as it underperformed a lot compared to the other two models used. The summary of the model performances can be seen in the table below, with the first row being the same as the last row of the previous table:

Black = *baseline/equal*
Green = *better (compared to baseline)*
Blue = *best*
Orange = *worse (compared to baseline)*
Red = *worst*

Model\Metric	**Precision**	**Recall**	**mAP50**	**mAP50-95**	**cls_loss**
RT-DETR (without GOPR)	0.87861	0.84868	0.86812	0.51838	0.46685
RT-DETR (extra images)	0.934	0.831	0.89	0.544	N.A.
Detectron2 (bbox)	N.A.	N.A.	0.88064	0.57138	N.A.
Detectron2 (segments)	N.A.	N.A.	0.88250	0.49873	N.A.

For a detailed report for this part, see Appendix B.

Out of the RT-DETR and Detectron2 models, both of the newest models show similar performances, with each having some objects missed in different images. Detectron2 does have consistently higher confidence scores than RT-DETR but struggles more with smaller objects.

Out of these models, we would recommend the RT-DETR model, as the models perform very similar, but RT-DETR is easier to set up and run due to the possibility to run it on the CPU, while the Detectron2 model requires PyTorch and a GPU to run. Additionally, during the hyperparameter tuning of both models, only RT-DETR showed improvements, while Detectron2's performance didn't. It is, however, still worth looking into improving Detectron2, as it has a lot more parameters that can be tweaked compared to RT-DETR.

6. How can the current AI model for trash detection be improved in terms of accuracy and efficiency?

The changes that we applied

During the 18 weeks of production time, many different things were tried to improve the accuracy and efficiency of the existing models and the newly introduced models. First, we further expanded the original UAVVaste dataset of 772 images with the images from MRR. The images received from MRR were 169 images. These were raw images without annotations. These images were therefore first annotated with the CVAT tool, an annotation tool that allows manual annotation for Machine Learning. After the annotations were finished, they were exported as COCO 1.0 format, which is a format used for the YOLO, RT-DETR & Detectron2 models. These were merged with the UAVVaste dataset in the modelling notebooks, after which they were split into train, test, and validation subsets. Adding the images helped significantly because the performance of the original model improved a lot. The metrics for this can be found in the previous chapter.

In addition to expanding the dataset, we also tried to improve the performance of the model by means of hyperparameter tuning. For example, we tried to increase and decrease the number of epochs during training, in order to see what the effect of the number of epochs was on the above metrics. We also played with the batch size in order to optimally use the GPU limits of the server. Furthermore, parameters were also added to improve the learning curve. For example, an adjusted initial learning rate was applied with an end factor in combination with a cosine learning scheme, so that convergence can be achieved more smoothly.

Within hyperparameter tuning, we have applied parameters for detection in addition to the learning scheme. For example, we have applied a confidence score with IoU threshold for more accurate predictions, but also to ensure that the predictions made by the model are applied more strictly. To prevent overfitting, we have applied a patience, but also a dropout and weight_decay. Finally, amp and nbs have been applied for improved training speed because this handles the available GPU memory better. This uses less memory and model weights are updated more consistently.

After tuning the hyperparameters, it was seen that the model still saw people who were photographed from above as trash. This is because the model was less trained on people who were in the image. As a result, the model confused people with trash. This was looked at by adding more data. A dataset was found with drone images that showed people. This dataset was published on RoboFlow, but sadly this dataset has been removed since. We have placed a copy of this dataset on the Google Drive of MRR, where the images can still be downloaded.

These images were then added to the images from UAVVaste and MRR that were available. After adding this dataset, an improvement was seen in how the model saw people in relation to trash. However, there still appeared to be some confusion because the model still sometimes saw people as rubbish. Although this occurred to a lesser extent, there was still a need to limit this as much as possible. An attempt was made to add more data to the existing dataset by applying images that showed both people and rubbish. For this, a beach dataset was found from RoboFlow. This is a beach dataset with approximately 1,500 images with both rubbish and people. After the entire beach dataset was applied, we saw that the model performed less well. This was because the model was dependent on the beach dataset images. To solve this, we looked at reducing the added beach dataset to a few hundred images. This also turned out to have a negative effect on the performance of the model.

Ultimately, it was decided to keep the first dataset with people and remove the second dataset with beach images completely. The model currently has false positives for people in the images, but because these are limited to an acceptable minimum, it is safe to implement within the MRR organization for now.

The improvements for the future

Adding new data

The main recommendation is to add more data. Currently, the dataset contains approximately 967 images. This is the total of MRR, UAVVaste and the dataset with people and is relatively limited. Especially considering the variation in images that is needed to train a generalizing, robust model. If this number is increased by means of new drone images with more variation in surface type and waste types, the model will also learn to deal better with various scenarios, which will also improve the predictions on the test dataset. An extension of the dataset should focus on the following improvements:

1. More Variation in Surface Types:

The current dataset could use improvements in the diversity of surface types. This is the surface on which the waste is located. Currently, grass, mud/sand and stone/concrete are included as surfaces in the dataset. By expanding this with surfaces such as snow, rocky terrain, leaves/swamp, vegetation, etc., the model can generalize much better. It is also important here that the distribution of the different surfaces is approximately the same, so that the model is not biased towards a specific type.

2. Expansion of Waste Types:

In addition to the different surface types, it is also important that different waste types are represented. If we look specifically at the MRR dataset, the photos were taken on the same types of waste. This should be diversified as much as possible, so that the model can generalize better to different waste types in addition to the surface types.

3. Different Light and Weather Conditions:

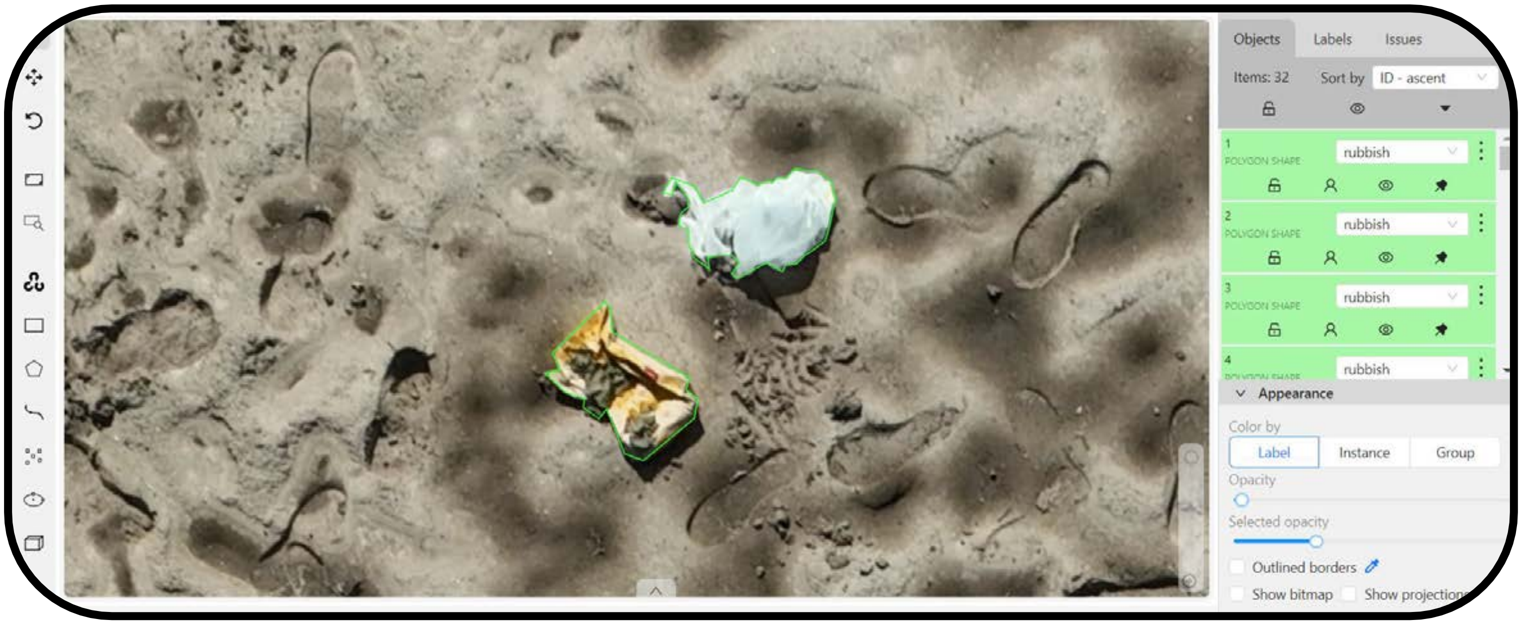
The light and weather conditions have less influence on the way the model trains. For example, the images of MRR are always taken during the day. However, when taking the photos for future database expansion, it is important to look at the different light and weather conditions, because in addition to MRR, the model also trains on the images of UAVVaste, which were taken in different light and weather conditions. Because part of these images is in the test subset, an increase in variation in this respect can increase the performance of the model.

4. Use of Angle and Height Variations:

The use of different angle and height variations is also important when creating the images. The MRR dataset contained a number of images that were taken at different angles and heights. For example, the images were sometimes taken from the side, but mostly from above. Because the representation of images taken from the side was not sufficient, the model had difficulty predicting the waste in these images. By adding more of these images, the model becomes more robust, allowing it to be used in multiple camera perspectives.

5. Data Annotation:

Improvements can also be made when annotating the images. We used CVAT, which is an annotation tool with certain limitations. For example, it is not possible to annotate directly, but it is necessary to draw around each waste. This is extremely time-consuming and also does not provide precise annotations. By using other alternatives such as Label Studio, COCO Annotator or VoTT. If this allows for more precise/faster annotations, this will also have a positive effect on annotation efficiency and model performance.



Searching for more balanced data with waste and people
As previously described, both RT-DETR and Detectron2 still struggle when there are people in the images, because the model confuses this with waste. In order to limit this problem as much as possible, new datasets can be applied where people and waste are present. The beach dataset that was used previously can be used. It is important to ensure that not too much data is taken from this dataset, because this can negatively affect the performance of the model. It is recommended to take a subset of this dataset and to find comparable datasets with other surfaces, such as grass, stone, concrete, etc. An equivalent subset size should also be taken from this, to ensure that the distribution is approximately the same. This can be annotated again if necessary and added to the existing dataset. Due to this variation in data, it is possible that the detection capabilities of the model for waste with people can be significantly improved.

Working on a new model?
It is not recommended to investigate other models to implement based on this use case, since all major object detection models have been partially or fully investigated. For example, we have already investigated the following models:

- EfficientDET
- YOLO V11
- Detectron2 (Masked R-CNN & R-CNN)
- RT-DETR
- RetinaNet (ResNet)
- U-Net

It is recommended to use RT-DETR or Detectron2 as a broad base and to possibly improve this by broadening the dataset with the above recommendations.

7. What are the critical steps and considerations for deploying and integrating the new AI trash detection model into the existing system?

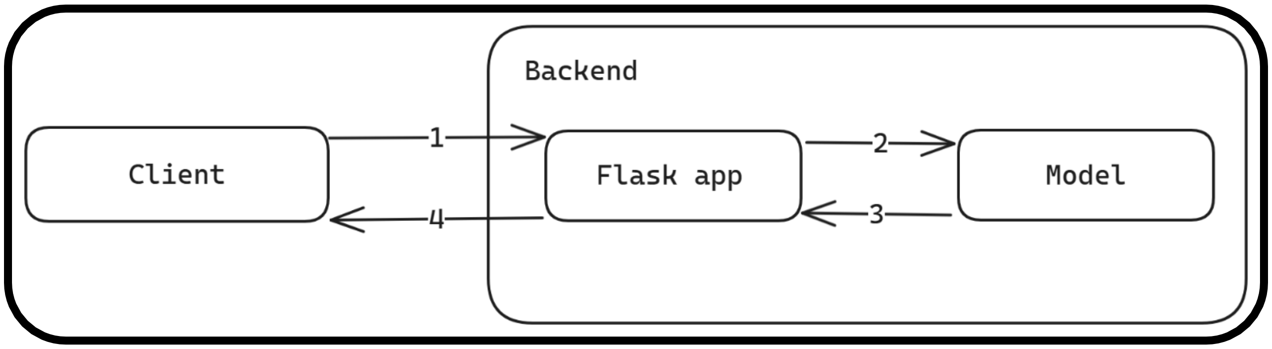
The existing system

MRR has created a platform where they can slot in plugins, these plugins should be hosted somewhere. This could be on the PC owned by MRR or a cloud environment like AWS. Each of these plugins is a different application of object detection using these drones. As this platform grows and more plugins are developed, MRR will build a large collection of issues their drones can solve.

Their platform contains multiple separate components that communicate with each other. They have a dashboard within which they can plan and execute drone flights. When the drone has flown its predetermined path, it uploads the images that it has taken to their AWS S3 bucket. When that is done, they can initiate the processing of the images from a client application. What happens after is where our app comes in.

