2023

INTERNSHIP REPORT

ASIAN INSTITUTE OF TECHNOLOGY
GEOINFORMATICS CENTER

SACHIN GIRI
www.sachingiri03.com.np

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the Asian Institute of Technology (AIT), especially the GeoInformatics Center, for providing me with the opportunity to undertake my internship and work on the project focused on plastic litter detection in water bodies using computer vision object detection algorithms.

I am immensely thankful to my supervisor, Sriram Reddy, for his exceptional guidance, valuable insights, and unwavering support throughout the course of my internship. His expertise in the field of GeoInformatics and computer vision has been instrumental in shaping this project and enhancing my understanding of the subject matter.

I am also grateful to the entire team at the GeoInformatics Center for their assistance, valuable discussions, and constructive feedback, which have significantly contributed to the development and success of this project. Their collective expertise and enthusiasm have been truly inspiring.

I am honoured to have had the opportunity to be a part of the Asian Institute of Technology and the GeoInformatics Center, and I am grateful for the experiences and knowledge gained during my internship. It has been an enriching and fulfilling journey that has deepened my passion for research and furthered my understanding of the importance of leveraging technology for environmental conservation.

TABLE OF CONTENT

IMPLEMENTATION OF YOLO VARIANTS

1. INTRODUCTION

YOLO (You Only Look Once) is a popular object detection algorithm that revolutionised the field of computer vision. Unlike traditional object detection methods that involve multiple stages and complex pipelines, YOLO takes a different approach by framing the object detection task as a single regression problem.

The YOLO algorithm divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. It simultaneously performs object detection and classification in a unified framework. YOLO can detect multiple objects in a single pass of the neural network, hence the name "You Only Look Once."

Here's a brief overview of the YOLO algorithm:

1. Input Processing: The input image is divided into a grid of cells, typically with dimensions like 7x7 or 13x13. Each grid cell is responsible for predicting bounding boxes and class probabilities for objects present within it.
2. Bounding Box Prediction: For each grid cell, YOLO predicts a fixed number of bounding boxes. Each bounding box consists of five attributes: the coordinates of the box's centre, width, height, and confidence score. The confidence score represents the probability of an object being present within the box.
3. Class Prediction: Along with bounding box predictions, YOLO also predicts class probabilities for each grid cell. It assigns a probability to each predefined class that represents the likelihood of the detected object belonging to that class.
4. Non-Maximum Suppression: YOLO employs a post-processing step called non-maximum suppression to eliminate redundant and overlapping bounding box predictions. It selects the most confident bounding box for each object based on the confidence scores and a predefined threshold.

YOLO's key advantages include its real-time performance, as it can process images or videos at impressive speeds, and its ability to detect objects with good accuracy, even for small objects. However, it may struggle with detecting objects that are closely clustered or have significant overlaps.

The YOLO algorithm has evolved over the years, with different versions such as YOLOv2, YOLOv3, and YOLOv4, each improving upon the previous version's performance and capabilities. These advancements have made YOLO a popular choice for a wide range of applications, including autonomous driving, surveillance systems, and object tracking.

For this project I have implemented YOLOv5 and YOLONas.

