Final Report for the Community-Based Project (Project Based Learning), EE299.

Name of the project

Group 6:

Louis Dwyer 21335163

Eoin Sheerin 20465956

Robert Modestov 20338303

Vladislava Batarevica 20739289

Project Facilitator: Mr. Andrew Meehan

May 2024

Department of Electronic Engineering

Maynooth University

Abstract

This document describes the problem of Irish canals' pollution and one of the possible solutions how to deal with this problem using Artificial Intelligence (AI). It details the principles of AI and neural networks, the code and necessary actions for the AI to learn and identify objects in a river scenario using a computer vision model YOLOv8. The objects that we sought out for the AI to identify were plastic bottles, chip wrappers and metal cans. There were three steps to our project, collection of the data and sorting it and then implementing it into a camera. Many of the images we have collected are open sourced for AI training, and 100 of the images of river trash were taken by group members. The AI code and training were done inside Google Collab with the resulting 84% accuracy of trash detection. The trained AI was implemented in a Luxonis Oak-D lite computer-vision camera.

Keywords: ["machine-learning", "deep learning", "Roboflow", "computer-vision", "Artificial Intelligence", "AI", "YOLOv8", "Ultralytics"]

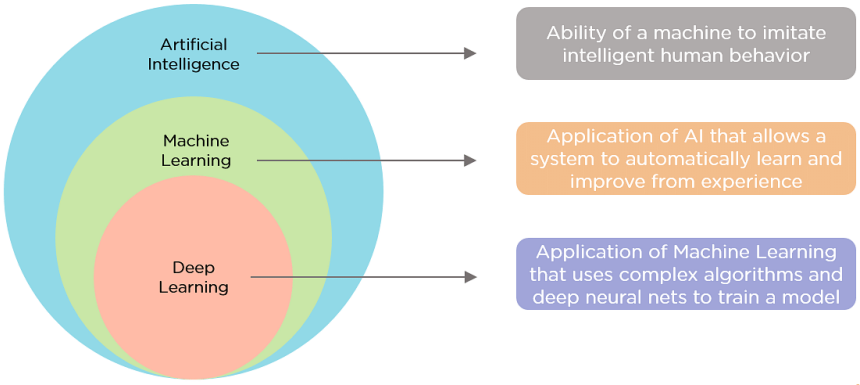
Introduction

Ireland has not only a long coastline and huge marine territory but also lots of rivers and canals. The canals are man-made waterways, the longest two canals in Ireland are Royal Canal (146 km) and Grand

Canal (144) km [1]. Both waterways run through towns and are quite littered in these areas. The most common types of litter in Irish waterways are food-related waste – plastic bottles, wrappers, cups, and cigarette butts [2]. The main problem of plastic waste is that it cannot be decomposed by itself. It is dangerous for flora and fauna, for example, there is a risk of plastic ingestion and suffocation by birds. Another problem is plastic breaks down into microplastic particles which are harmful to animals and even humans.

We were interested in this topic because the Royal Canal runs through Maynooth, and we can see how it gets littered on daily basis. The final topic of the project was chosen with help of engineering and mostly non-engineering community via survey (see appendix). The project is done in support of our local communities.

To solve the problem, we decided to use a computer vision tool - Yolov8 to train an AI how to detect the most common types of litter, mainly focusing on the plastic bottles, in our local canal. The goal was to reach 80% accuracy of plastic bottles detection on the water surface. This is done via collection images of trash in rivers and modifying it in Roboflow. The image dataset that has been collected is turned into computer code that the Yolov8 format can read and process. The trained AI was implemented



in computer-vision camera to detect litter in real-time.

Technical and Related Background

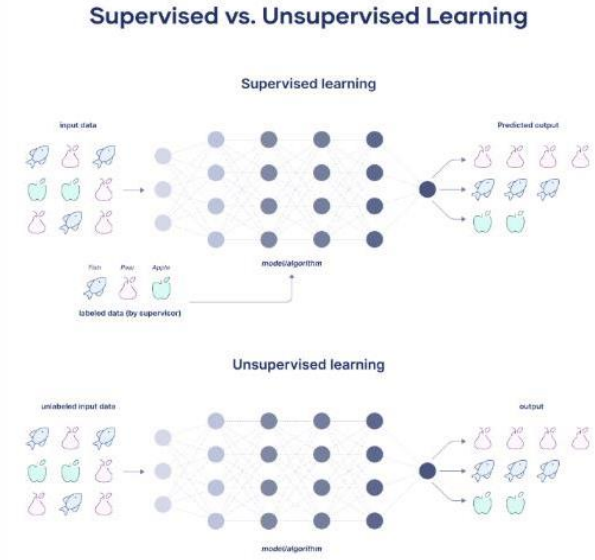
Artificial Intelligence

Artificial Intelligence (AI) is the use of computer systems to simulate human intelligence [3][4]. AI utilises multiple techniques to enable computers to perform tasks that previously required human input. Some examples of these techniques are machine learning and deep learning. AI systems needs to be trained to react how you would like them to under set situations. To do this AI models are given large

datasets. AI then analyses these datasets and recognises patterns and forms, predictions from the patterns recognised can be made about future states. A set of rules, which are called algorithms must be used to train the AI to ensure that the predictions are accurate. We train the AI using machine learning, and in the latter stages of machine learning, AI has the potential to adapt the existing algorithms to better fit the task at hand.

Machine learning

Machine learning is a subsection of AI, seen from the graph above which is made up of algorithms that, much like a human learn and adapt as they are exposed to more data over time, for example, we used **X** number of images of plastic bottles and it yielded a **Y** percentage of error. Machine learning involves getting the computer to recognise patterns in the datasets given and allowing the AI to improve at its respective task as time goes on. As a result, AI can perform tasks without human input as they are



trained on more and more data.

The two types of machine learning are Supervised and Unsupervised Machine Learning,

Unsupervised algorithms look for patterns in the dataset its given. It bases its output on the data it is given. It bases its output on the input data alone and as a result requires fewer manual data preparation, however, it is also difficult to measure accuracy or effectiveness due to lack of predefined answers during training [6]. Supervised learning bases the response of the system on both the input data (like unsupervised) and output data. This means that in supervised learning, we can have an exact idea about the classes of objects, however the training requires lots of computation times. The difference of these two types is shown in Figure 2.

When choosing a machine learning algorithm, it's important to consider what you want the system to do. Supervised learning algorithms come into play in systems where the computer needs to make

predictions. Therefore, unsupervised learning algorithms are ideal for detecting objects because the computer needs to interpret the image (using computer vision) and predict what will appear in the image. Unsupervised learning algorithms on the other hand, are much better at going through large amounts of data and finding suitable representations. Unsupervised learning algorithms are often used to divide huge amounts of data into smaller subsets of data**.**

Deep learning

Deep learning is a subset of machine learning that uses multi-layered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain [6]. Deep learning consists of naturally improving computer algorithms, which decreases the need for human interference. The "deep" in deep learning comes from the multiple layers found within the neural network. In deep learning the use of multiple layers to progressively extract high-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify different concepts, for example, if we wanted to recognise a human, we could use the higher layers to detect a face. Models are trained by using a large set of labelled data, and each individual level of the architecture takes the data and transforms it into more abstract concepts, for instance, in an image recognition system, where the input is purely raw pixels, the first level may encode the edges, the second layer may encode eyes and a nose, the third could encode a face and so on until the system fully recognises the image input into it.