Standardized Request and Response Format

We received documentation containing multiple things, including information about the high-level flow of the system. More importantly, the documentation specifies the data that the client needs to send to our API (request) and the data it expects to receive (response). Our API has to carefully follow these conventions to ensure proper working between the components.



1. Below you can see a diagram of the flow between our modules.
   The client sends a request containing asset data like image download-URLs attributes our backend might need like latitudes and longitudes of the images.
2. Our backend downloads the images using the URLs and sends them to the model for inference.
3. The model returns the results, and our backend generates a heatmap.
4. The client receives the heatmap along with the prediction-data in the response.

Deployment

Docker

After our app has been developed, we'll need to host it in MRR's environment. They have an AWS environment which hosts an array of systems, along with MRR's own servers which is able to run a code server which we make us of, for example.

No matter where it's hosted, the process should be as easy as can be. In order to achieve that we'll make use of containerization. This is an approach which runs software inside mini-virtual machines (containers), so that the software runs independently from the host-machine's hardware.

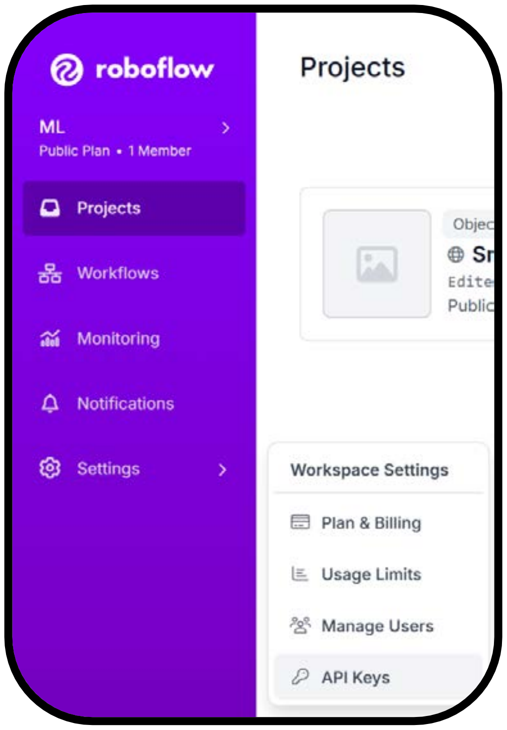
For this case we'll utilize Docker. Docker is a platform for containerization that allows you to package applications and their dependencies into lightweight, portable containers. These containers allow us to run the app in every environment consistently, avoiding "it works on my machine" scenarios.

For this, we'll create a Dockerfile, which defines the steps to build our application image, and a docker-compose.yml file to specify GPU-support. Once set up, this approach will ensure that anyone can run the application with just a few commands, making the process seamless and efficient.

The .env file

There should also be a .env file present. This file contains values that are used within the project, that may be easily changed if additions are made in the future. This file, for example contains some paths in which files are generated.

Another *env* file that is necessary for our project is the one inside the *notebooks* folder. This file contains your private RoboFlow API-key, which is needed to download a RoboFlow dataset which is used in our project. This API key is easily accessible through the main RoboFlow website. After logging in, click on *settings* and then on *API Keys*. Copy the *Private API Key* and paste it into the .env file under the key *ROBOFLOW_API_KEY* and you should be good to go.



Switching models

Our backend works by making use of the weights of the trained model that are present within the project. This means that there must be weights present in order for the backend to work at all. The code expects there to be a *model* folder containing the weights of a single model. In its current state, the code assumes the weights belong to a Detectron2 model, as the code is not compatible with weights from other models.

This folder is intentionally excluded from the Dockerfile during the build process. Instead, the project accesses the model weights through a volume bind. Volume binds enable Docker to access files directly from the host system rather than limiting access to the container's internal file system. For example, if we wanted to modify something in the code, we would typically need to rebuild the image and then start a new container using the updated image. However, with a volume bind, we can swap out the model weights on the host system without rebuilding the image.

This approach also allows MRR to run multiple instances of this backend for different projects simultaneously, each requiring unique model weights. For instance, if one container is dedicated to a trash-detection use case, another container can run concurrently to handle a rust-detection model, all without conflicts or redundant builds.

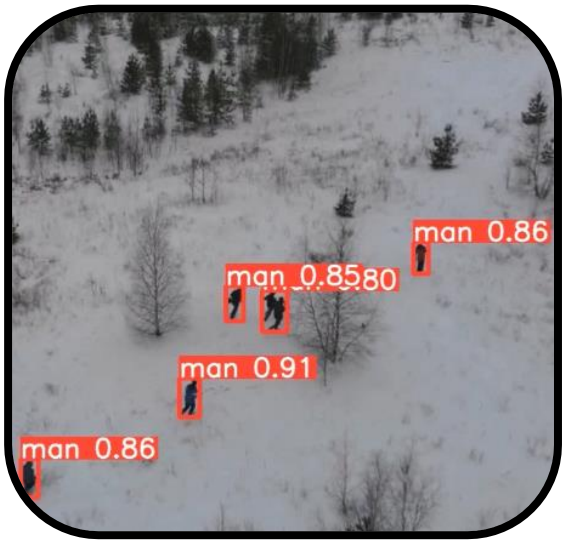
8. In what other aspects can AI and drone technology be implemented for MRR to use in their core business? What are the latest trends?

What kinds of ideas can be implemented within MRR?

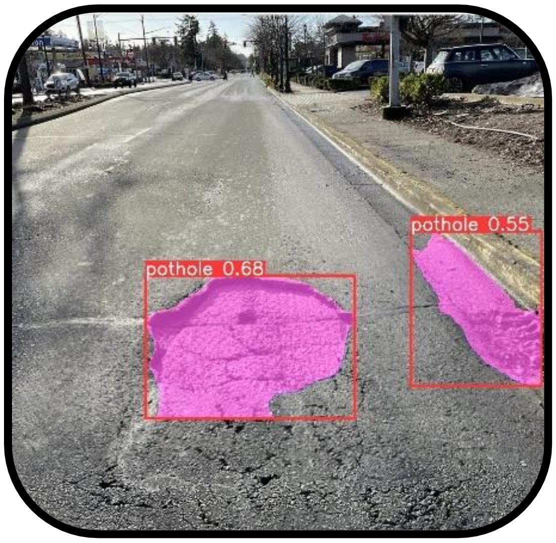
Drone technology in combination with AI is being further innovated, so the applicability of drone technology with AI is increasing in various areas. The reason for the increase in drone use is mainly due to the efficiency and saving on resources that it brings with it. For this research question, the intention is to create a new idea that MRR can apply in their core business. In this, a few ideas are described, and the best is selected based on the availability of data and the wishes of the stakeholder.

Finding missing people in remote areas.

Approximately 40,000 people are reported missing in the Netherlands each year, of which 70% are found within 24 hours. Of the people who are found within 24 hours, 80% are found within 48 hours and 90% of those within three weeks. However, 1,500 people remain missing each year and are also referred to as long-term missing. The problem of missing people is therefore significant in the Netherlands. Drone technology combined with AI can contribute to this by means of more efficient and faster searches in remote areas. Think of areas such as farm fields, forests, mountains, beaches and the like. Emergency services such as police, fire brigades, rescue teams, and families of missing persons are ideal stakeholders who can potentially use this technology with the aim of locating missing people. For this use case, the "Search for Missing People" dataset on RoboFlow would be a good data source. This dataset contains people annotated using bounding boxes which can be loaded directly without downloading.

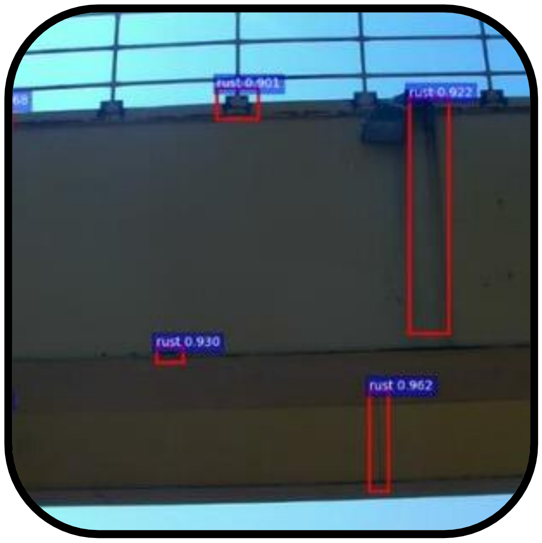


Detection of defects in road for preventative maintenance.



The application of drone technology & AI for inspections and preventive maintenance of infrastructure such as buildings, bridges, roads. Think of detecting defects such as cracks or fractures that can be detected and acted upon. This is interesting, because the demand for automation in maintenance has increased significantly in recent years, especially in difficult to access or dangerous areas. This, with a view to increasing safety and efficiency and at the same time reducing risk. Examples of target groups that MRR can focus on with this are governments at all levels, municipal authorities, infrastructure companies, and contractors responsible for maintenance. For this model, the best datasets to use are for potholes in roads or defects in roads, both of which can be found on RoboFlow. The latter dataset is recommended, as it is more focused on drone images.

Detecting corrosion on metal surfaces for preventive maintenance.



Detecting corrosion on metal surfaces, such as bridges, high voltage or large pipelines would also be interesting to look into. Mainly because these inspections require hiring people who often have to climb to great heights. Automating these tasks not only saves money on personnel, but also ensures a safer working environment. Especially because a drone can fly to the target location within minutes. Operators of bridges, pipelines and industrial facilities, and companies in inspection and maintenance could be potential target groups for this technological application. For this use case, a dataset was found on RoboFlow with corrosion on different surfaces. Unfortunately, this dataset does not offer enough images made by drones. However, there are enough images to train the model and experiment.

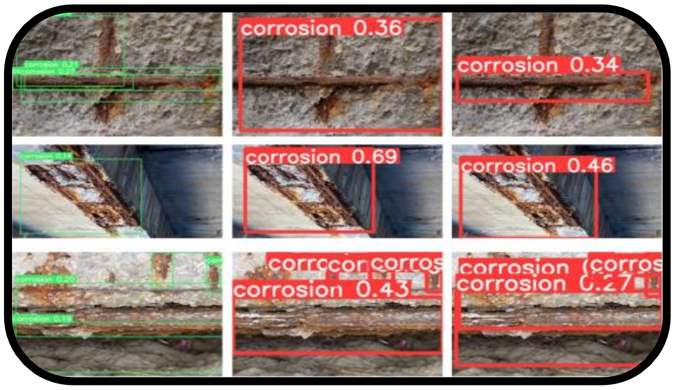
Inspection of agricultural land for detection of stress and diseases in plants.

The demand for drone technology with AI is also increasing significantly in the agricultural industry. Here, farmers are looking for a way to make irrigation of agricultural land as optimal and environmentally friendly as possible. AI can contribute to this with image detection to detect stress or diseases in plants, so that rapid intervention can be carried out to minimize losses and reduce the negative impact on the environment. Consider the application of drones and AI to work towards precision agriculture, where losses in raw materials are minimized, and profits are optimally achieved. This application of technology is mainly aimed at farmers, agricultural companies, agronomic service providers, and organizations in sustainable agriculture. For this project, it is possible to use a HuggingFace dataset, which can also be easily loaded without downloading. This dataset contains drone images of different types of crops with annotations in YOLO format.

Detection of pollution of water bodies.

The application of drone technology in combination with AI can also offer opportunities in the water purification industry. In particular in the classification or detection of pollution in water bodies, such as rivers, lakes, streams or seas. This allows large surfaces to be scanned simultaneously. Areas that are difficult to reach can also be inspected with this, for example rivers with strong currents. Various aspects of the water bodies can be scanned, such as oil spills, chemical discharges, large floating waste, but also algal blooms that can be harmful to health and the environment. The primary target groups would mainly be water purification companies, environmental protection organizations, government agencies, and local communities that deal with this type of pollution. For this application, two datasets can be combined that contain information about oil spills and harmful blue-green algae in water, both forms of pollution in water bodies. These datasets can also be found on RoboFlow and are based solely on segmentation (Blue Algae Dataset, Oil Spill Dataset).

The chosen idea



We have had several conversations with our stakeholder, which show that there is a preference for three applications, namely:

- Detection of pollution of water bodies.
- Detection of corrosion on metal surfaces for preventive maintenance.
- Detection of defects in roads / real estate for preventive maintenance.

Here, there is the most interest in detecting corrosion on metal surfaces and detecting major pollution in water bodies, depending on the amount of data that is available. The interest is mainly based on what competitors offer. After the conversation with Sieuwe Elferink, we gained the insight that the detection of defects on roads is already widely used by competitors, and that it actually offers little unique sales value to the customer. Between the use case regarding pollution in water and rust on metal surfaces, the stakeholder has made the choice to opt for detecting corrosion on metal surfaces. This is due to a great interest that a potential customer of MRR has for this use case. It is therefore wise to use these when creating the new assignment, because MRR currently benefits the most from this.