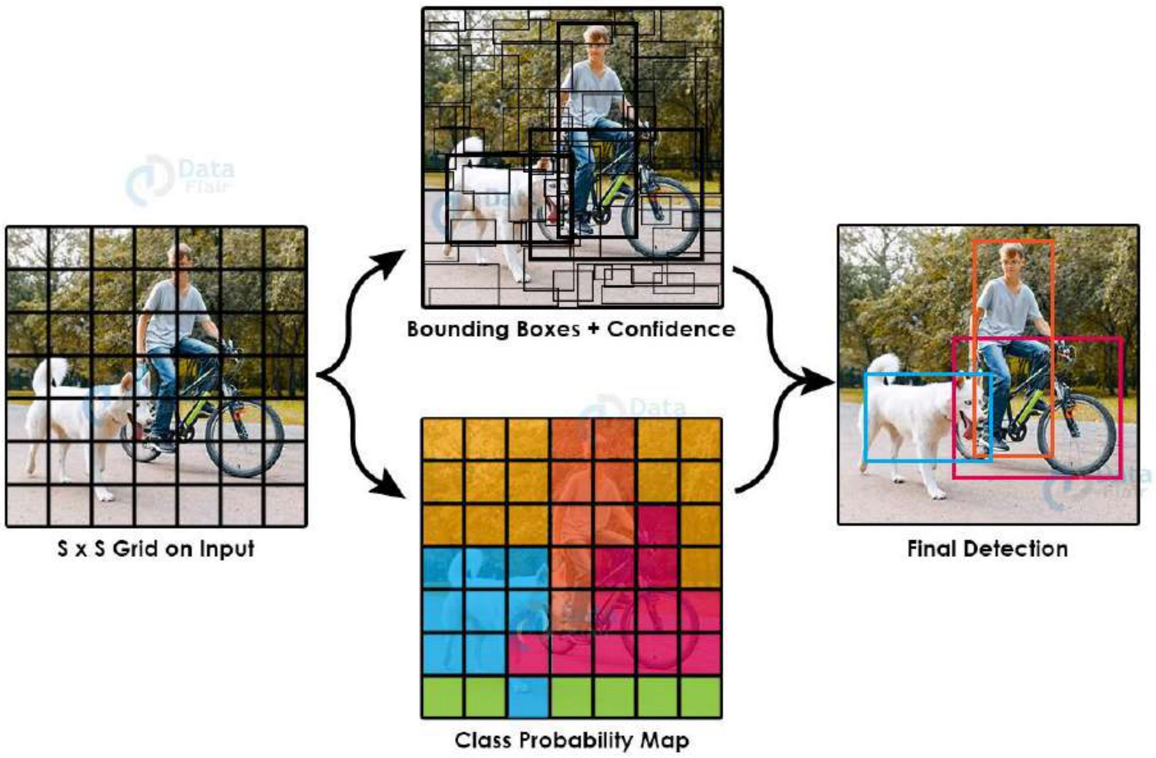


Fig: Working of YOLO algorithm

YOLOv5 is an upgraded version of the popular object detection algorithm, YOLO (You Only Look Once). It offers improved accuracy and speed compared to its predecessors. YOLOv5 introduces a streamlined architecture and incorporates advanced training techniques, such as data augmentation and multi-scale training, to enhance object detection performance. It provides different model sizes to balance accuracy and speed requirements. YOLOv5 is highly customizable, allowing researchers and developers to fine-tune the model for their specific datasets and applications. It has gained popularity for its user-friendly framework, real-time performance, and wide range of applications in computer vision.

Deci is thrilled to announce the release of a new object detection model, YOLO-NAS – a game-changer in the world of object detection, providing superior real-time object detection capabilities and production-ready performance. Deci's mission is to provide AI teams with tools to remove development barriers and attain efficient inference performance more quickly. The new YOLO-NAS delivers state-of-the-art (SOTA) performance with unparalleled accuracy-speed performance, outperforming other models such as YOLOv5, YOLOv6, YOLOv7, and YOLOv8.

2. DATASET

The YOLOv5 TXT dataset format is a text-based format used for annotating object detection datasets that are compatible with the YOLOv5 algorithm. It provides a standardised way of representing bounding box annotations and class labels for objects within an image.

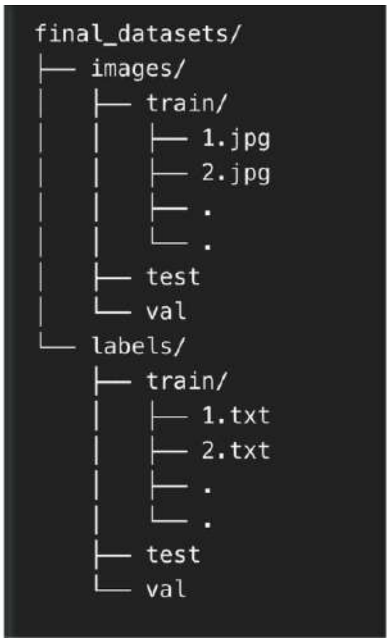
YOLOv5 TXT dataset format includes:

a. File Structure: Each image in the dataset is associated with a separate text file with the same name as the image file, but with a ".txt" extension.

b. Annotation Format: Each line in the TXT file represents a single object annotation and follows the format: class_id x_center y_center width height, where:
    - i. class_id is an integer representing the class label or category of the object.
    - ii. x_center and y_center are the normalised coordinates of the object's centre point relative to the image width and height, respectively.
    - iii. width and height are the normalised dimensions of the object's bounding box relative to the image width and height, respectively.

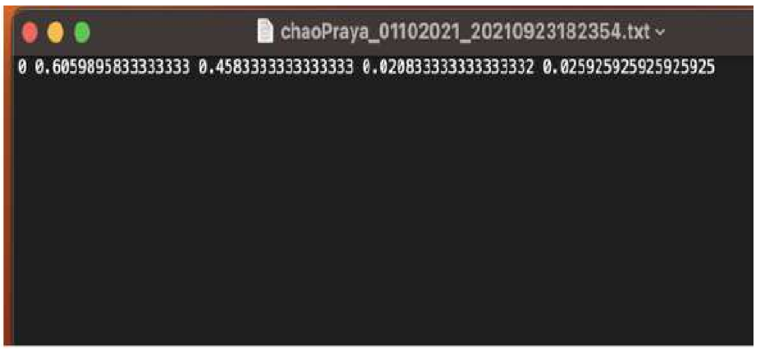
c. Normalisation: The coordinates and dimensions in the TXT file are normalised between 0 and 1, representing the relative position and size of the objects within the image.

d. Class Labelling: The class_id corresponds to a specific class label or category assigned to the object. Typically, each unique class label is assigned a unique integer value.



```
final_datasets/
├── images/
│   ├── train/
│   │   ├── 1.jpg
│   │   ├── 2.jpg
│   │   ├── .
│   │   └── .
│   ├── test
│   └── val
└── labels/
    ├── train/
    │   ├── 1.txt
    │   ├── 2.txt
    │   ├── .
    │   └── .
    ├── test
    └── val
```

Fig: YOLO TXT Format



```
0 0.6059895833333333 0.4583333333333333 0.0208333333333332 0.025925925925925925
```

Fig: YOLO TXT image label

3. RESULTS AND EVALUATION

a. YOLOv5

A batch size of 8 is chosen, which means that the model updates its parameters based on the gradients calculated from 8 images at a time. Furthermore, the training process spans over 40 epochs, indicating that the entire dataset is iterated 40 times during training.

The evaluation of the trained model is typically measured using metrics such as mean average precision (mAP). mAP50 represents the average precision at a threshold of 50%, which is commonly used for object detection evaluation. In this case, **the mAP50 score achieved is 0.779**, indicating a relatively high level of accuracy in detecting objects.

Additionally, mAP50-90 is another evaluation metric that represents the average precision over a range of thresholds, from 50% to 90%. **The mAP50-90 score achieved is 0.467**, which indicates the model's performance across a broader range of thresholds.