An example of deep learning is the use of CNN's (Convolutional Neural Networks), which can be used for image recognition applications. One of the most popular programs is You Only Look Once (YOLO), which uses CNNs to detect objects from visual inputs in real-time.

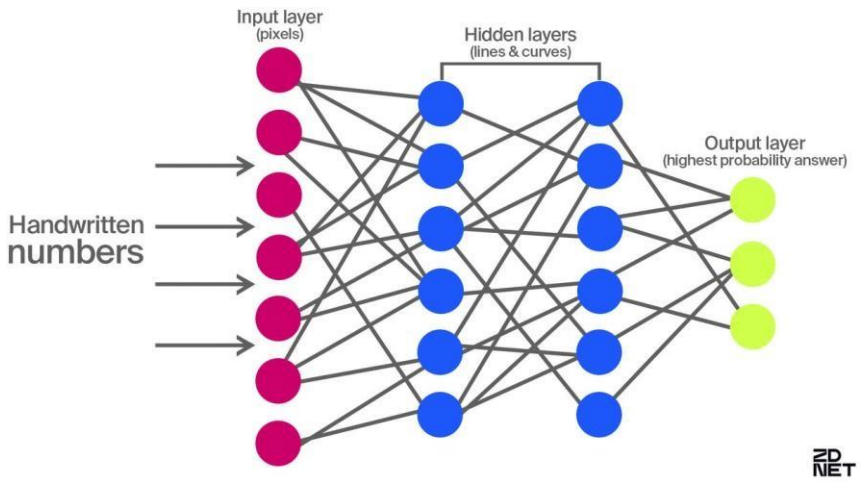


Figure 1 Deep learning algorithm [8]

Computer vision

Computer vision is a field of artificial intelligence that uses machine learning and neural networks to teach computers and systems to derive information from visual inputs and to make recommendations or take actions when they see defects or issues [9]. The aim of computer vision is to replicate what a human would typically see and identify. For instance, if a person views an image of a boat, they can identify what type of boat it is, barge, sailboat, kayak, fishing boat, etc. A human is also able to identify the colour of the boat. Computer vision aims to replicate this using machine and deep learning algorithms to dissect the image, it allows the system to provide information on the image that was input into it.

Artificial Neural Networks

Artificial Neural Networks (ANN) is a type of model inspired by the structure and function of biological neural networks in human brains.

An ANN consists of connected nodes called artificial neurons, which are based on the neurons in a brain. Each artificial neuron receives signals from connected neurons, then processes them and sends a signal to other connected neurons. The "signal" is a real number, and the output of each neuron is computed by the *activation function* which is simply of the sum of its inputs. The strength of the signal at each connection is determined by a weight, which adjusts during the learning process.

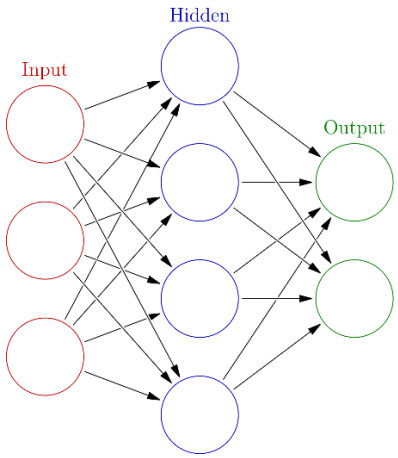


Figure 2 Example of neural network structure [10]

Usually, these neurons are divided into layers. Different layers perform different functions. Signals pass through many layers to reach the output layer, and we can call the neural network a *deep network* if it has more than 2 layers. Neural networks are used for many tasks including language

processing and computer vision. They excel at tasks requiring pattern recognition which make them great for an object detection system.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special type of deep learning algorithm primarily designed for tasks that require object detection, such as image classification and recognition. CNNs are used in a variety of practical scenarios, including self-driving cars and security camera systems. CNNs contain at least 3 layers, the input layer, the hidden layers and the output layer, see Figure 4. The hidden layers typically contain the Convolutional layer, the pooling layer and the Fully Connected Layer. [4]

The convolutional layer is the main building block of a CNN, in the convolutional layer small grids (kernels) are used to slide over the image that as input into the CNN. Each small grid is like a mini magnifying glass that looks for specific patterns in the photo, like lines, curves, or shapes. As it moves across the photo, it creates a new grid that highlights where it found these patterns. Using these we can use different kernels for different things, for example, we could use one kernel for finding curves and another for finding straight lines. By using many different kernels, we can get an idea of the patterns that make up an image.

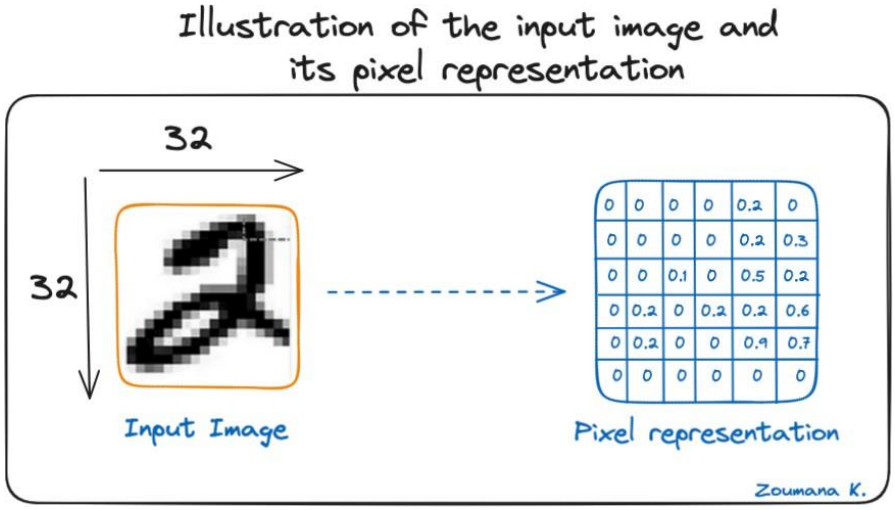


Figure 3 Illustration of the input image and its pixel representation [11]

In real life, the weights of the kernels are determined during the training process of the neural network.

Using these two matrices, we can perform the convolution operation by applying the dot product, and work as follows:

1. Apply the kernel matrix from the top-left corner to the right.

2. Perform element-wise multiplication.
3. Sum the values of the products.
4. The resulting value corresponds to the first value (top-left corner) in the convoluted matrix.
5. Move the kernel down with respect to the size of the sliding window.
6. Repeat steps 1 to 5 until the image matrix is fully covered.

After each convolution operation, a Rectified Linear Unit (ReLU) activation function is applied. This feature helps the network learn nonlinear relationships between features in images, making the network more robust in identifying different patterns.

The purpose of the pooling layer is to obtain the most important features from the convolution matrix. This is done by applying several aggregation operations that reduce the dimensionality of the feature map (convolution matrix) and reduce the memory used to train the network. Pooling is also associated with reducing overfitting. The most commonly used aggregate functions are:

- Max pooling, which is the maximum value of the feature map.
- Sum pooling corresponds to the sum of all the values of the feature map.
- Average pooling is the average of all the values.