How are we going to implement this?

For this new idea, we will first start looking for a suitable dataset. We will apply Data Understanding & Data Preparation to this dataset in order to have suitable images that we can feed to the model. For this approach, we will use the models that also performed well on the Trash Detection, namely the Detectron2 & RT-DETR models that both have a high recall & precision between 85%-90%. We will evaluate and explain the performance of these models. For this new project, the intention is to only create and submit a Data Quality notebook (Data Requirements, Data Collection, Data Understanding, Data Preparation) and a Machine Learning & Reporting notebook as our contribution to the new assignment. These documents will also be documented, so that future groups that will work on it will have a broad starting point to work on.

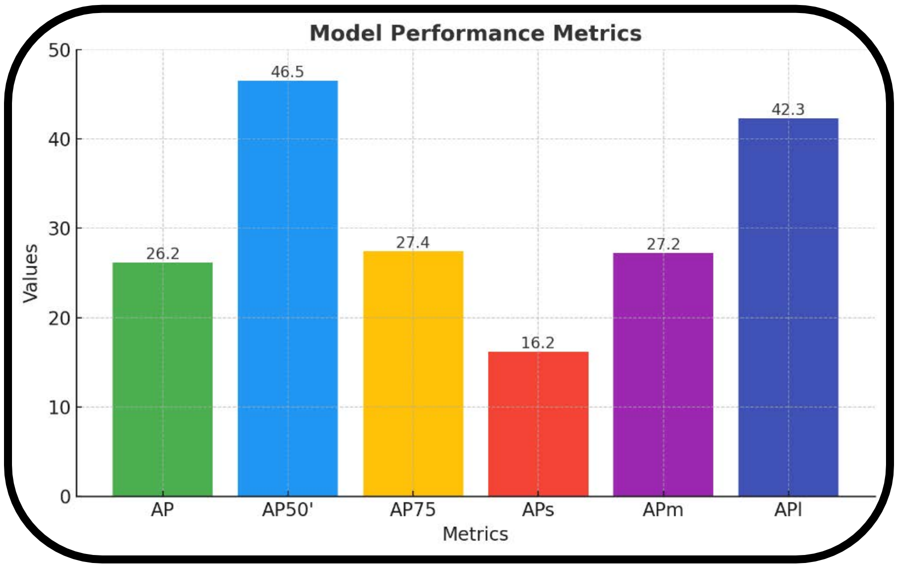
AI Rust Detection

During the final phase of the project, we conducted EDA, data preparation, and modelling for the chosen Rust Detection project. For this purpose, we created a dedicated GitHub repository within the MRR environment. We imported images from the RoboFlow dataset and ran the Detectron2 model, which had also been used for Trash Detection. Unfortunately, the results fell short of expectations due to the limited availability of data.

Online data sources containing drone images of large, rusted machines are scarce. The only dataset we found featured rust on various types of iron, with images captured from different perspectives. This lack of diverse and specific data reduced the model's robustness and made generalization challenging, as the dataset failed to cover the wide variety of possible scenarios.

An alternative approach involved extracting and annotating data from the 3D platform ourselves. However, this was not feasible within the current timeline due to the 3D portal's inability to allow bulk downloads. Manually downloading and annotating each image would have been too time-consuming. Consequently, it is recommended that another working group take on this project to achieve better results.

The current model's performance highlights its limitations. The highest metric achieved was 46.5%, indicating that nearly half of the labelled objects were not predicted at all. The model struggles significantly with small objects (APs), as expected, while showing better performance with medium (APm) and large (APl) labels. At this stage, the model is largely guessing, which reflects the limitations of the dataset.



To enhance the model's predictive capabilities, future efforts should focus on expanding and improving the dataset, as this would have the greatest impact on the metrics. While hyperparameter tuning could offer some improvements, database expansion or renewal would provide a far more substantial boost. Despite these limitations, the model shows promise, particularly for applications targeting larger objects.

9. Conclusion

By answering the questions, we got a much better picture of MRR and the systems that are used within MRR. For example, we know which services MRR offers, and we also got a better picture of the 3D Portal that is used in combination with their API. Answering these questions was essential to know where we stood and how we could improve the current model within the policy of the organization.

Answering the questions about the data gave us a good picture of the data that was used. This helped us to concluded that more images of different surfaces and with different waste types are needed to make the model more generalizable.

Regarding the questions of improving the performance of the (new) model, we concluded that we succeeded in finding better alternatives for previous models and in improving their performance metrics. The previous YOLOV8 model was improved with YOLOV11 in terms of recall and precision and later, two new models, RT-DETR & Detectron2, were added for this project, which work much better than the model of the other group (YOLOV8). These efforts and increasing the dataset have helped making a more robust model.

These two new models have also been implemented in the API, which connects to the MRR portal. Within this API, two different docker containers can be opened, each with its own model. This can be used to run our model on images of the tasks on the 3D portal.

We have made a start with a new project on rust detection with drone images. However, this project faced challenges due to limited online data availability, particularly drone images of large machines with rust and lack of diverse datasets that hindered model robustness and generalization. Extracting and annotating the data from the 3D platform was considered, but the inability to bulk download images and the required annotation time made this infeasible within the current available time. Future groups could address these issues to achieve better results.

In conclusion, we have learned that the power of artificial intelligence can efficiently detect trash to tackle the current, inefficient detection of it in an innovative way. It is important to note that AI Trash Detection is only a tool, which in the hands of municipalities and environmental companies is an important tool to tackle this problem faster and more efficiently. This, so that we can work towards a more sustainable future in which smart technology like this contributes to a cleaner planet.

Together we can make a difference, one piece of waste at a time.

10. Recommendations

From our findings from our research document as well as our testing of the models, we would like to propose the following recommendations for the continuation of this project.

The following is a summary of the key recommendations made in Chapter 6 of this document. A more detailed explanation can be found in said chapter:

- Adding new data
  - More Variation in Surface Types
  - Expansion of Waste Types
  - Different Light and Weather Conditions
  - Use of Angle and Height Variations
  - Data Annotations
- Searching for more balanced data with waste and people

Regarding the rust detection project, we would like to recommend the following:

- Take drone images with rust on it and manually annotate the rust in the images.

The reason for this is that the images available on the internet don't have good quality annotations and are not taken from a desired perspective (drone perspective.) Creating this manually is necessary to ensure the model is trained on the correct data for reliable detection of rust.

We have also concluded that RT-DETR still works better than Detectron2. This is because RT-DETR works easier than Detectron2 and does not require a GPU, which makes it much more flexible. Finally, we also think that adding additional data can improve the models further.

References

Farzad. (2023, October). *Pothole_Segmentation_YOLOv8 Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/farzad/pothole_segmentation_yolov8

Innercore. (2024, January). *Corrosion detection Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/innercore/corrosion-detection-bczan

KaraAgro AI Foundation. (2023). *Drone-based-Agricultural-Dataset-for-Crop-Yield-Estimation (Revision 2530d1d)*. Retrieved from Hugging Face: https://huggingface.co/datasets/KaraAgroAI/Drone-based-Agricultural-Dataset-for-Crop-Yield-Estimation

Kraft, M., Piechocki, M., Ptak, B., & Walas, K. (2021). Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle. *MDPI*.

morningstar. (2024, November). *blue-algae-seg Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/morningstar/blue-algae-seg

Punthon. (2024, May). *oil-spill-segmentation Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/punthon-i5rmx/oil-spill-segmentation-tysge

PUTvision. (2023, September 26). *UAVVaste*. Retrieved from GitHub: https://github.com/PUTvision/UAVVaste

Silva, L. A. (2023, March). *crddc22_china_spain_uav Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/luis-augusto-silva-bq4bv/crddc22_china_spain_uav

Slachtofferwijzer. (2024). *Feiten en cijfers van vermissingen in Nederland*. Retrieved from Slachtofferwijzer: https://slachtofferwijzer.nl/artikelen/feiten-cijfers-vermissingen-nederland

TFMAI. (2024, April). *Search for missing people Computer Vision Project*. Retrieved from Roboflow: https://universe.roboflow.com/tfmai/search-for-missing-people

Yohanandan, S. (2020, June 5). *What is Mean Average Precision (MAP) and how does it work*. Retrieved from Xailient: https://xailient.com/blog/what-is-mean-average-precision-and-how-does-it-work/

Appendix

Appendix A: Model results as of October 10th, 2024

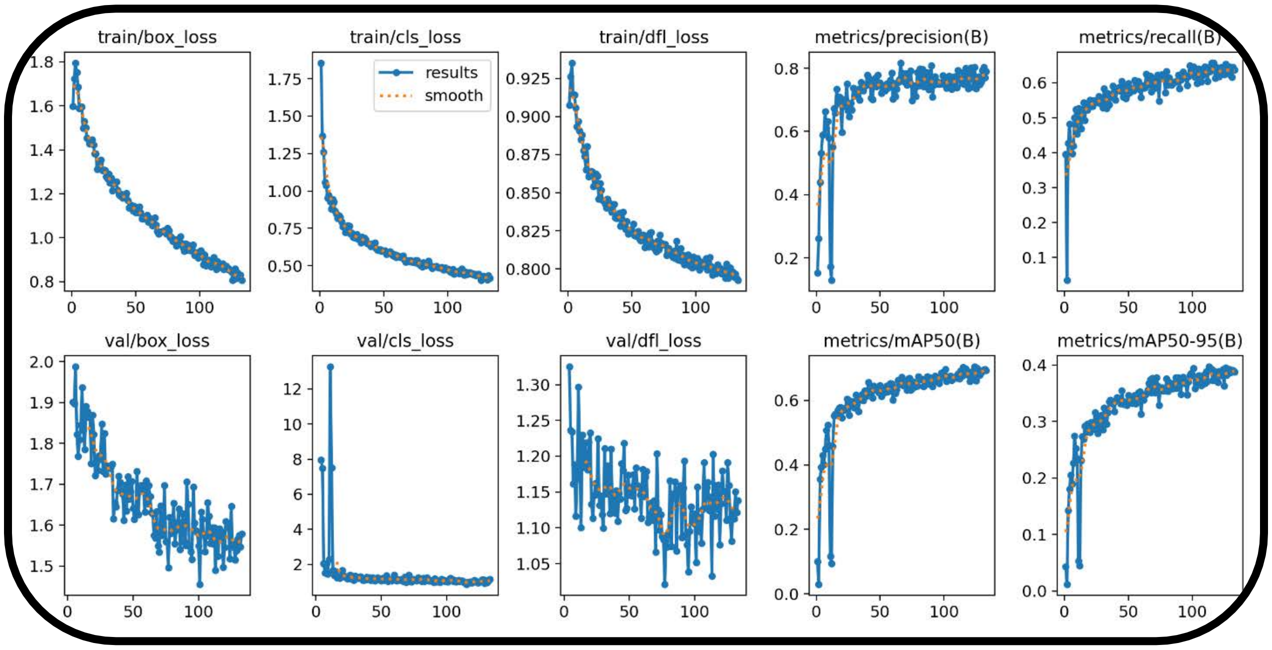
As of October 10th, 2024, we have trained two different models, both trained on two different datasets. The two models used in this phase of the project are YOLO and RT-DETR in their base form. This is done with the intention of testing the performance of the models as is and to determine which of the two will be used in the next steps of the project. The first model of both YOLO and RT-DETR consists of all the images of the UAVVaste and MRR Drones dataset, while the second model has the GOPR images removed from the dataset, due to a data issue where the images were turned upside down while the annotations kept the original orientation. On October 11th, 2024, the issue with these images were fixed.

Model Evaluations

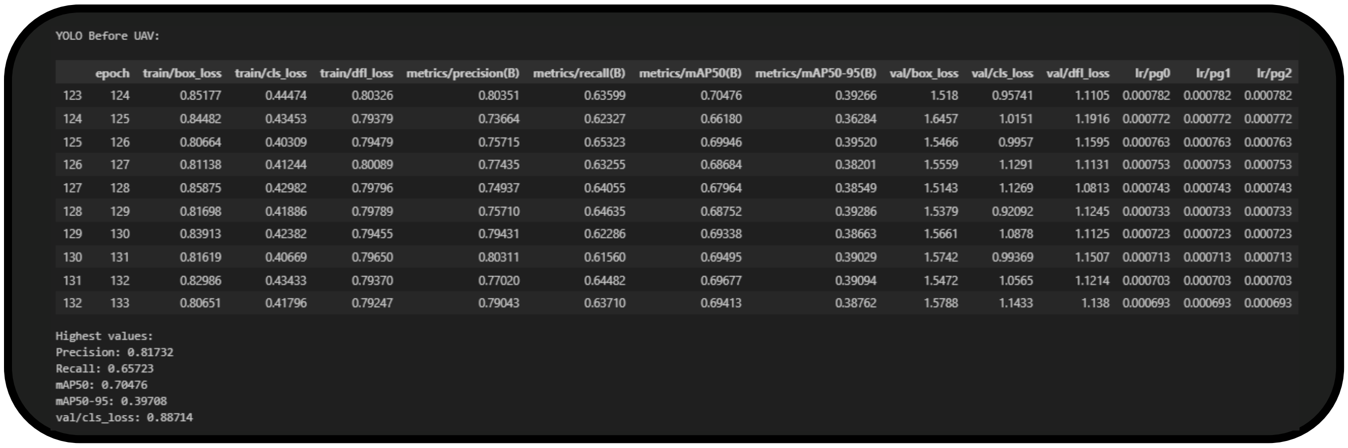
Down here, the graphs and metrics are displayed with some observational remarks. Due to the different metrics for loss being used between the YOLO and RT-DETR models, only the metrics that are present for both models will be discussed.