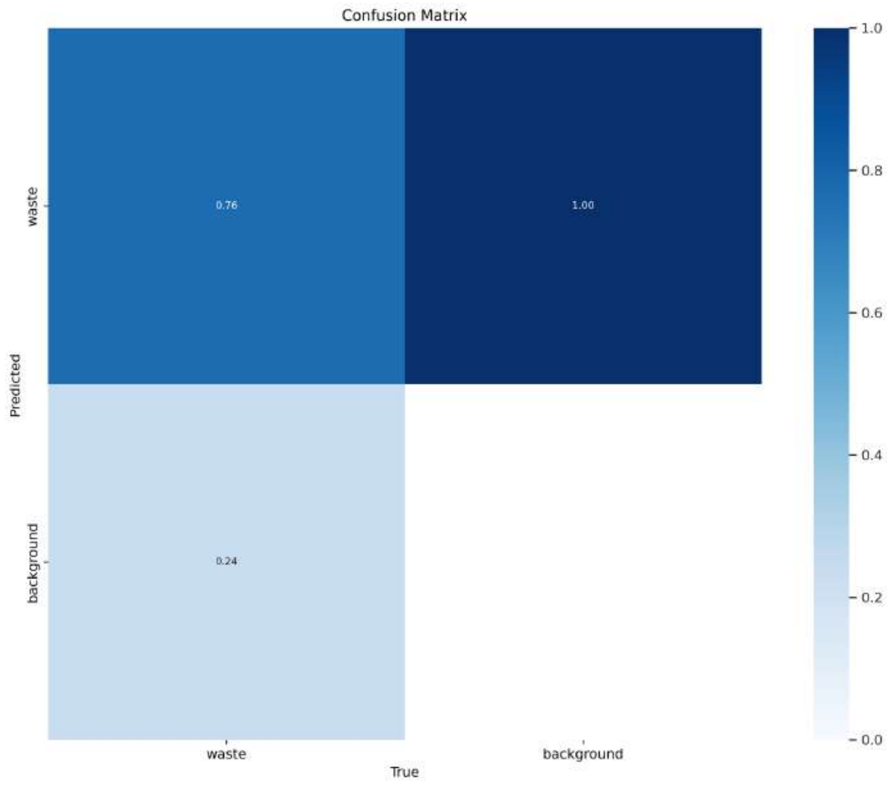


Fig: YOLOv5l confusion matrix

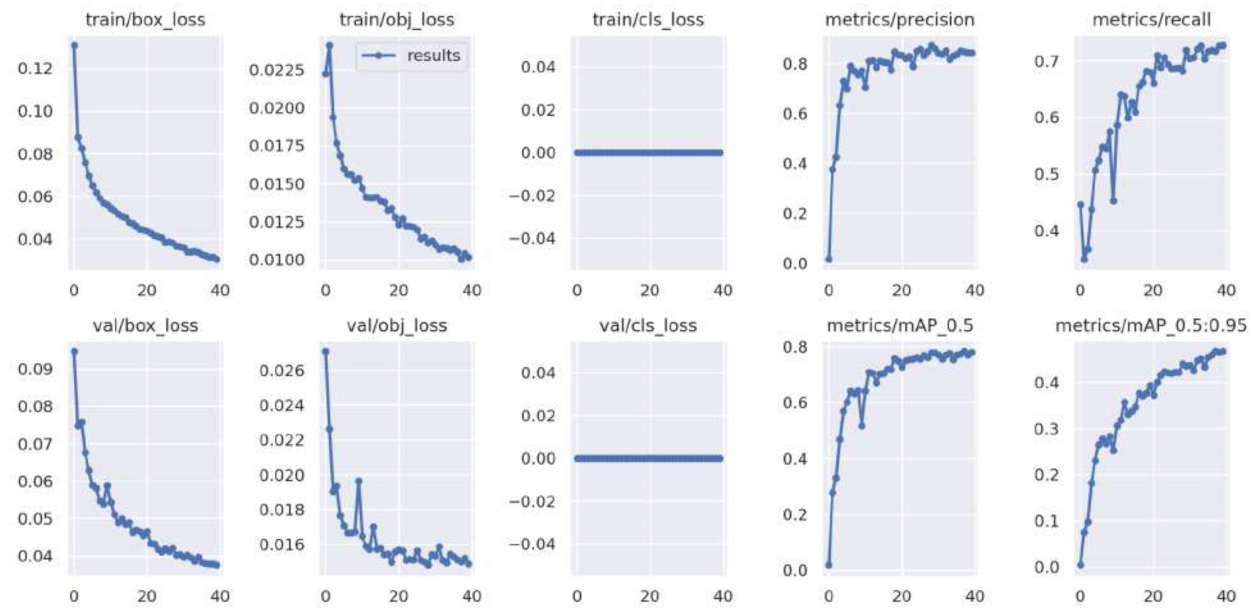


Fig: YOLOv5 results

Fig: Original Image

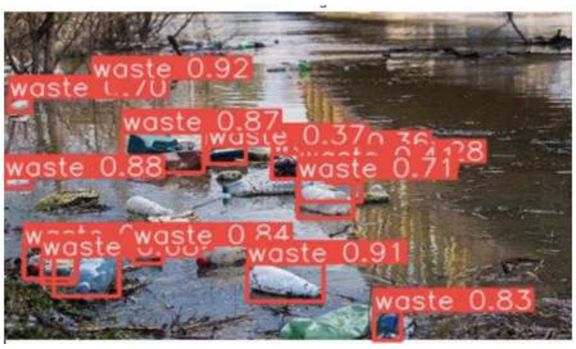


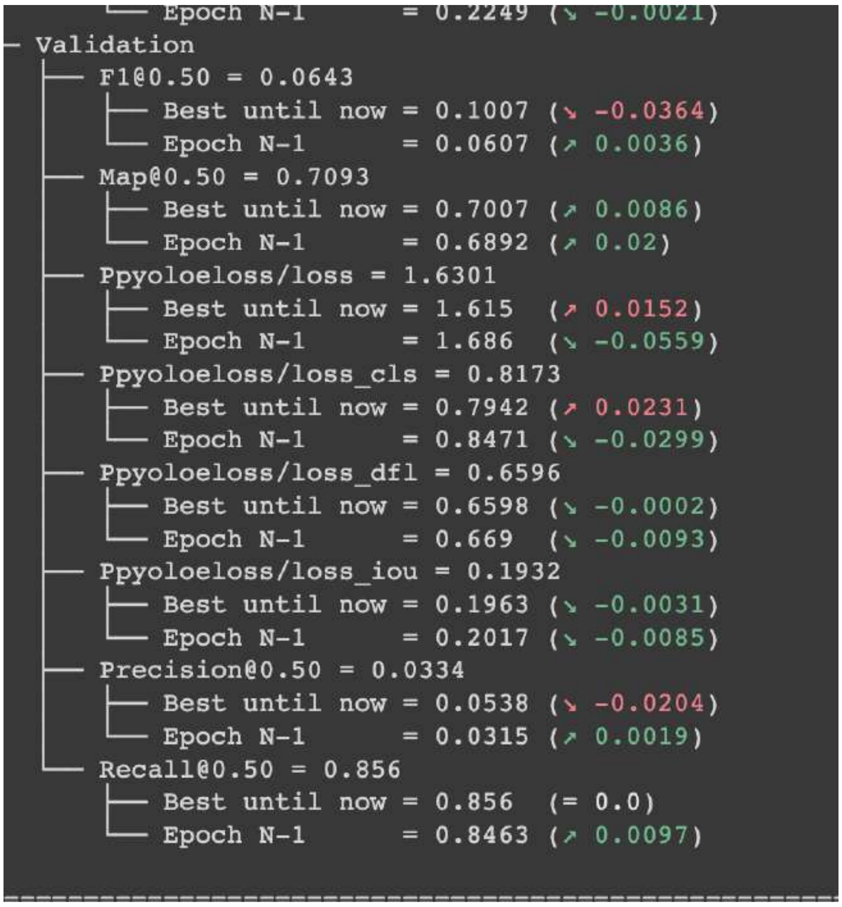
Fig: YOLOv5 prediction

b. YOLO-NAS

A batch size of 8 was chosen to balance memory usage and computational efficiency during training. The model was trained for a total of 20 epochs, indicating that the entire dataset was passed through the model 20 times during training.

The initial learning rate (lr) was set to **5e-4**, a relatively small value that allowed for a careful exploration of the parameter space. The learning rate was then adjusted using a cosine annealing schedule, with a final cosine learning **rate ratio of 0.1**. This approach gradually reduced the learning rate over time, allowing the model to converge towards optimal performance.

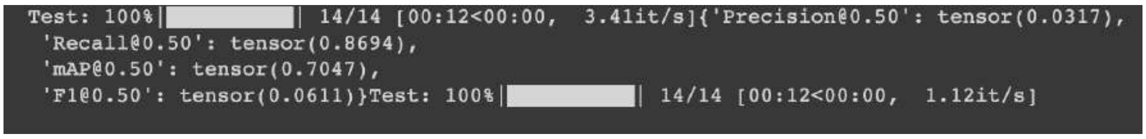
The performance of the trained model was evaluated using mean average precision (mAP50) as the evaluation metric. The model achieved an **mAP50 score of 0.704** on the test data, indicating that it performed well in accurately detecting and localising objects in previously unseen images. Additionally, the model achieved an **mAP50 score of 0.7** on the validation data, suggesting its generalisation ability and consistency across different subsets of the dataset.