The last hidden layer in a CNN is always a fully connected (FC) layer. The FC layer performs the task of classification based on features extracted from the hidden layers before the convolution and pooling layers. A list of classes that can be classified by CNN is output, and each class corresponds to a confidence level.

YOLO algorithm

When designing our system originally, we needed to take into consideration how to implement the detection system that would be the base of our project. We initially discussed the work involved in designing our own system but realised that this task would be much too time consuming. Looking further into already designed and implemented systems we found the You Only Look Once(YOLO) system is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015 [12]. It is fast and efficient, thus making it an excellent choice for real-time object detection tasks. It functions using predictions of bounding boxes and class probabilities at once. This was a different approach when compared to previous object detection algorithms and provided improvements by a large margin.

Functionality

The idea behind the YOLO algorithm is to divide the inputted image into a group of 'boxes', and for each box, it tries to predict the probability of the presence of an object in the box and the bounding coordinates of the object within the box.

The processes undertaken by YOLO are as follows:

1. The input image is passed through a CNN to extract features from the image.
2. The features are then passed through a series of fully connected layers which predict bounding box coordinates.
3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.
4. The output of the network is a set of bounding boxes and class probabilities for each cell.
5. The bounding box is then filtered using a post-processing algorithm called non-maximal suppression to remove duplicate frames and select the frame with the highest probability.
6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

One of the key advantages of YOLO is that the image processing happens in a single pass which allows it greater speed and efficiency than some of its counterparts which rely on two passes for image processing, like R-CNN.

In previous versions, there was a thing called anchor boxes introduced, the aim of anchor boxes is to reduce false positive flags for object within each 'box'.

For example, in YOLO v3 an image might be divided into a 13x13, 26x26, or 52x52 grid. Each cell in the grid is responsible for detecting objects whose centre falls within the cell. Each grid would be assigned multiple anchor boxes (usually between 3 and 9) each with a different shape and size.

For each anchor box, the network predicts:

1. The offset of the box centre relative to the grid cell.
2. The width and height adjustments relative to the anchor box dimensions.
3. The confidence score, which indicates how likely it is that the box contains an object.
4. The class probabilities, which indicate the type of object within the box.

Centre Coordinates (t_x, t_y): YOLO predicts offsets for the centre coordinates relative to the grid cell. The actual centre coordinates (b_x, b_y) are computed using a sigmoid function to constrain them within the cell:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

where c_x, c_y are the top-left coordinates of the grid cell. (1)

$$b_w = p_w e^{tw}$$

$$b_h = p_h e^{th}$$

- **Width and Height (t_w, t_h)**: YOLO predicts log-space transformations for the width and height. The actual width and height (b_w, b_h) are calculated as:

where p_w,p_h are the predefined anchor box dimensions. (2)

The Confidence Score is predicted directly using a sigmoid function, representing the probability that an object is present in the box.

After predicting the bounding boxes, YOLO applies Non-Maximum Suppression to remove duplicate boxes. Which includes sorting the boxes by their confidence scores and repeatedly selecting the box with the highest score and removing boxes that have a high Intersection over Union (IoU) with the selected box.

In YOLOv8, however, it does not rely on predefined anchor boxes to generate object proposals. Instead, YOLOv8 directly predicts the bounding boxes and class probabilities for each object in the input image. This approach reduces the complexity of the model and allows for more flexibility in detecting objects of various sizes and aspect ratios.

Intersection over Union

As stated previously Intersection over Union (IOU) is a number that quantifies the degree of overlap between two bounding boxes between the predicted box and the actual ground truth bounding box. In Figure 6 we can see that our object detector has detected the presence of a stop sign in an image. The *predicted* bounding box is drawn in *red* while the *ground-truth* (i.e., hand labelled) bounding box is drawn in green [13]. More accurately IOU is the ratio of the overlap area to the combined area of prediction and ground truth. When the predicted box's area is larger, the denominator in the Intersection over Union (IoU) calculation will be greater. As a result, the IoU value will be lower. IoU values are expressed on a scale of 0 to 1, where 0 indicates no overlap between the predicted and ground truth boxes, while a value of 1 denotes perfect overlap. To surmise the closer the value is to one the more accurate the system.

IOU is defined as: $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$ (3)

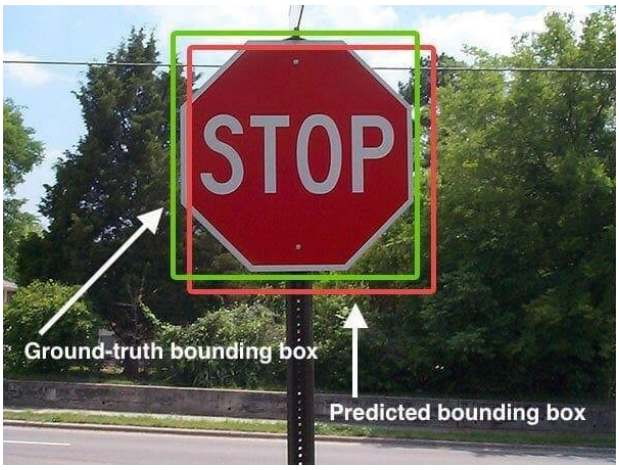


Figure 4 Hand-labelled box (green) and predicted box (red) [13]

Non-max suppression

Non-max suppression (NMS) is the final step of these object detection algorithms and is used to select the most appropriate bounding box for the object. The primary purpose of NMS is to remove redundant and overlapping bounding boxes, ensuring that each detected object is represented by a single, high-confidence bounding box.

The following steps are involved in NMS:

1. First, all the predicted bounding boxes are sorted in descending order based on their confidence scores. The confidence score represents the likelihood that the box contains an object.
2. The box with the highest confidence score is selected as the first box. This box is considered a strong candidate for containing an object.
3. 3: For the selected box, NMS calculates the Intersection over Union (IoU) with all other remaining boxes
4. 4: Any box with an IoU greater than a predefined threshold (e.g., 0.5) with the selected box is suppressed, meaning it is discarded from further consideration. This step helps in removing boxes that are likely to represent the same object.
5. 5: The process is repeated with the remaining boxes: selecting the next highest confidence box, calculating IoU with the remaining boxes, and suppressing those with high overlap. This continues until all boxes are either selected or suppressed.