YOLOv11 Baseline model (with GOPR images)

Evaluation graphs:



Evaluation metrics:



```
YOLO Before UAV:
```

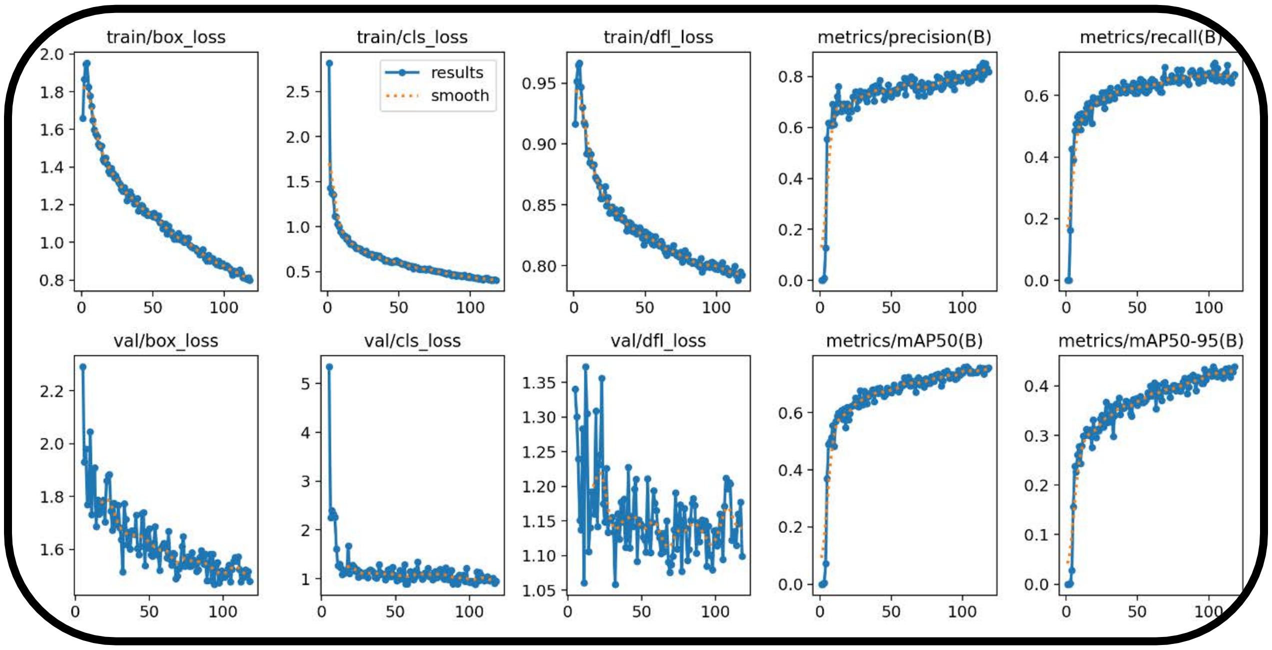
	epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
123	124	0.85177	0.44474	0.80326	0.80351	0.63599	0.70476	0.39266	1.518	0.95741	1.1105	0.000782	0.000782	0.000782
124	125	0.84482	0.43453	0.79379	0.73664	0.62327	0.66180	0.36284	1.6457	1.0151	1.1916	0.000772	0.000772	0.000772
125	126	0.80664	0.40309	0.79479	0.75715	0.65323	0.69946	0.39520	1.5466	0.9957	1.1595	0.000763	0.000763	0.000763
126	127	0.81138	0.41244	0.80089	0.77435	0.63255	0.68684	0.38201	1.5559	1.1291	1.1131	0.000753	0.000753	0.000753
127	128	0.85875	0.42982	0.79796	0.74937	0.64055	0.67964	0.38549	1.5143	1.1269	1.0813	0.000743	0.000743	0.000743
128	129	0.81698	0.41886	0.79789	0.75710	0.64635	0.68752	0.39286	1.5379	0.92092	1.1245	0.000733	0.000733	0.000733
129	130	0.83913	0.42382	0.79455	0.79431	0.62286	0.69338	0.38663	1.5661	1.0878	1.1125	0.000723	0.000723	0.000723
130	131	0.81619	0.40669	0.79650	0.80311	0.61560	0.69495	0.39029	1.5742	0.99369	1.1507	0.000713	0.000713	0.000713
131	132	0.82986	0.43433	0.79370	0.77020	0.64482	0.69677	0.39094	1.5472	1.0565	1.1214	0.000703	0.000703	0.000703
132	133	0.80651	0.41796	0.79247	0.79043	0.63710	0.69413	0.38762	1.5788	1.1433	1.138	0.000693	0.000693	0.000693

```
Highest values:
Precision: 0.81732
Recall: 0.65723
mAP50: 0.70476
mAP50-95: 0.39708
val/cls_loss: 0.88714
```

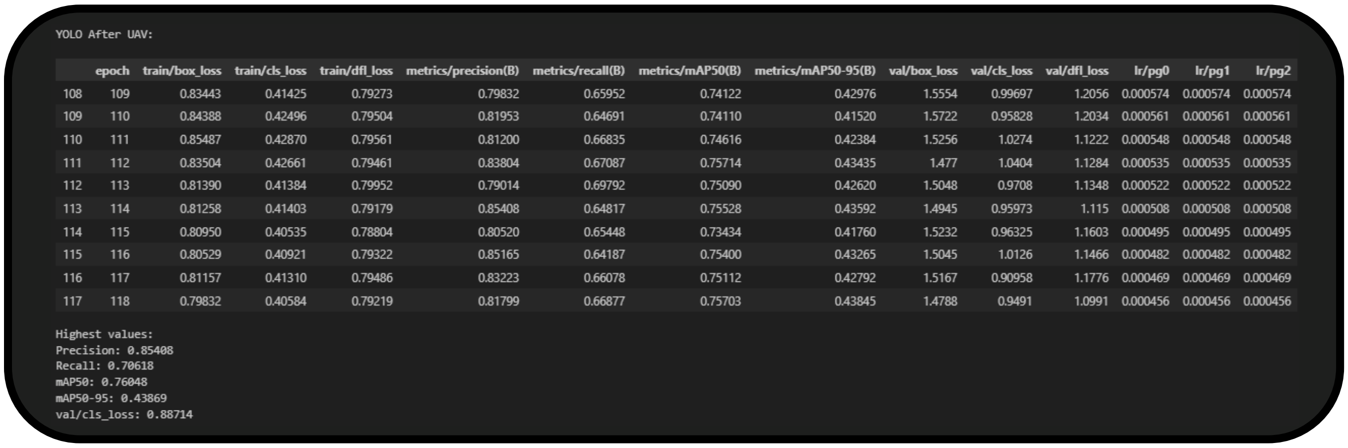
Precision of ~0.81732 and recall of 0.65723, meaning a higher false negative rate compared to false positives. As a lower amount of false negatives is desired by the stakeholder Sieuwe, this the lower recall score is undesirable. The graph shows that the precision stops significantly increasing from around epoch 70, while the recall stops showing improvements at around epoch 115. The mAP50 score is 0.70476, indicating an accuracy of that number when requiring an IoU of 50%. When taking an IoU of 50-95%, the mAP50-95 score drops to 0.39708, meaning that the around 30% of the predictions do not surpass the IoU threshold ranging from 50-95%. The cls_loss is high, with the lowest score of 0.88714, meaning the error between predicting trash and background is high.

YOLOv11 (without GOPR images)

Evaluation graphs:



Evaluation metrics:



YOLO After UAV:

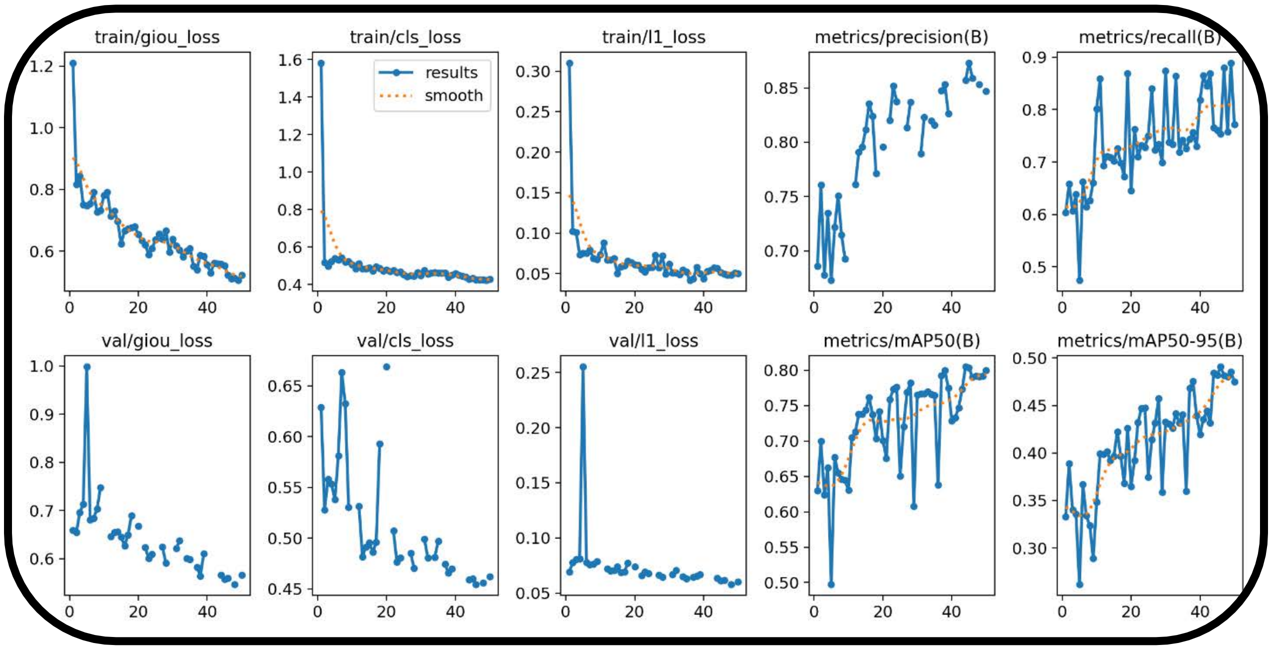
	epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
108	109	0.83443	0.41425	0.79273	0.79832	0.65952	0.74122	0.42976	1.5554	0.99697	1.2056	0.000574	0.000574	0.000574
109	110	0.84388	0.42496	0.79504	0.81953	0.64691	0.74110	0.41520	1.5722	0.95828	1.2034	0.000561	0.000561	0.000561
110	111	0.85487	0.42870	0.79561	0.81200	0.66835	0.74616	0.42384	1.5256	1.0274	1.1222	0.000548	0.000548	0.000548
111	112	0.83504	0.42661	0.79461	0.83804	0.67087	0.75714	0.43435	1.477	1.0404	1.1284	0.000535	0.000535	0.000535
112	113	0.81390	0.41384	0.79952	0.79014	0.69792	0.75090	0.42620	1.5048	0.9708	1.1348	0.000522	0.000522	0.000522
113	114	0.81258	0.41403	0.79179	0.85408	0.64817	0.75528	0.43592	1.4945	0.95973	1.115	0.000508	0.000508	0.000508
114	115	0.80950	0.40535	0.78804	0.80520	0.65448	0.73434	0.41760	1.5232	0.96325	1.1603	0.000495	0.000495	0.000495
115	116	0.80529	0.40921	0.79322	0.85165	0.64187	0.75400	0.43265	1.5045	1.0126	1.1466	0.000482	0.000482	0.000482
116	117	0.81157	0.41310	0.79486	0.83223	0.66078	0.75112	0.42792	1.5167	0.90958	1.1776	0.000469	0.000469	0.000469
117	118	0.79832	0.40584	0.79219	0.81799	0.66877	0.75703	0.43845	1.4788	0.9491	1.0991	0.000456	0.000456	0.000456

```
Highest values:
Precision: 0.85408
Recall: 0.70618
mAP50: 0.76048
mAP50-95: 0.43869
val/cls_loss: 0.88714
```