```
    └── Epoch N-1       = 0.2249  (↘ -0.0021)
─ Validation
    ├── F1@0.50 = 0.0643
    │   ├── Best until now = 0.1007 (↘ -0.0364)
    │   └── Epoch N-1      = 0.0607 (↗ 0.0036)
    ├── Map@0.50 = 0.7093
    │   ├── Best until now = 0.7007 (↗ 0.0086)
    │   └── Epoch N-1      = 0.6892 (↗ 0.02)
    ├── Ppyoloeloss/loss = 1.6301
    │   ├── Best until now = 1.615  (↗ 0.0152)
    │   └── Epoch N-1      = 1.686  (↘ -0.0559)
    ├── Ppyoloeloss/loss_cls = 0.8173
    │   ├── Best until now = 0.7942 (↗ 0.0231)
    │   └── Epoch N-1      = 0.8471 (↘ -0.0299)
    ├── Ppyoloeloss/loss_dfl = 0.6596
    │   ├── Best until now = 0.6598 (↘ -0.0002)
    │   └── Epoch N-1      = 0.669  (↘ -0.0093)
    ├── Ppyoloeloss/loss_iou = 0.1932
    │   ├── Best until now = 0.1963 (↘ -0.0031)
    │   └── Epoch N-1      = 0.2017 (↘ -0.0085)
    ├── Precision@0.50 = 0.0334
    │   ├── Best until now = 0.0538 (↘ -0.0204)
    │   └── Epoch N-1      = 0.0315 (↗ 0.0019)
    └── Recall@0.50 = 0.856
        ├── Best until now = 0.856  (= 0.0)
        └── Epoch N-1      = 0.8463 (↗ 0.0097)

===============================================
```

Fig: YOLO-NAS training evaluation



```
Test: 100%|██████████| 14/14 [00:12<00:00,  3.41it/s]{'Precision@0.50': tensor(0.0317),
 'Recall@0.50': tensor(0.8694),
 'mAP@0.50': tensor(0.7047),
 'F1@0.50': tensor(0.0611)}Test: 100%|██████████| 14/14 [00:12<00:00,  1.12it/s]
```