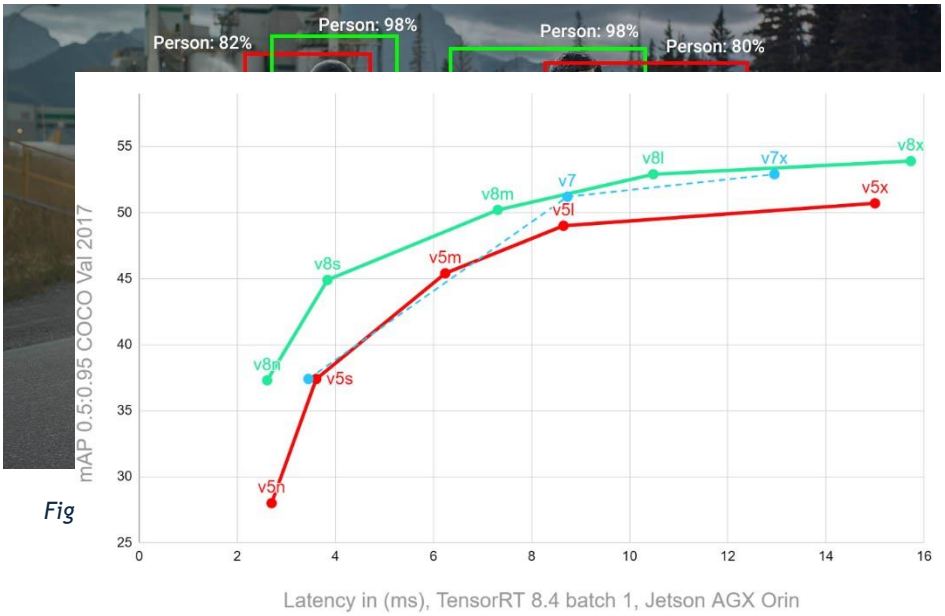
We must keep in mind that selecting the threshold value is very important, an example of a 'good' value is 0.5, which means boxes with more than 50% overlap are considered duplicates and are suppressed. However, this value can be adjusted based on the specific requirements of the system.

In Figure 7, if NMS functions correctly, it will select the green bounding box as it has the highest probability and remove red boxes (because they have a high overlap with the green box).

Selecting correct YOLO version

YOLOv8, being the latest version in the YOLO series, has several advantages over its predecessors.

One of the main reasons we chose YOLOv8 is its faster detection speed. YOLOv8 can process images in real-time, enabling quick and reliable object recognition. YOLOv8 also excels in detecting small



Fig

Figure 6 YOLOv5 vs v7 vs v8 [15]

objects, which is ideal for identifying trash which comes in all different shapes and sizes. Another key feature of YOLOv8 is its anchor-free architecture. By eliminating the need for predefined anchor boxes, YOLOv8 adapts to different object sizes and proportions more effectively, resulting in improved detection accuracy.

As we can see from the figure above, YOLOv8 is the fastest in terms of speed in comparison to its predecessors, coupled with the fact that it has anchr free architecture which allows it to not have predefined hyper-parameters associated with the anchor boxes, and so do not need to search for the optimal values for these hyper-parameters. Which allows greater speed. Also, there is no need to set different predefined anchor boxes for different datasets, so it has better generalization ability.

PyTorch Framework

In terms of programming, a framework is a reusable, semi-complete application that provides a foundation for developing software applications. A framework provides the support and a basic structure which is used to build upon to develop software and create projects and system. It is designed to help

developers by offering pre-written code, guidelines, and best practices, thus speeding up the development process and ensuring consistency across different projects.

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab (FAIR). It is widely used for applications such as computer vision. It is based on the Python programming language and the Torch framework and is mainly used for applications using CPUs and GPUs. The PyTorch framework enables fast, flexible, and efficient development through a user-friendly front-end and a vast range of tools and libraries.

The two main features of PyTorch are [16]:
• Tensor computation with GPU acceleration support
• Automatic differentiation for creating and training deep neural network

How Pytorch works is there are tensors, which are n-dimensional array that used for representing and storing data, similar to numPy arrays. Tensors represent the basic data structure for storing and manipulating data in PyTorch. These tensors can represent a number, vector, matrix, or multi-dimensional array. PyTorch provides a rich set of tensor operations that allow users to perform mathematical computations efficiently. These operations are optimized for both CPU and GPU execution.

PyTorch's 'autograd' package provides automatic differentiation functionality, which allows users to compute gradients of tensor operations with respect to input tensors. This feature is essential for training neural networks using gradient-based optimization algorithms like stochastic gradient descent (SGD). By simply calling the backward () method on a scalar tensor representing the loss, PyTorch computes gradients for all tensors involved in the computation graph, which enables efficient backpropagation.

Backpropagation, short for "backward propagation of errors", is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights [17].

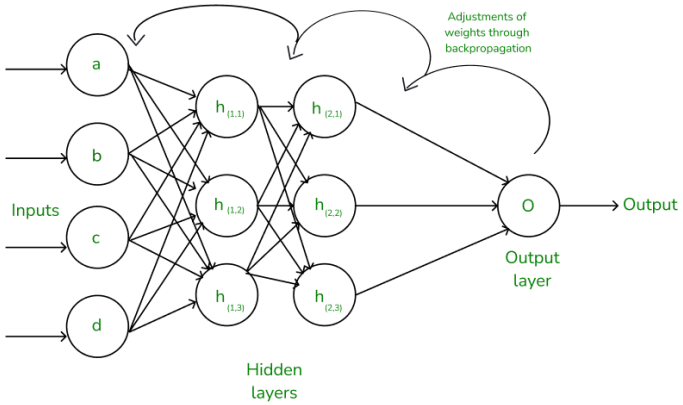


Figure 7 Backpropogation [18]

Object Tracking

Object tracking is a fundamental task in computer vision that involves finding and tracking objects of interest across a set of images in a video or image stream. Unlike object detection, which identifies objects within individual images, object tracking involves assigning a unique identification to each detected object and tracking the object through each frame of the video once the initial position of the target object is defined. This involves several important steps, such as object initialization. The tracker identifies the object in the first image and then performs motion estimation and state estimation to predict the object's position in subsequent images. Various tracking algorithms are used to track objects based on their movement, appearance, or a combination of both, such as Kalman filters and deep learning-based approaches such as SORT (Simple Online and Real-Time Tracking) . Perhaps the most widely known use case of Object tracking is self-driving cars. In this case, object tracking has several purposes such as obstacle detection, pedestrian detection, trajectory estimation, collision avoidance, vehicle speed estimation, traffic monitoring, and route estimation [19].

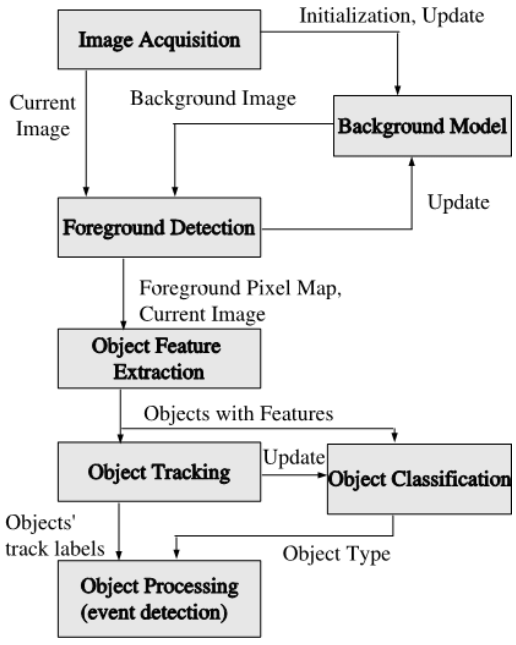


Figure 8 Object detection block diagram

A high-level overview of the concept of object tracking with YOLO implementation can be seen in Figure 10.