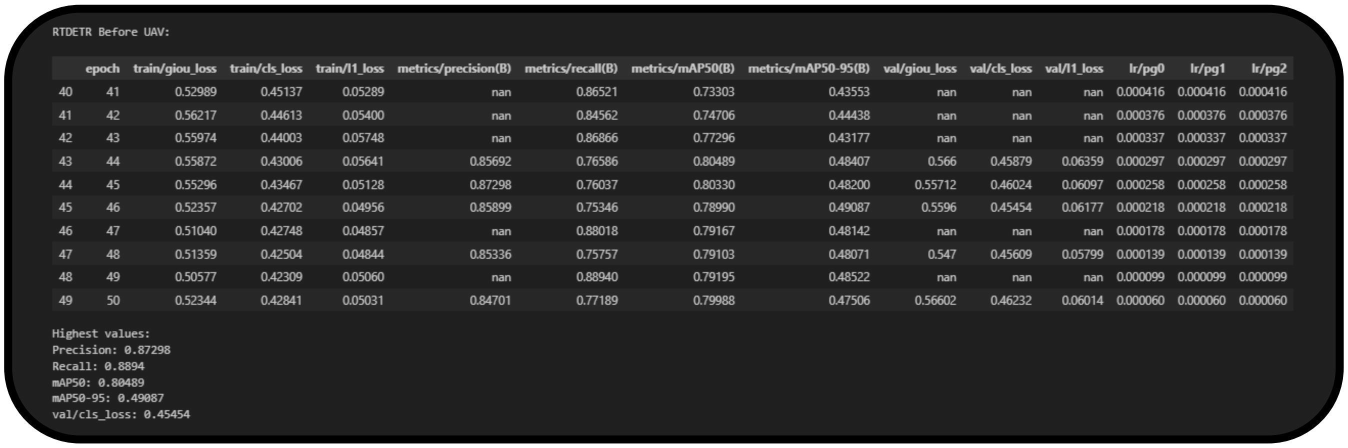
Precision of 0.85408 and 0.70618, which is around 4% and 5% higher respectively compared to the base model. Removing the flipped GOPR images likely helped with avoiding training on wrong data. The precision seems to still show improvement at the final epoch, while the recall seems to have stagnated at around epoch 100. The mAP50 and mAP50-95 scores are now 0.76048 and 0.43869, which is 6% and 4% higher respectively compared to the base model. The cls_loss score did not improve, having the exact same score of 0.88714.

RT-DETR (with GOPR images)

Evaluation graphs:



Evaluation metrics:



```
RTDETR Before UAV:
```

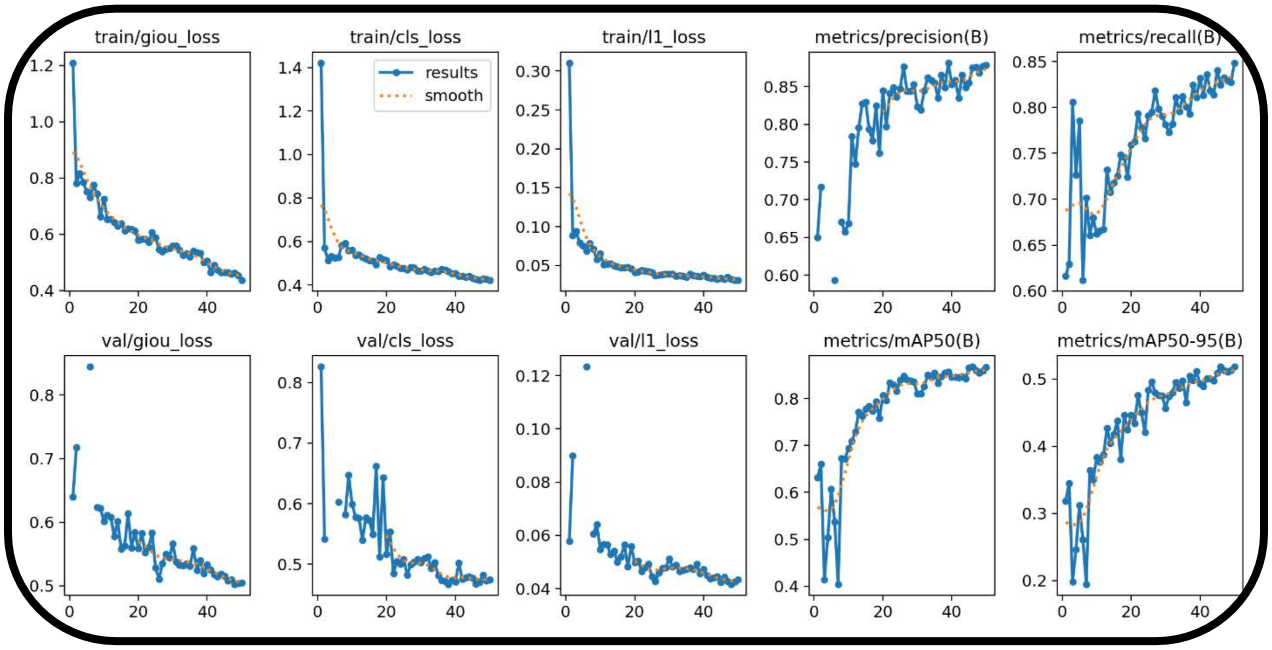
	epoch	train/giou_loss	train/cls_loss	train/l1_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/giou_loss	val/cls_loss	val/l1_loss	lr/pg0	lr/pg1	lr/pg2
40	41	0.52989	0.45137	0.05289	nan	0.86521	0.73303	0.43553	nan	nan	nan	0.000416	0.000416	0.000416
41	42	0.56217	0.44613	0.05400	nan	0.84562	0.74706	0.44438	nan	nan	nan	0.000376	0.000376	0.000376
42	43	0.55974	0.44003	0.05748	nan	0.86866	0.77296	0.43177	nan	nan	nan	0.000337	0.000337	0.000337
43	44	0.55872	0.43006	0.05641	0.85692	0.76586	0.80489	0.48407	0.566	0.45879	0.06359	0.000297	0.000297	0.000297
44	45	0.55296	0.43467	0.05128	0.87298	0.76037	0.80330	0.48200	0.55712	0.46024	0.06097	0.000258	0.000258	0.000258
45	46	0.52357	0.42702	0.04956	0.85899	0.75346	0.78990	0.49087	0.5596	0.45454	0.06177	0.000218	0.000218	0.000218
46	47	0.51040	0.42748	0.04857	nan	0.88018	0.79167	0.48142	nan	nan	nan	0.000178	0.000178	0.000178
47	48	0.51359	0.42504	0.04844	0.85336	0.75757	0.79103	0.48071	0.547	0.45609	0.05799	0.000139	0.000139	0.000139
48	49	0.50577	0.42309	0.05060	nan	0.88940	0.79195	0.48522	nan	nan	nan	0.000099	0.000099	0.000099
49	50	0.52344	0.42841	0.05031	0.84701	0.77189	0.79988	0.47506	0.56602	0.46232	0.06014	0.000060	0.000060	0.000060

```
Highest values:
Precision: 0.87298
Recall: 0.8894
mAP50: 0.80489
mAP50-95: 0.49087
val/cls_loss: 0.45454
```

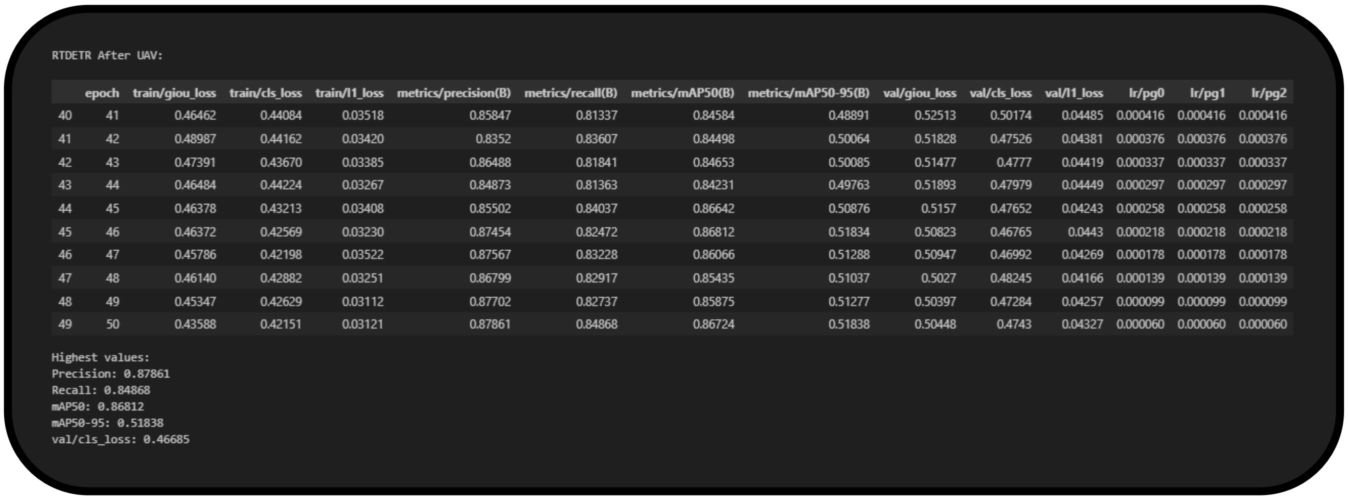
Precision of 0.87298 and 0.8894, which is an improvement over both YOLO models with less than half the epochs (133 and 118 vs 50 epochs). Compared to the baseline, it's 6% and 23% higher respectively. The significant improvement in recall aligns with the desired result of the stakeholder of having as few false negatives as possible. The graphs do show some irregularities with many nan values or large drops in score over multiple graphs. This is likely caused by the flipped GOPR images within the dataset. The mAP50 and mAP50-95 scores are also better compared to both YOLO models, scoring 10% higher in both compared to the baseline. The cls_val is a lot better than the previous YOLO models, with this model scoring 0.45454, which is about 0.43 lower, indicating less errors in the prediction of the 2 classes.

RT-DETR (without GOPR images)

Evaluation graphs:



Evaluation metrics:



```
RTDETR After UAV:
```

	epoch	train/giou_loss	train/cls_loss	train/l1_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/giou_loss	val/cls_loss	val/l1_loss	lr/pg0	lr/pg1	lr/pg2
40	41	0.46462	0.44084	0.03518	0.85847	0.81337	0.84584	0.48891	0.52513	0.50174	0.04485	0.000416	0.000416	0.000416
41	42	0.48987	0.44162	0.03420	0.8352	0.83607	0.84498	0.50064	0.51828	0.47526	0.04381	0.000376	0.000376	0.000376
42	43	0.47391	0.43670	0.03385	0.86488	0.81841	0.84653	0.50085	0.51477	0.4777	0.04419	0.000337	0.000337	0.000337
43	44	0.46484	0.44224	0.03267	0.84873	0.81363	0.84231	0.49763	0.51893	0.47979	0.04449	0.000297	0.000297	0.000297
44	45	0.46378	0.43213	0.03408	0.85502	0.84037	0.86642	0.50876	0.5157	0.47652	0.04243	0.000258	0.000258	0.000258
45	46	0.46372	0.42569	0.03230	0.87454	0.82472	0.86812	0.51834	0.50823	0.46765	0.0443	0.000218	0.000218	0.000218
46	47	0.45786	0.42198	0.03522	0.87567	0.83228	0.86066	0.51288	0.50947	0.46992	0.04269	0.000178	0.000178	0.000178
47	48	0.46140	0.42882	0.03251	0.86799	0.82917	0.85435	0.51037	0.5027	0.48245	0.04166	0.000139	0.000139	0.000139
48	49	0.45347	0.42629	0.03112	0.87702	0.82737	0.85875	0.51277	0.50397	0.47284	0.04257	0.000099	0.000099	0.000099
49	50	0.43588	0.42151	0.03121	0.87861	0.84868	0.86724	0.51838	0.50448	0.4743	0.04327	0.000060	0.000060	0.000060

```
Highest values:
Precision: 0.87861
Recall: 0.84868
mAP50: 0.86812
mAP50-95: 0.51838
val/cls_loss: 0.46685
```

Precision of 0.87861 and recall of 0.84686, which is a small increase in precision and a decrease in recall score compared to the previous RT-DETR model. Precision is almost the same, improving by around 0.6%, while the recall is reduced by 4%. Both precision and recall do still show an increase in score in the graphs, so it’s likely still able to improve with longer training times. The mAP50 and mAP50-95 do have an increase compared to the previous model, now scoring 0.86812 and 0.51838. This is an improvement of 6% and 3% respectively. The cls_loss is slightly worse though, scoring around 0.01 higher.

Evaluation comparison

Down here is a compilation of the best metrics of the 4 models displayed within a table.