Fig: YOLO-NAS Test evaluation

Fig: Original Image

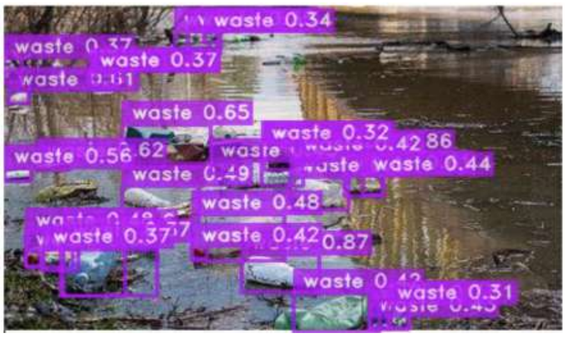


Fig: YOLO-NAS prediction

IMPLEMENTATION OF MASK R-CNN

1. INTRODUCTION

Mask R-CNN (Region-based Convolutional Neural Network) is an advanced instance segmentation algorithm that extends the capabilities of the popular Faster R-CNN object detection framework. It was developed to tackle the task of pixel-level segmentation, where the goal is not only to detect objects but also to accurately delineate their boundaries and classify each pixel within them.

Mask R-CNN combines the power of region proposal networks (RPNs) and fully convolutional networks (FCNs) to achieve both object detection and instance segmentation in a single end-to-end framework.

Brief overview of the Mask R-CNN algorithm:

1. Backbone Network: Similar to other deep learning-based models, Mask R-CNN starts with a backbone network, such as ResNet or VGG, which extracts rich features from the input image.
2. Region Proposal Network (RPN): Mask R-CNN utilises an RPN to generate region proposals or candidate object bounding boxes. The RPN suggests potential regions of interest that are likely to contain objects, based on predefined anchor boxes and their associated confidence scores.
3. RoI Align: Unlike previous methods, Mask R-CNN introduces a RoI (Region of Interest) Align layer that accurately extracts features from each region proposal, preserving spatial information and avoiding misalignments.
4. Mask Head: Mask R-CNN includes an additional branch, called the mask head, which is responsible for predicting masks at the pixel level for each proposed region. The mask head takes the region features as input and performs fully convolutional operations to generate binary masks for each object instance.
5. Classification and Bounding Box Regression: Alongside instance segmentation, Mask R-CNN also predicts class probabilities for each proposed region and performs bounding box regression to refine the localization of the objects.

By incorporating these components, Mask R-CNN is able to achieve precise object detection, accurate instance segmentation, and class predictions simultaneously. It has demonstrated remarkable performance in a variety of applications, including semantic segmentation, object recognition, and image understanding tasks.

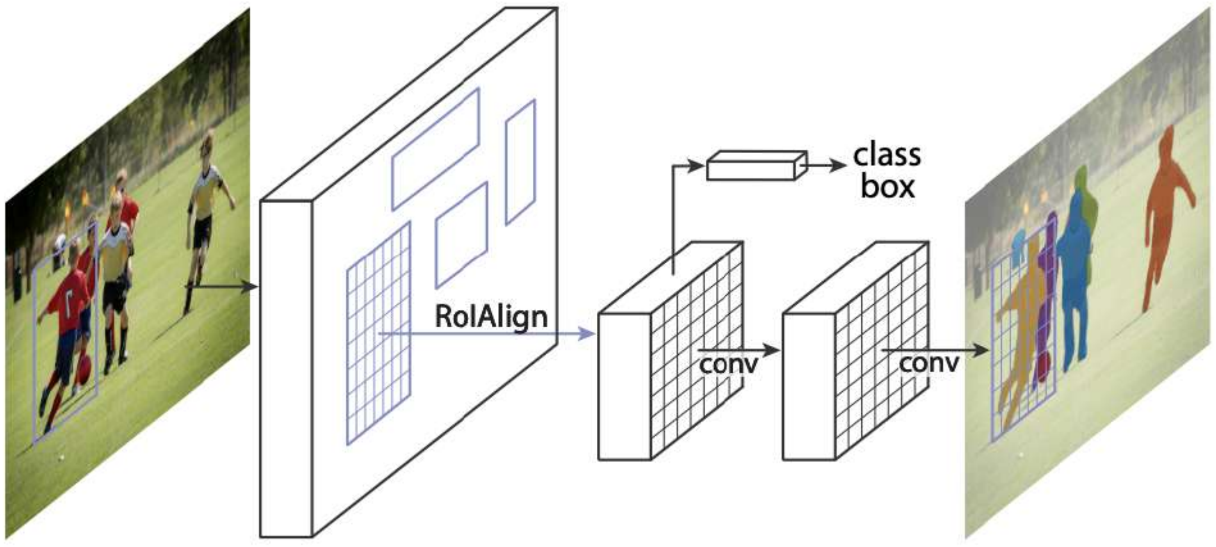


Fig: Architecture of Mask R-CNN

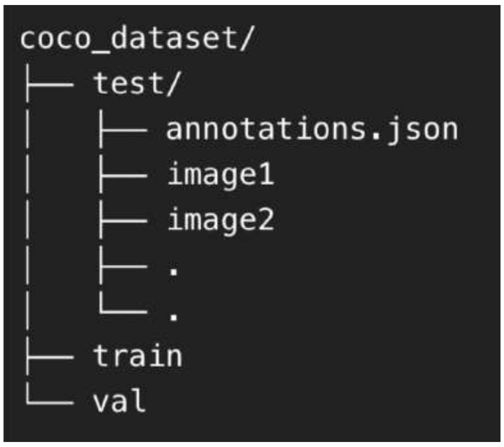
2. DATASETS

The COCO (Common Objects in Context) dataset is a widely used benchmark dataset for object detection, instance segmentation, and other computer vision tasks. It provides a comprehensive and diverse collection of images that depict complex scenes with a variety of object categories and annotations.
Here's a brief explanation of the COCO dataset:

1. Image Collection: The COCO dataset consists of a vast collection of images covering a wide range of real-world scenes. It contains images captured in different environments, such as indoor, outdoor, and natural settings. The dataset includes a significant number of images, making it suitable for training and evaluating deep learning models.
2. Object Annotations: The COCO dataset is known for its precise and detailed annotations. Each image in the dataset is annotated with bounding boxes that tightly enclose the objects of interest. Additionally, it provides pixel-level segmentation masks for each annotated object, enabling instance segmentation tasks. The annotations also include object category labels, allowing for object classification and categorization.
3. Object Categories: The COCO dataset encompasses a large number of object categories, ranging from common everyday objects to more specific items. It covers a wide range of classes, including people, animals, vehicles, household items, and more. The dataset's diversity ensures that models trained on COCO can handle a broad spectrum of object types and scenarios.

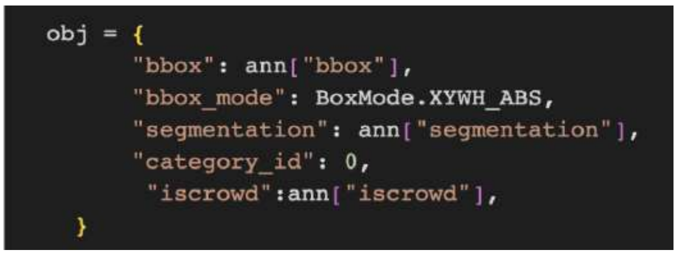
4. Evaluation Metrics: COCO provides standardised evaluation metrics for object detection, instance segmentation, and other tasks. The most commonly used metric for object detection is the mean Average Precision (mAP), which considers the precision and recall of the predicted bounding boxes. This allows researchers to compare and assess the performance of different models on a common benchmark.

The COCO dataset has become a standard reference for evaluating and benchmarking the performance of various computer vision algorithms. Its rich annotations, diverse image collection, and comprehensive evaluation metrics make it a valuable resource for developing and assessing state-of-the-art models in the field of object detection, instance segmentation, and related tasks.



```
coco_dataset/
├── test/
│   ├── annotations.json
│   ├── image1
│   ├── image2
│   ├── .
│   └── .
├── train
└── val
```

Fig:: COCO Dataset Directory Structure



```
obj = {
        "bbox": ann["bbox"],
        "bbox_mode": BoxMode.XYWH_ABS,
        "segmentation": ann["segmentation"],
        "category_id": 0,
         "iscrowd":ann["iscrowd"],
   }
```

Fig: COCO Annotation Features

I implemented Mask R-CNN in Detectron2 which is an open-source deep learning framework developed by Facebook. It is built on top of PyTorch and has state-of-the-art models and algorithms. It provides a rich set of tools and utilities for data loading, model evaluation, and visualisation, simplifying the development process. Also, Detectron2 has an active community and is continuously updated with the latest research advancements.

3. RESULT AND EVALUATION

The **batch size is set to 8**, which means that during each iteration of training, the model processes eight images at a time. The batch size affects training efficiency and memory consumption. A larger batch size can lead to faster training but may require more memory.

The learning rate **(LR) is set to 0.00025**, which determines the step size at which the model learns from the training data. A smaller learning rate allows the model to make smaller adjustments during training. The learning rate is often adjusted during training to achieve better convergence.

The training process is performed **for 500 iterations**. Each iteration represents one forward and backward pass of the training data through the neural network. Iterations capture the progress of the model during training, allowing it to gradually improve its performance over time.

The evaluation metrics provided are Bbox AP50 and Segmentation AP50. Bbox AP50 represents the average precision at an intersection over union (IoU) threshold of 0.5 for bounding box predictions. It measures the accuracy of the bounding box localization. Segmentation AP50 represents the average precision at an IoU threshold of 0.5 for segmentation predictions. It evaluates the accuracy of the instance segmentation masks generated by the model.

The model achieves a **Bbox AP50 of 40.58** and a **Segmentation AP50 of 41.785.** These metrics indicate the performance of the model in terms of object localization and segmentation accuracy, respectively. The higher the AP50 values, the better the model's performance in these specific evaluation metrics.