The steps involving in object tracking are as follows. First frames of a video are passed through an object detection model. Then The object detection model detects and classifies each object of interest in the frame by producing a bounding box and confidence score for each object. After that the object tracking algorithm assigns each object that has been identified with a unique identification label. Lastly the object

tracking algorithm keeps track of the detected objects as each frame is passed through the object detection model while storing the relevant information on the tracked object and maintaining the assigned IDs.

Object Tracking algorithms

Object tracking technology can be classified as single object tracking (SOT) or multiple object tracking (MOT). Signal object tracking is the tracking of a single class of specific target objects within each frame of a video. In SOT, the bounding box of the target object of interest is defined in the first frame, and the object tracking algorithm finds and tracks the same object in the remaining frames. By comparison, multiple object tracking is a more complex type of tracking because it requires tracking multiple objects at the same time. In MOT, each object of interest is detected every frame, and the object tracking algorithm tracks the movement of each object over consecutive frames until the object is no longer visible in the frame.

Kalman Filters

Kalman filters are ideal for systems which are continuously changing. They have the advantage that they are light on memory, and they are very fast, making them well suited for real time problems and embedded systems. You can use a Kalman filter in any place where you have uncertain information about some dynamic system, and you can make an educated guess about what the system is going to do next.

Say we have a system based on an object falling for simplicity. This system, for example, has a model of:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

$$\mathbf{P}_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

where the best estimate $\hat{\mathbf{x}}_k$ (the mean) and its covariance matrix Pk (4)

Next, we need some way to look at the current state (at time k-1) and predict the next state at time k.

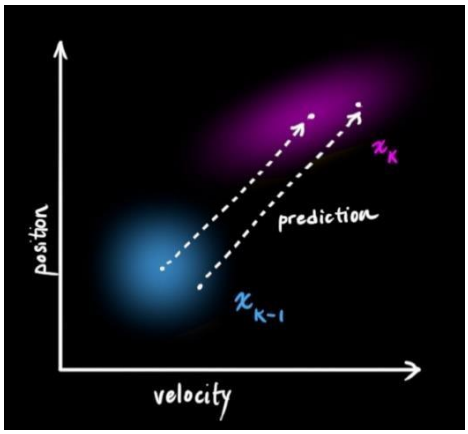


Figure 10 Next state prediction [21]

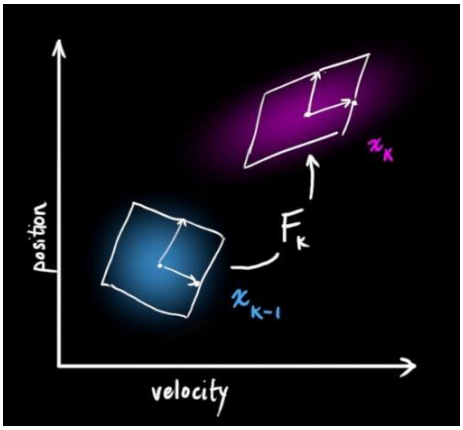


Figure 9 Predicted location [21]

We can represent this prediction step with a matrix, $\mathbf{F_k}$, (Figure 12).

$$\begin{matrix} p_k \\ v_k \end{matrix} = \begin{matrix} p_{k-1} + \Delta t \\ = \end{matrix} \begin{matrix} v_{k-1} \\ v_{k-1} \end{matrix}$$

This takes every point in our original estimate and moves it to a new predicted location, which is where the system would move if that original estimate was the right one.

We use this formula.

(5)

Which allows us to get the prediction matrix which in turn allows us to get the next state:

$$\begin{matrix} \hat{\mathbf{x}}_k \\ Cov(x) \\ Cov(\mathbf{A}x) \end{matrix} = \begin{matrix} \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \Sigma = \mathbf{F_k} \\ = \mathbf{A}\Sigma\mathbf{A}^T \end{matrix} \qquad (6)$$

However, the last thing we need to do is update the Covariance matrix ∑:

Which we can get using

$$\begin{matrix} \hat{\mathbf{x}}_k \\ \mathbf{P}_k \end{matrix} = \begin{matrix} \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \\ = \mathbf{F_k} \mathbf{P}_{k-1} \mathbf{F}_k^T \end{matrix} \qquad (7)$$

Therefore, we are left with

(8)

[21]

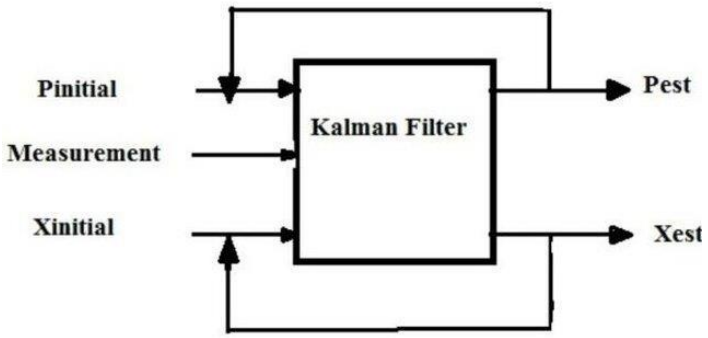


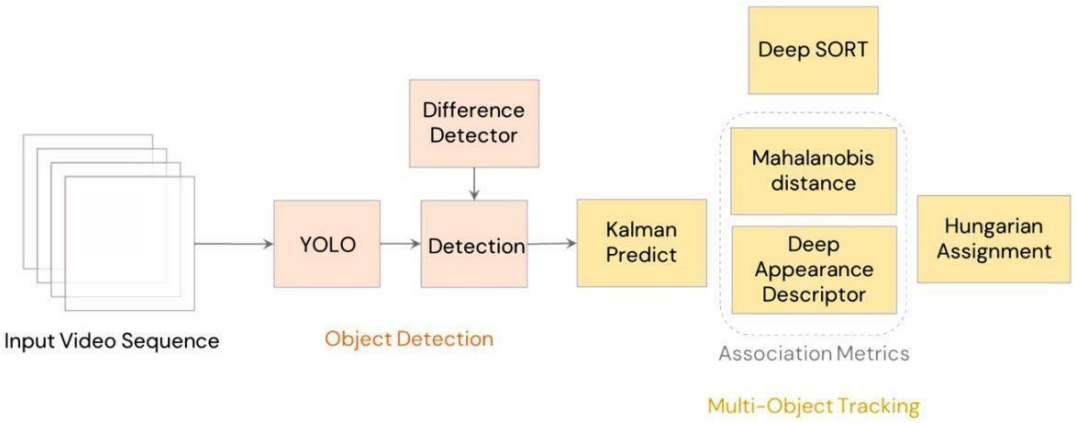
Figure 11 Kalman filter simple diagram [22]

SORT Algorithm

The SORT algorithm is the basis of the DeepSORT (Deep Learning for Multiple Object Tracking) system. SORT is a simple yet effective algorithm for data mapping and tracking initiation in multiple object-tracking scenarios. In SORT we use a detection-based approach to map objects in consecutive frames, then use a Kalman filter to predict the object's future position and correct the mapping based on actual measurements.