Black	*= baseline/equal*
Green	*= better (compared to baseline)*
Blue	*= best*
Orange	*= worse (compared to baseline)*
Red	*= worst*

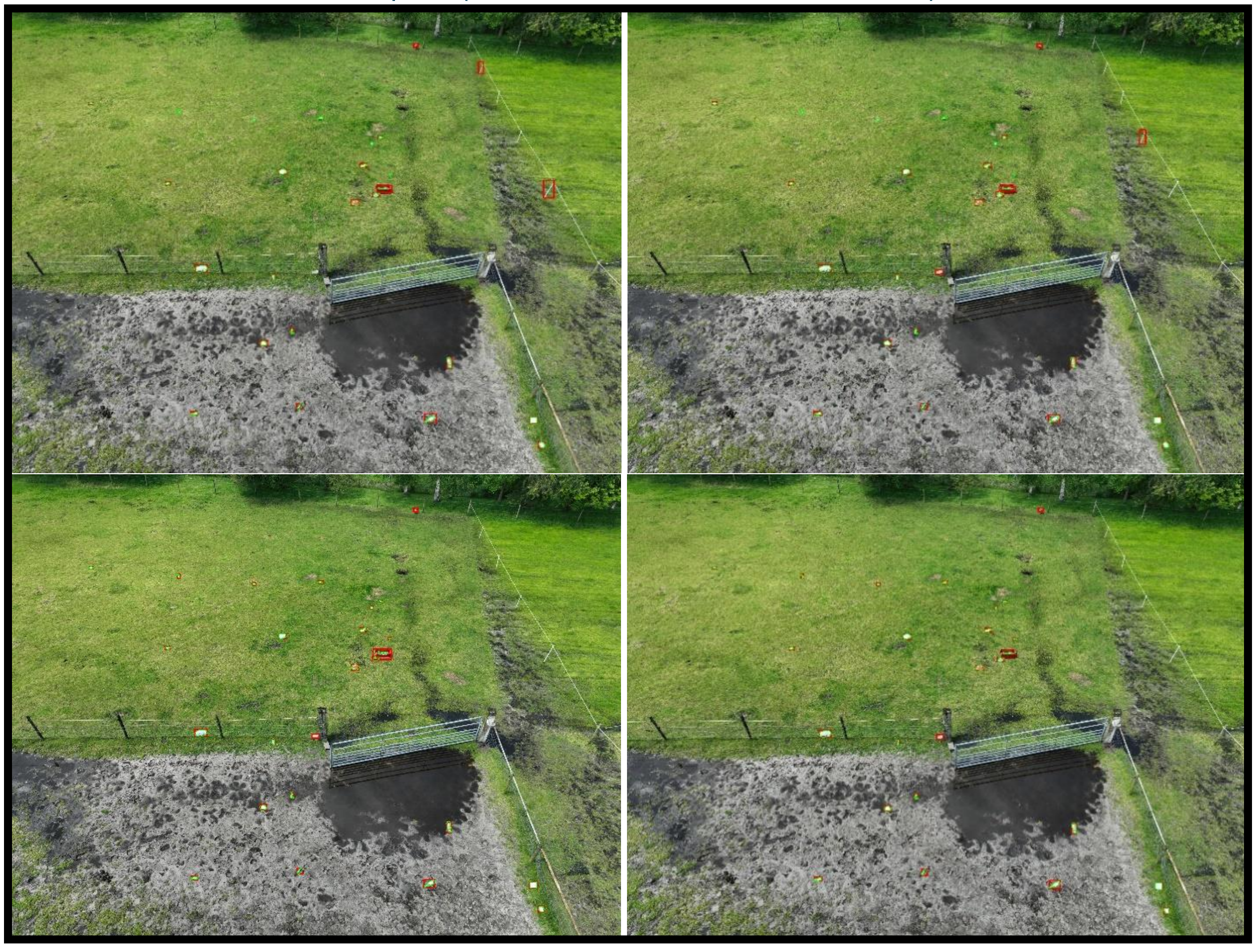
Model\Metric	**Precision**	**Recall**	**mAP50**	**mAP50-95**	**cls_loss**
YOLOv11 (with GOPR)	0.81732	0.65723	0.70476	0.39708	0.88714
YOLOv11 (without GOPR)	0.85408	0.70618	0.76048	0.43869	0.88714
RT-DETR (with GOPR)	0.87298	0.8894	0.80489	0.49087	0.45454
RT-DETR (without GOPR)	0.87861	0.84868	0.86812	0.51838	0.46685

Visual examples

Down here are some examples predictions by the 4 models with some notable predictions. The green shapes are the annotations from the dataset and the red boxes are the predictions made by the models. The layout of the images is:

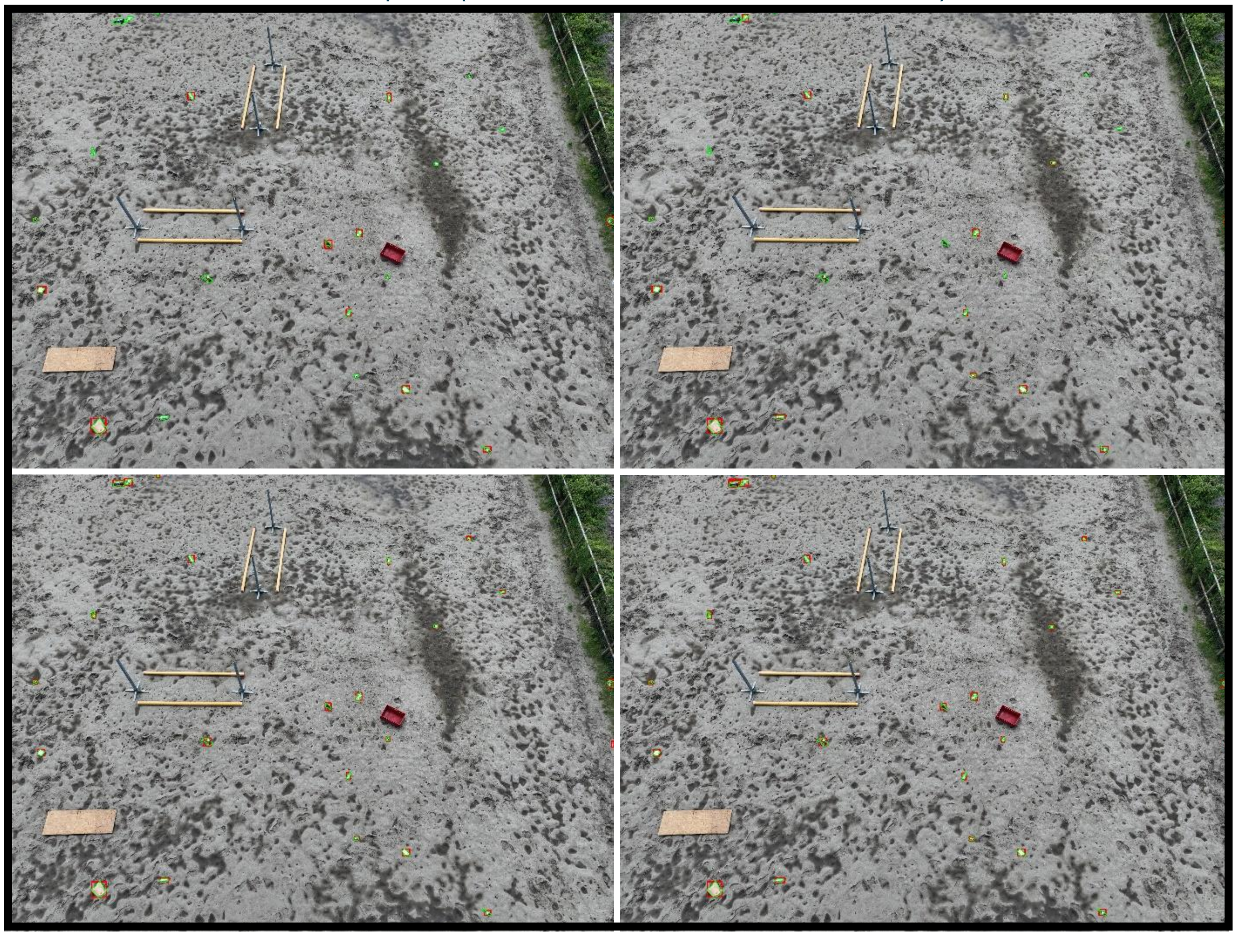
- Top-left: YOLO (with GOPR images) [Model1]
- Top-right: YOLO (without GOPR images) [Model2]
- Bottom-left: RT-DETR (with GOPR images) [Model3]
- Bottom-right: RT-DETR (without GOPR images) [Model4]

Example 1 (DJI_20240617130035_0036_V.JPG)



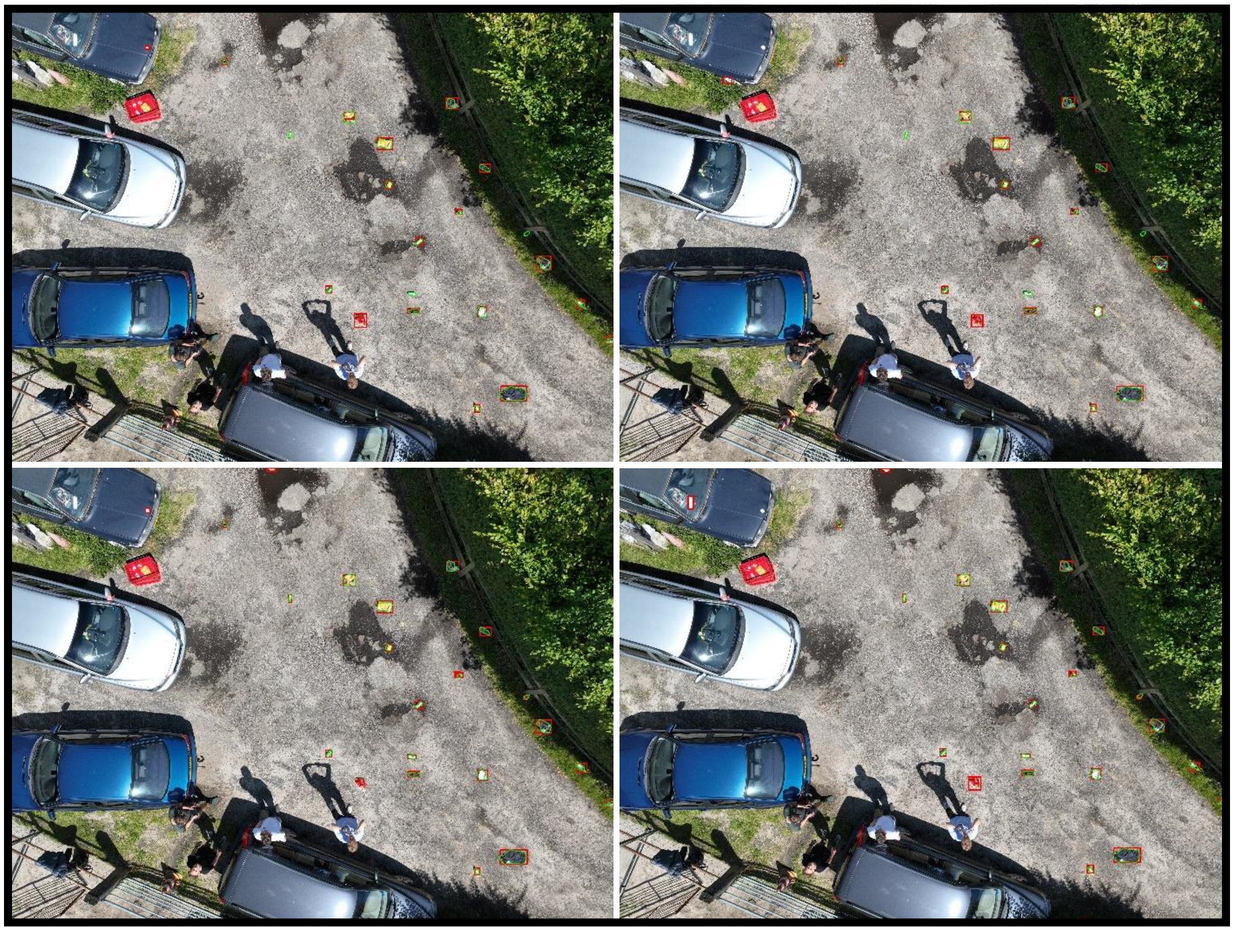
The YOLO models seem to struggle with photo's that aren't oriented straight to the ground. Lots of the trash objects in the distance are missed by the YOLO model, while the RT-DETR model predicted them.

Example 2 (DJI_20240617130214_0048_V.JPG)



Same as example 1, where the YOLO model struggles with small objects in the distance with tilted photos, while RT-DETR detects them without issue.

Example 3 (DJI_20240617143418_0002_V.JPG)



The YOLO models are missing some of the smaller cans and bottles, while the RT-DETR models detect them. The models that have the GOPR data ([Model1] & [Model3]), the badge of the car in the top-left is wrongly predicted as trash. The models without the GOPR data do not predict the badge as trash but have other wrongly predicted objects, like the car rim in [Model2] and a white object in the car in [Model4].