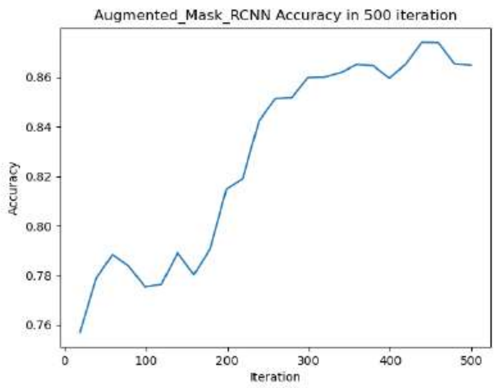


Fig: Mask Accuracy

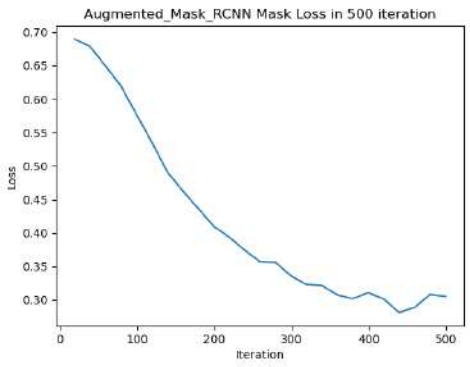


Fig: Mask Loss

Fig: Original Image

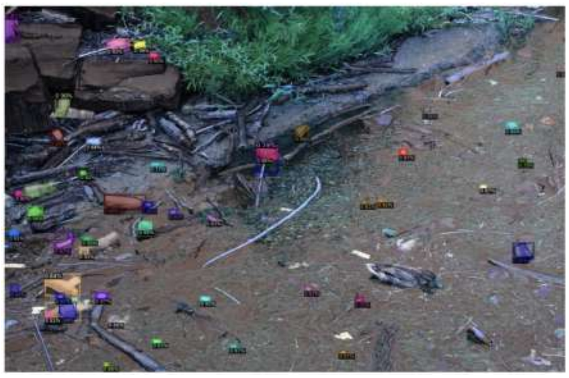


Fig: Masked Image

IMPLEMENTATION OF RETINANET

1. INTRODUCTION

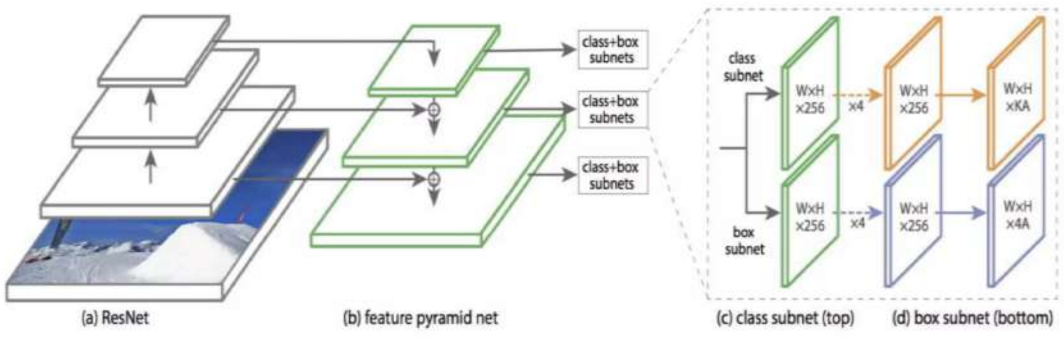
RetinaNet is an object detection algorithm that addresses the challenge of detecting objects at different scales and handling the problem of class imbalance in training data. It was introduced by researchers at Facebook AI Research.

The key innovation of RetinaNet lies in its novel feature pyramid network (FPN) architecture and a focal loss function. The FPN enables the network to capture features at multiple scales, allowing accurate detection of objects of various sizes. It combines feature maps from different layers of a backbone network to create a feature pyramid that maintains both high-resolution details and semantic information.

The focal loss function is designed to mitigate the problem of class imbalance inherent in object detection datasets, where the number of background (non-object) samples far exceeds the number of object samples. The focal loss assigns higher weights to challenging samples that are difficult to classify correctly, thus emphasising the learning of hard examples and reducing the dominance of easy background samples.

RetinaNet consists of a backbone network, a feature pyramid network (FPN), and two sibling subnetworks: a classification subnet and a regression subnet. The classification subnet predicts object classes, while the regression subnet predicts accurate bounding box coordinates for each object.

The architecture of RetinaNet allows it to efficiently detect objects of various scales and maintain a good balance between accuracy and speed. It has shown strong performance in object detection tasks, particularly in scenarios with a wide range of object sizes. RetinaNet has been widely adopted and serves as the basis for many state-of-the-art object detection models.



RetinaNet

Fig: Architecture of RetinaNet

2. DATASETS

The same dataset format as the Mask R-CNN model was applied. The directory structure and annotation features remained unchanged.

3. RESULTS AND EVALUATION

A **batch size of 8** means that eight images are fed into the model at a time. This setting balances the computational efficiency and memory requirements, enabling efficient training of the model.

The learning rate (LR) is a key hyperparameter that controls the step size at which the model updates its weights during training. A **learning rate of 0.00025** is chosen, and the model is trained for 500 iterations. This combination of LR and iterations helps the model converge gradually and improve its accuracy over time.

To evaluate the trained model, Bbox AP50 is used. Bbox AP50 measures the average precision (AP) of bounding box predictions at a threshold of 50% intersection over union (IoU). A **Bbox AP50 value of 26.79** indicates the accuracy of the model in correctly localising and identifying objects in the evaluation dataset.