Deep SORT Algorithm

On the other hand, Deep SORT extends SORT by incorporating deep learning-based appearance features for robust object identity tracking. It utilizes a deep Convolutional Neural Network to extract appearance embeddings from object detections. This allows the algorithm to handle situations where objects may temporarily disappear or become occluded in the video stream especially in scenarios with occlusions and cluttered environments. Deep SORT also incorporates ID assignment to track individual objects across multiple frames, Similar to SORT, Deep SORT uses a Kalman filter for state prediction. The state of an object typically includes its position, velocity, and acceleration. The Kalman filter



predicts the state of each object in the current frame based on its last known state, accounting for the object's motion dynamics [23].

Deep-SORT is one of the most popular objects tracking algorithms which we feel is due to the use of appearance descriptors, which allow Deep SORT algorithms to re-identify objects after they may have been occluded. They also allow Deep SORT algorithms to distinguish between similar-looking objects.

Off-Board Processing vs On-Board Processing

In designing the system, various considerations were weighed when deciding between on-board and off-board processing. This decision involved assessing the processing complexity, resource availability, network bandwidth, and latency constraints for handling the collected data. The decision was to determine whether processing should occur externally or be centralized within the device, eliminating the need for data transmission from the source. Off-board processing entails streaming video to an external processing unit, enabling the execution of more computationally intensive tasks due to better processing capability. However, this method may introduce notable latency and bandwidth limitations, particularly if the camera is situated in a far-off location.

Off Board Processing

In off board processing we would need to design a system architecture including, data collection devices which could be cameras equipped with image-capturing capabilities to monitor the canal surfaces. We would also need reliable data transmission channels, for example, 4G or Wi-Fi to send collected data to the processing server which we would also need to be able to run YOLOv8. Databases or cloud storage to store both the raw and processed data. Lastly, we would require a User Interface which may include a mobile app for visualisation and real-time alerts.

On Board Processing

On-Board Processing requires a small processing unit and capture device to be built into a single module. This has the advantage of a low latency time for video streaming. The captures are then processed, and detection results are generated using a Python script. The graphical intensity of such a process requires a GPU with quite a bit of memory, for example, a 1080 ti which has 8gb of GDDR6 memory, this enables the handling of high workloads while also remaining efficient. The processing unit used has to be small enough to be mounted with the camera without compromising functionality. Smaller

modules have less memory available for processing, which can affect the speed of data processing, depending on the overall system size.

Choosing between the two.

To summarise On-board processing allows for real-time analysis and immediate feedback, reducing latency and reliance on data transmission networks. However, it is often limited by the processing power and energy capacity of the device. In comparison, off-board processing entails transmitting the collected data to a more powerful external server for analysis, which can handle more complex and larger-scale computations but introduces latency and dependency on reliable data transmission. Currently, we have opted for on-board processing to achieve immediate results and minimize dependence on network stability. However, our ultimate goal is to transition to fully off-board processing to utilise its additional computational power, allowing for more sophisticated analysis and continuous improvement of our trash detection system.

Work Completed

We are using Yolov8 by Ultralytics. The code is run in google colab. First, we must install the dependencies for our project, everything is in one bundle. Then we can install Ultralytics. A certain version is required for roboflow. Even though this version 8.0.196 it is still three months out of date.

!

```
Requirement satisfied: ultralytics==8.0.196 in /usr/local/lib/python3.10/dist-packages (8.0.196)
Requirement satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (3.7.1)
Requirement satisfied: numpy>=1.22.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (1.25.2)
Requirement satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (4.8.0.76)
Requirement satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (9.4.0)
Requirement satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (6.0.1)
Requirement satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (2.31.0)
Requirement satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (1.11.4)
Requirement satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (2.2.1+cu121)
Requirement satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (0.17.1+cu121)
Requirement satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (4.66.2)
Requirement satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (2.0.3)
Requirement satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (0.13.1)
Requirement satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (5.9.5)
Requirement satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (9.0.0)
Requirement satisfied: thop>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.196) (0.1.1.post2209072238)
Requirement satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (1.2.1)
Requirement satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (0.10.0)
Requirement satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (4.51.0)
Requirement satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (1.4.5)
Requirement satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (24.0)
Requirement satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (3.1.2)
Requirement satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics==8.0.196) (2.8.2)
```

pip install ultralytics==8.0.196

```
import torch
from ultralytics import YOLO
import os
import torch
from IPython.display import display , Image
from IPython import display
display.clear_output()
!yolo mode=checks
```

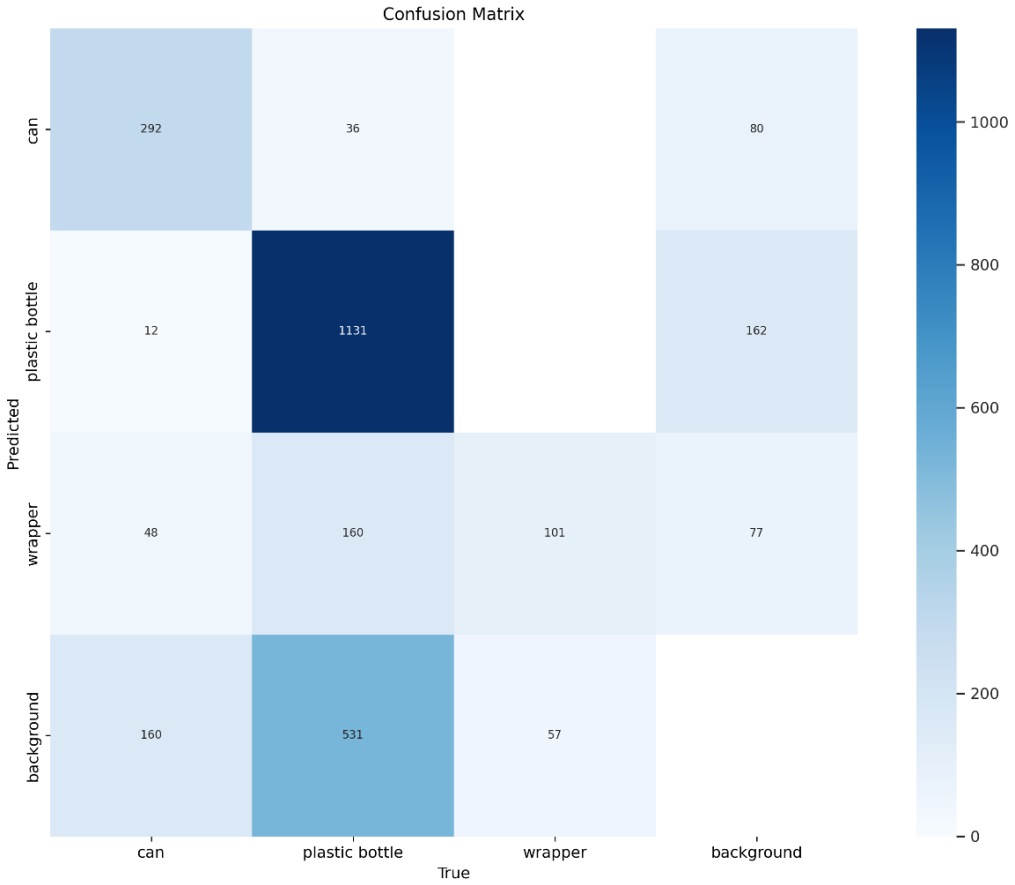
```
File "/usr/local/bin/yolo", line 8, in <module>
  sys.exit(entrypoint())
 File "/usr/local/lib/python3.10/dist-packages/ultralytics/cfg/__init__.py", line 393, in entrypoint
  raise ValueError(f"Invalid 'mode={mode}'. Valid modes are {MODES}.\n{CLI_HELP_MSG}")
ValueError: Invalid 'mode=<module 'ultralytics.utils.checks' from '/usr/local/lib/python3.10/dist-packages/ultralytics/utils/checks.py'>'. Valid modes are ('train', 'val', 'predict', 'export', 'track', 'benchmark').


  Arguments received: ['yolo', 'mode=checks']. Ultralytics 'yolo' commands use the following syntax:


    yolo TASK MODE ARGS


    Where  TASK (optional) is one of ('detect', 'segment', 'classify', 'pose')
          MODE (required) is one of ('train', 'val', 'predict', 'export', 'track', 'benchmark')
          ARGS (optional) are any number of custom 'arg=value' pairs like 'imgsz=320' that override defaults.
            See all ARGS at https://docs.ultralytics.com/usage/cfg or with 'yolo cfg'


  1. Train a detection model for 10 epochs with an initial learning_rate of 0.01
    yolo train data=coco128.yaml model=yolov8n.pt epochs=10 lr0=0.01


  2. Predict a YouTube video using a pretrained segmentation model at image size 320:
```

```
!pip install roboflow
```

```
from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("uni-project-hbxtp").project("irish-canals")
version = project.version(2)
dataset = version.download("yolov8")
```

```
Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-packages (1.1.28)
Requirement already satisfied: certifi==2023.7.22 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2023.7.22)
Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.0.0)
Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.10.0)
Requirement already satisfied: idna==2.10 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.10)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.25.2)
Requirement already satisfied: opencv-python-headless==4.8.0.74 in /usr/local/lib/python3.10/dist-packages (from
roboflow) (4.8.0.74)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (9.4.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.0.7)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.2)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.1)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: python-magic in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.4.27)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow)
(1.2.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow)
(4.51.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow)
(24.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow)
(3.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from
requests->roboflow) (3.3.2)
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Irish-Canals-2 to yolov8:: 100%|■■■■■■■■■■| 374709/374709 [00:22<00:00,
16587.32it/s]Extracting Dataset Version Zip to Irish-Canals-2 in yolov8:: 100%|■■■■■■■■■■| 16082/16082
[00:03<00:00, 5311.15it/s]
```

```
!yolo task=detect mode=train model=yolov8m.pt
data={dataset.location}/data.yaml epochs=20 imgsz=640
```