Appendix B: Model results as of January 7th, 2025

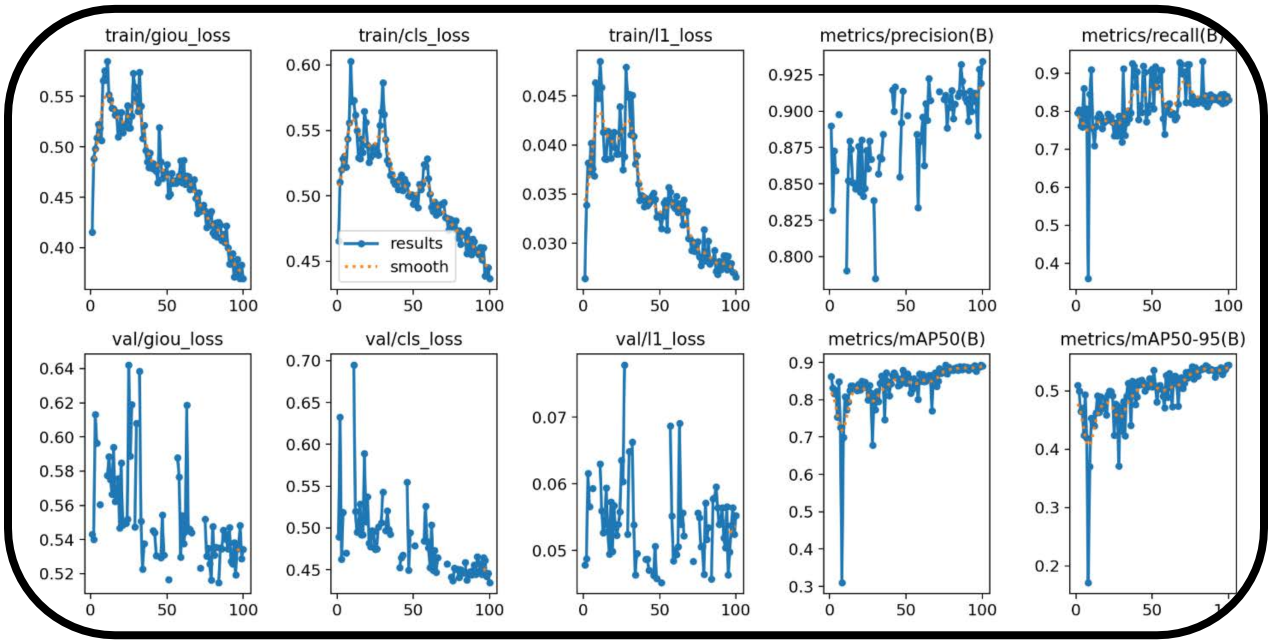
As of January 7th, 2025, we have made multiple iterations on the RT-DETR model as well as multiple iterations of the new Detectron2 models. Both these models have had hyperparameter tuning applied to them, where the RT-DETR model showed improvements, while the Detectron2 model did not improve with our hyperparameter tuning. Both these models are also trained with enriched datasets, having images with people added to train the models to differentiate humans from trash. The old YOLOv11 models were not improved, as they underperformed compared to the other 2 models.

Model evaluations

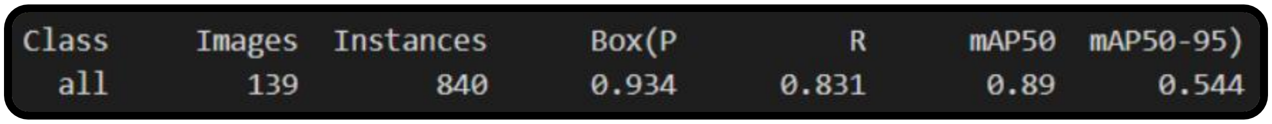
Down here, the graphs and metrics are displayed with some observational remarks on the best RT-DETR and Detectron2 models since the last comparison. Due to the lack of many of the evaluation metrics within Detectron2 and the difficulty of implementing said metrics, the metrics that can be compared are limited to the (m)AP values.

RT-DETR (enriched data with images humans)

Evaluation graphs:



Evaluation metrics (bounding boxes):



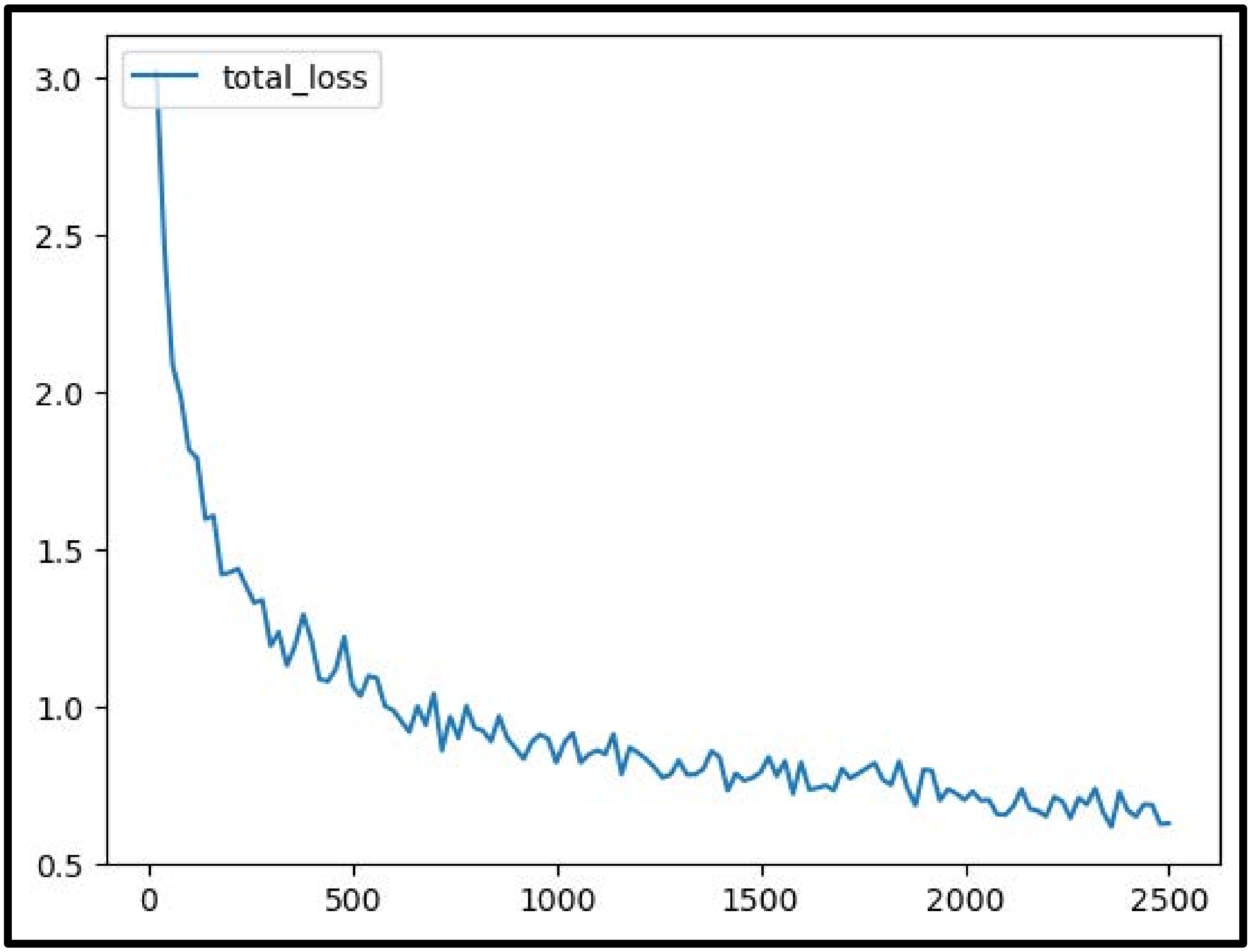
```
Class     Images  Instances      Box(P          R      mAP50  mAP50-95)
  all        139        840      0.934      0.831       0.89      0.544
```

Compared to the previous models in Appendix A, the new model does score slightly lower in the recall department, only scoring 0.831 compared to last time's best score of 0.8894. However, the precision, mAP50 and mAP50-95 are higher compared to the highest scores of last time, where the scores are now 0.934, 0.89 and 0.554 instead of 0.87861, 0.86812 and 0.51838 respectively. The cls_loss also sees a small improvement, now reaching somewhere around 0.43 instead of the previous 0.45454.

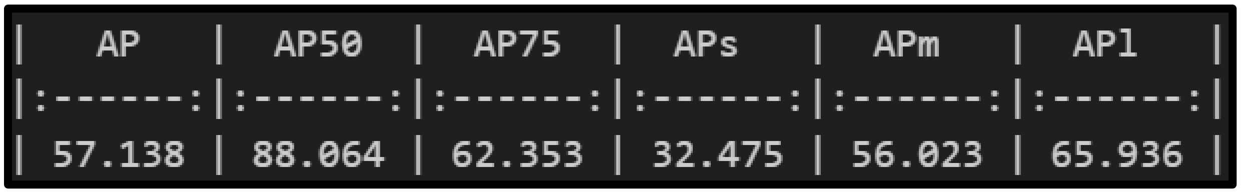
Detectron2 (enriched data with images humans)

Note: as stated before, Detectron2 has less evaluation metrics built into the model, and the lack of documentation of how to access/implement these metrics, these will be limited to AP scores and a general loss graph for this model.

Evaluation graph:

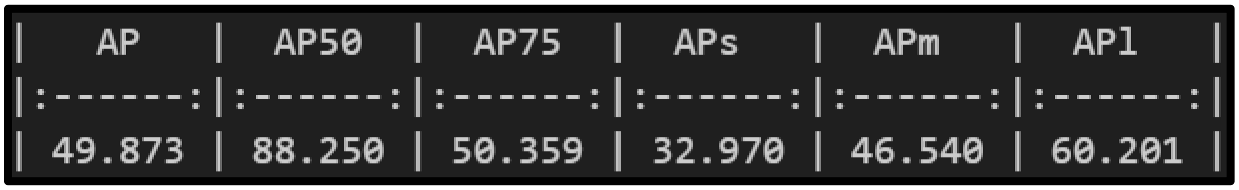


Evaluation metrics (bounding boxes):



```
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 57.138 | 88.064 | 62.353 | 32.475 | 56.023 | 65.936 |
```

Evaluation metrics (segmentations):



```
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 49.873 | 88.250 | 50.359 | 32.970 | 46.540 | 60.201 |
```

There are a lot less metrics to work with in the Detectron2 evaluation. The only comparable metric within Detectron2 is (m)AP50 and (m)AP(50-95). According to (Yohanandan, 2020), there are no distinctions made between AP and mAP in certain cases, and in the documentation from (cocodataset, n.d.), their AP metric is the AP with an IoU ranging from 5-95, which is the same as the mAP50-95 metric from the YOLO and RT-DETR models. It is assumed that these two are the same. The loss metrics were also not built-in, and we were only able to extract the total loss, which is not one of the metrics in the other models.

For the AP50 metric, it scores higher than the previous models in the evaluation in Appendix A, scoring 0.88064 on bounding boxes and 0.88250 on segmentations. The AP scores are 0.57138 and 0.49873. When comparing the bounding box metrics, the Detectron2 scores higher than all other models up to this point, given that all other models are also using bounding box predictions in our inferences. The segmentation AP is quite a bit lower, scoring 0.07265 lower compared to the bounding box counterpart, while the difference in AP50 scores is a lot smaller, only differing 0.00186, with the segmentation one scoring better.

The Detectron2 metrics do add extra metrics which evaluate the AP scores for small (APs), medium (APm) and large (APl) sized objects. In both cases, smaller objects have more difficulty in being correctly detected by the Detectron2 model.

Evaluation comparison

Down here is a compilation of the best metrics of the 2 new models along with the best model from the previous evaluation (Model4) as baseline displayed within a table.

Black	*= baseline/equal*
Green	*= better (compared to baseline)*
Blue	*= best*
Orange	*= worse (compared to baseline)*
Red	*= worst*

Model\Metric	**Precision**	**Recall**	**mAP50**	**mAP50-95**	**cls_loss**
RT-DETR (without GOPR)	0.87861	0.84868	0.86812	0.51838	0.46685
RT-DETR (extra images)	0.934	0.831	0.89	0.544	N.A.
Detectron2 (bbox)	N.A.	N.A.	0.88064	0.57138	N.A.
Detectron2 (segments)	N.A.	N.A.	0.88250	0.49873	N.A.