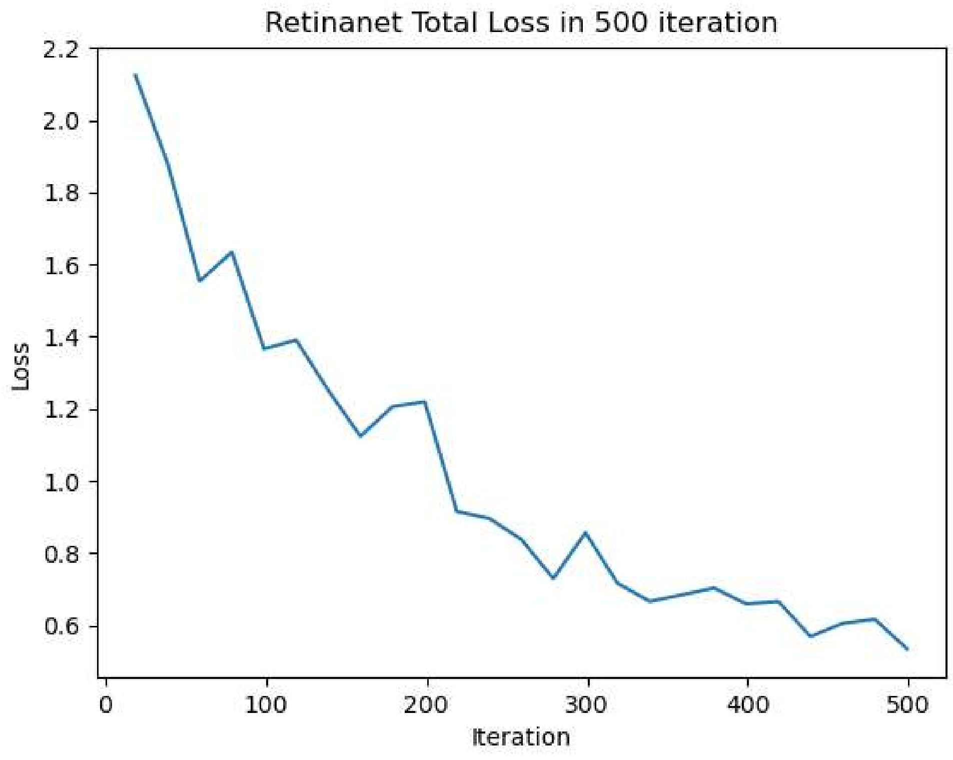


Fig: RetinaNet Total Loss

Fig: Original Image

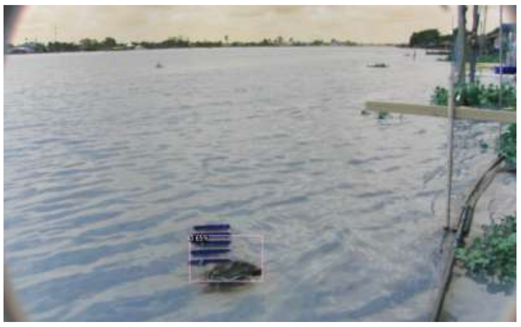
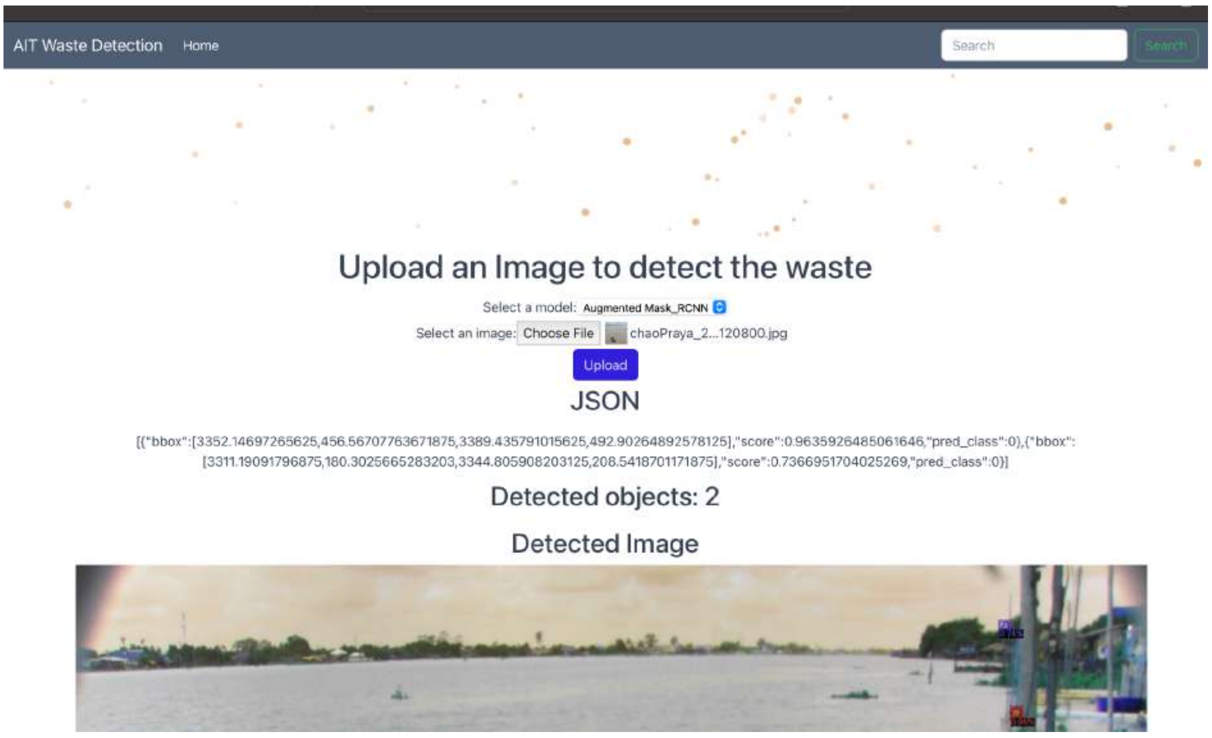


Fig: Detected Image

API AND DEPLOYMENT

API for each model is created with FAST API. Any model can be used with the API endpoints. Only the Yolov5 model API is hosted at Render. Other models could not be hosted due to free plan limitations. Simple UI is made with React JS and hosted at Netlify.



Upload an Image to detect the waste

Select a model: Augmented Mask_RCNN

Select an image: Choose File chaoPraya_2...120800.jpg

JSON

[{"bbox":[3352.14697265625,456.56707763671875,3389.435791015625,492.90264892578125],"score":0.9635926485061646,"pred_class":0},{"bbox":[3311.19091796875,180.3025665283203,3344.805908203125,208.5418701171875],"score":0.7366951704025269,"pred_class":0}]

Detected objects: 2

Detected Image

Fig: UI to upload image and get model prediction results

CONCLUSION

YOLO variants, Mask- RCNN and RetinaNet are implemented in a dataset of plastic litter images. Dataset modifications and augmentations are made as required. Detectron2 is used to implement Mask R-CNN and RetinaNet. COCO evaluation is performed and the results and related graphs are also obtained. Finally, the model API endpoint is created and the UI is designed for easy image upload. The predicted image is obtained from API.