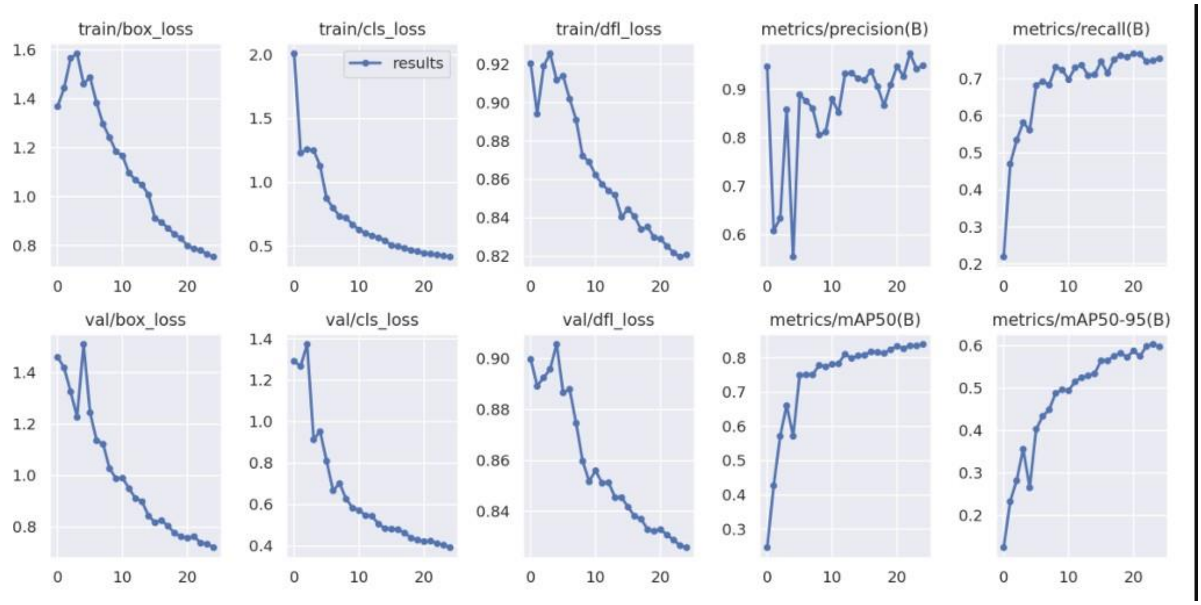


As seen the Ai had 6:2:2 ratio it saw three bottles for each wrapper and can. Since we wanted to have a good object detection. Plastic bottles where mistaken for 12 cans and the AI missed 162 bottles out of 2000 that it was tested on.

[23]

Luxonis Camera data

The best performing AI was used on our camera, the data below is for that.



Focus on the metrics/mAP50(B), after running the machine-learning for 20 epochs the overall average validation of new objects was found to be above 0.84 or in a different form 84%. That means the AI was accurate more than 80% of the time.

The model setting where on a medium bias setting, the task was detection, and it was validation. The AI was looking at new images that were given to it. There are some areas we could improve on, such as better human image identification. Some images have been misidentified.

To utilise the camera with YOLOv8 the following code was required:

```
import depthai as dai
import cv2
from yolov8.yolov8_detect import YOLOv8

# Initialize Oak-D Lite camera
pipeline = dai.Pipeline()

# Define the camera configuration
cam_rgb = pipeline.createColorCamera()
cam_rgb.setBoardSocket(dai.CameraBoardSocket.RGB)
cam_rgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P
)

# Set up neural network (YOLOv8)
yolo_nn = pipeline.createYoloDetectionNetwork()
yolo_nn.setConfidenceThreshold(0.5)
yolo_nn.setNumClasses(80)
yolo_nn.setCoordinateSize(4)
yolo_nn.setAnchors([10, 14, 23, 27, 37, 58, 81, 82, 135, 169, 344, 319])
yolo_nn.setAnchorMasks([[6, 7, 8], [3, 4, 5], [0, 1, 2]])
yolo_nn.setIouThreshold(0.5)

# Define input/output streams
cam_rgb.preview.link(yolo_nn.input)
yolo_nn.passthrough.link(xout_rgb.input)

# Create the DepthAI device
device = dai.Device(pipeline)
video_out = device.getOutputQueue(name="rgb", maxSize=1, blocking=False)

# Initialize YOLOv8 model
Yolov8 = YOLOv8(weights='yolov8s.pt', img_size=640)

# Main loop
while True:
    in_rgb = video_out.get()

    # Get the frame from the camera
    frame_rgb = in_rgb.getCvFrame()

    # Perform YOLOv8 object detection
    detections = yolov8.detect(frame_rgb)

    # Draw bounding boxes on the frame
    for detection in detections:
        x1, y1, x2, y2, conf, cls = detection
        cv2.rectangle(frame_rgb, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(frame_rgb, f'{cls}: {conf:.2f}', (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # Display the frame
    cv2.imshow('YOLOv8 Object Detection', frame_rgb)
    if cv2.waitKey(1) == ord('q'):
        break

# Clean up
cv2.destroyAllWindows()
```