Visual examples

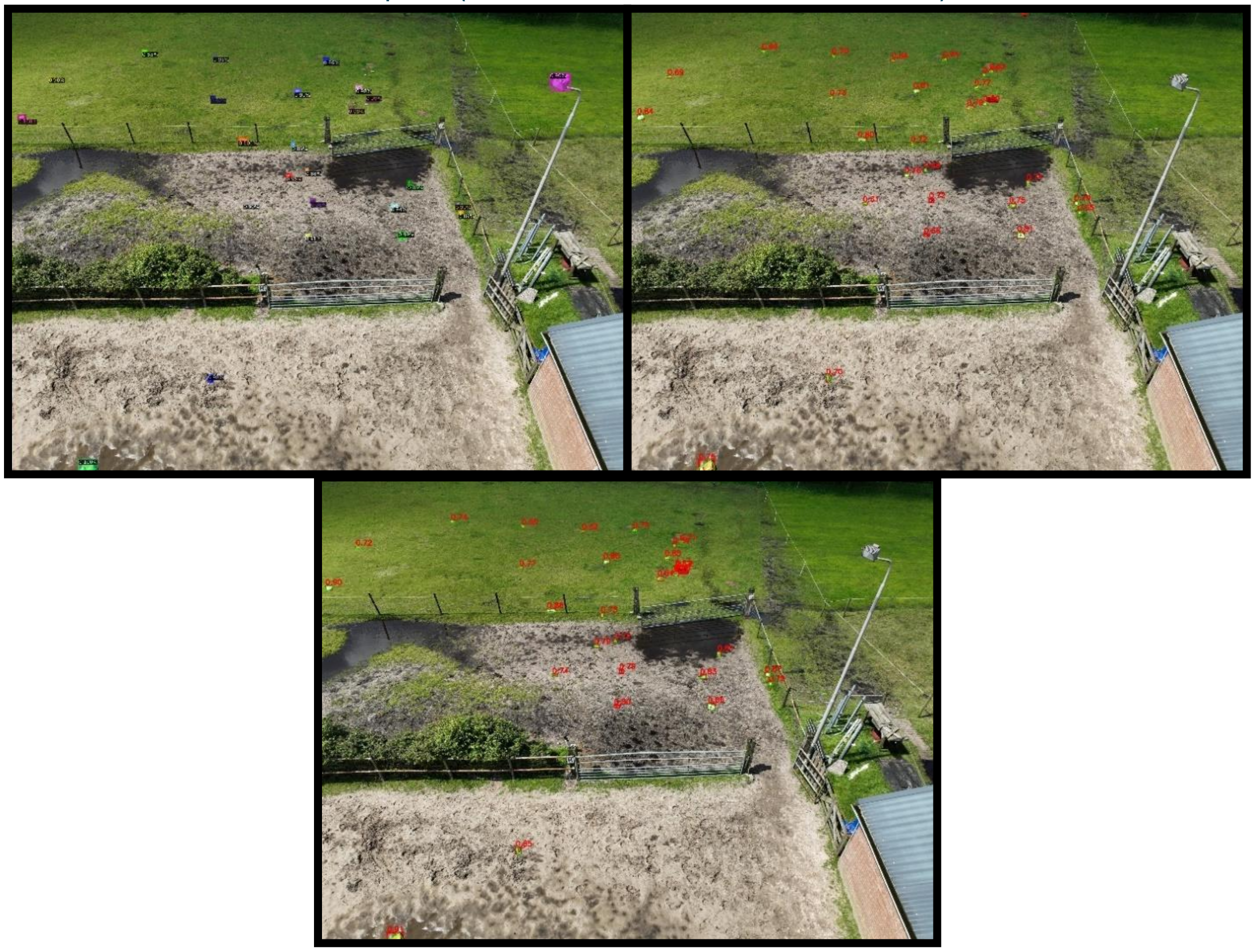
Down here are some examples predictions by the 3 models with some notable predictions. The green shapes are the annotations from the dataset and the red boxes/shapes are the predictions made by the models.

Note: As the newer models were trained with an additional dataset, the distribution of train/val/test are also different, with each set now containing different images compared to the previous comparison. As this is the case, the inference on the old model [Model4] is redone with the test set of the newer models, with the caveat that the predictions are made on images that the model was trained on. Something to keep in mind in this comparison for [Model4].

The layout of the images is:

- Top-left: RT-DETR (without GOPR images) [Model4]
- Top-right: RT-DETR (enriched data with images humans) [Model5]
- Bottom-left: Detectron2 (enriched data with images humans) [Model6]

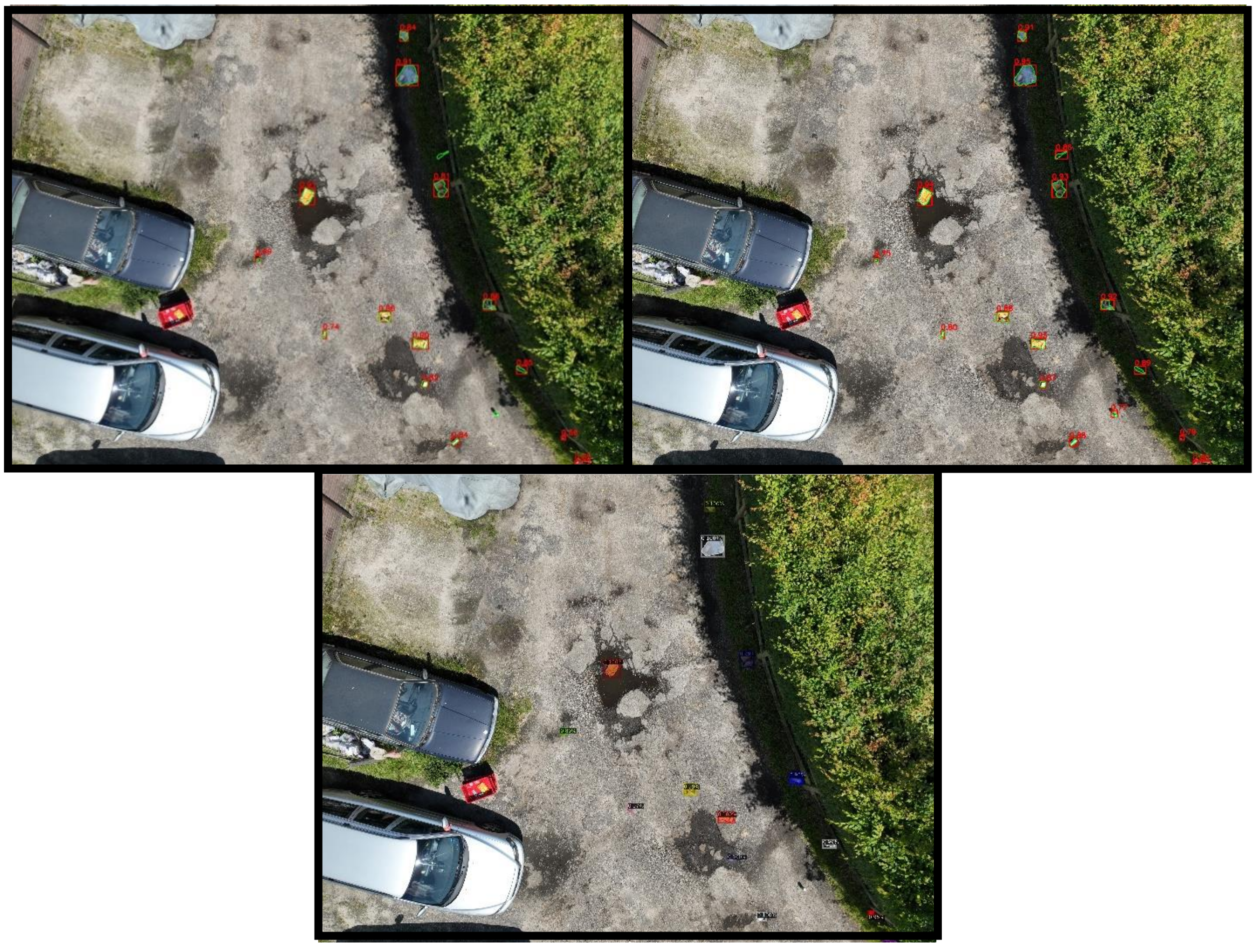
Example 1 (DJI_20240617130029_0035_V.JPG)



Comparing the two RT-DETR models [Model4 & Model5], the objects predicted don't seem too different from each other, besides one false positive in the far background in the old [Model4]. The newer [Model5] does have higher confidence scores for each prediction compared to the old [Model4], however.

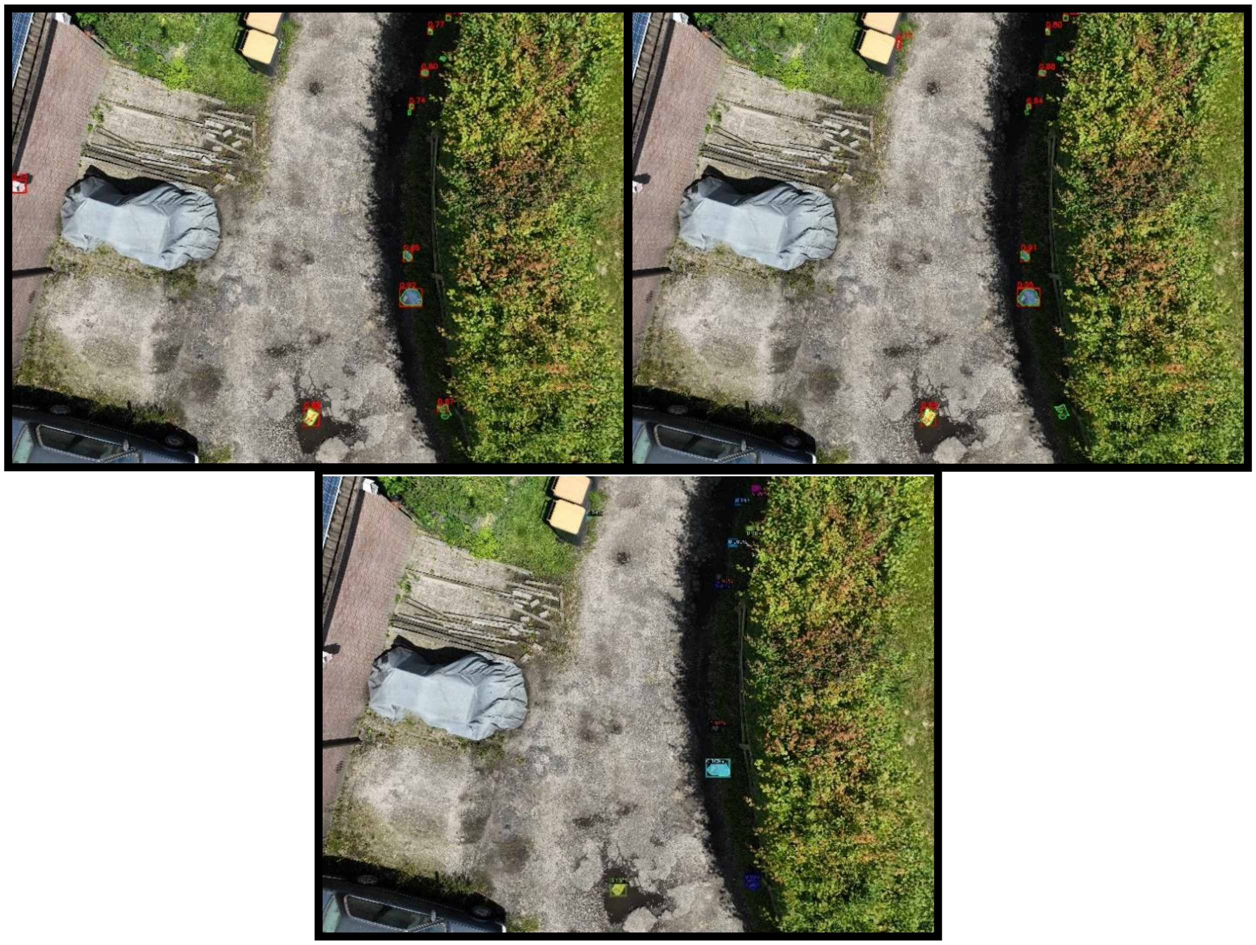
The Detectron2 model [Model6], the confidence scores are even higher than [Model5], but as seen in its metrics, it does have more issues with detecting smaller objects, which can be seen with the few objects above the red crate further in the background of the image.

Example 2 (DJI_20240617143426_0004_V.JPG)



This next example follows a similar pattern as with the previous image. The new RT-DETR model [Model5] scores better confidence scores compared to the old version [Model4], while also successfully detecting the hidden bottle near the bushes and a small cup in the bottom-right. The Detectron2 model [Model6] again has better confidence scores compared to the other two but is missing the same objects as the old [Model4].

Example 3 (DJI_20240617143430_0005_V.JPG)



In this example, the old [Model4] falsely marked the camera on the left as rubbish, which both the newer models don't. Both RT-DETR models do have trouble detecting the small object near the bushes at the top and the bag at the bottom, which the Detectron2 model got both. The newer models [Model5 & Model6] did manage to get one missed object near the garbage bins near the top of the image, which the old [Model4] missed.

The confidence scores again follow the same patterns as the previous examples.