Future Work

Making our system off board compatible

We would need a compact, powerful mini-computer to make our system Off-Board compatible, something like a raspberry Pi would be ideal. We could use the Raspberry Pi 4 as it has a quad-core ARM Cortex-A72 CPU, up to 8 GB of RAM, and versatile connectivity options which makes it a flexible platform for various projects. By integrating the Intel NCS2, which uses the Movidius Myriad X VPU for accelerated AI inference, this setup improves the Raspberry Pi's capability to run complex machine learning models efficiently. Using the OpenVINO toolkit, developers can deploy and optimize AI models for real-time trash detection and GPS-based location tracking. This combination is especially suitable for edge computing scenarios, offering a balance of performance, scalability, and cost-effectiveness while maintaining the versatility and ease of use associated with the Raspberry Pi ecosystem.

OpenVINO (Open Visual Inference and Neural Network Optimization) is an open-source toolkit developed by Intel, designed to accelerate deep learning inference on a variety of Intel hardware platforms. OpenVINO enables faster inference times and lower power consumption, making it ideal for edge devices with limited resources with models from frameworks like PyTorch and google colab. **[9]**

Implementing Deep SORT

In order to track moving/floating trash between successive frames of the video, the DeepSORT object tracking algorithm could be used. While there are many object tracking algorithms, we found that the DeepSORT object tracking algorithm provided the functionality to track multiple objects in real-time applications and could be implemented using YOLOv8 as an object detector which suited the requirements of our project. The DeepSORT code is available from the Python Package Index (PyPI) repository under the package name deep-sort-realtime. This package is an adaptation of DeepSORT from the official GitHub repository of Simple Online and Realtime Tracking with a Deep Association Metric (Deep SORT).This DeepSORT tracker uses the object detections from the YOLOv8 model in order track these detected objects as they move across the frame of the video.

GPS System.

Our eventual aim was to implement a GPS system once we had our off-board processing set up. This would ensure that whenever trash was detected there could be a ping of the coordinates of where the system was when the trash was detected. We could do this by equipping a Raspberry Pi with a GPS module and configuring it to interface with the camera used for trash detection. The Raspberry Pi captures images and processes them using built-in preliminary algorithms to detect a piece of potential trash. Once a piece of trash is identified, the Raspberry Pi retrieves the current GPS coordinates from the GPS module. This information, along with the image data, is sent to an external processing server for further analysis by more complex CNN models. The server can then verify the detection, record the coordinates, and trigger any necessary alerts or actions. By incorporating GPS data, the system ensures precise location tracking of detected waste, enabling efficient and targeted cleaning operations and improving the overall effectiveness of environmental monitoring systems.

This would allow our system to be used in interesting ways. When we spoke to the Green campus committee, we soon found out that the MUCK society were quite interested in the cleaning of canals. This is to be expected as kayaking usually takes place in canals. The GPS system would allow us to place one of these systems on a kayak and have the kayak padel down the canal. This system would then mean that the trash is identified we would have a list of the spots on the canal with trash in them thus increasing the cleanup time tremendously and we can also use it for data analysis and see if there is common places trash is dumped often. If this is the case precautionary measures can be taken to reduce these potential hubs such as signs or nets that reduce the amount of littering.

Ethics

Since our project involves the using video data to detect trash in Irelands canals, we must consider the ethics of recording/streaming video footage and handling this data without permission from the persons that may be walking through the frame of the video. This situation falls under the scope of the General Data Protection Regulation (GDPR). The ethics of collecting or using personal data either through recording or streaming video without permission for the purposes of our project depend on

various factors. In regard to streaming video footage and not storing this data, it allows us to circumvent a large majority of the GDPR regulations.

What is the GDPR legislation?

The General Data Protection Regulation (GDPR) is a comprehensive data security and privacy law that was introduced by the European Union (EU) in 2018. The GDPR replaced the previous Data Protection Directive and is intended to unify and strengthen data protection for all individuals within the EU. GDPR aims to strengthen and unify data protection for individuals within the EU and the European Economic Area (EEA), as well as regulate the export of personal data outside these regions. The regulation introduces several key principles and requirements for organizations that handle personal data, including obtaining explicit consent for data processing, implementing measures to ensure data security, providing individuals with the right to access and control their data, and imposing strict penalties for non-compliance. GDPR applies to all organizations, regardless of their location, that process personal data of individuals residing in the EU or EEA, making it one of the most significant and far-reaching data protection regulations globally. There are several key definitions defined in Article 4 [27] of the GDPR that are relevant, and they are:

• Personal data - "means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name"
• Processing - "means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction"
• Controller - "means the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law" Under the GDPR legislation, which applies to all EU member states including Ireland, there are strict rules regarding the processing of personal data, including images.

Article 6 of the GDPR outlines the lawful bases for processing personal data. It specifies the conditions under which organizations are permitted to collect, use, or process personal data of individuals within the European Union (EU) and the European Economic Area (EEA). Article 6 lists six lawful bases for processing personal data, which includes the necessity of processing for the performance of a contract, compliance with legal obligations, protection of vital interests, consent of the data subject, performance of tasks carried out in the public interest, and legitimate interests

pursued by the data controller or a third party. Organizations must determine a lawful basis for each instance of data processing and ensure that the processing complies with GDPR principles, including transparency, purpose limitation, and data minimization. Additionally, organizations must document and be able to demonstrate the lawful basis for their data processing activities to regulatory authorities. It is important to understand these regulations when planning to record images of individuals. To comply with GDPR regulations, individuals must be provided with clear and concise information about the processing of their personal data, including the reasons for recording their images, how the images will be used, and who will have access to them. Additionally, there must be a lawful basis for processing personal data, including images. Obtaining explicit consent from the individual is the most common lawful basis for processing images, but other options such as legitimate interests or obtaining legal certifications may also be available. It is important to note that if recorded data is to be retained, it must be stored in a secure location and only accessible by qualified personnel. This ensures that the privacy and security of the recorded data is always maintained.

Furthermore, since we are dealing with Irelands Waterways so we must also contact waterways Ireland as they are the authoritative body managing these waterways. Securing permission from Waterways Ireland ensures that the project complies with all legal and safety regulations, protecting both ourselves and the environment. Additionally, they can provide valuable resources, information, and guidance that can enhance the quality and accuracy of the project. Furthermore, collaboration with this organization can open opportunities for educational support and potential partnerships.

[26f] **https://docs.openvino.ai/2024/home.html**

[23f] **https://www.ikomia.ai/blog/deep-sort-object-tracking-guide**

[27f] **https://gdpr-info.eu/art-4-gdpr/**