CAPSTONE PROJECT END-TERM REPORT

on

Moving Object Detection, Tracking and Counting using YOLO

Submitted by

Shray Gupta (1032190368)
Rishabh Garg(1032191511)

Under the Guidance of

Dr. Anuradha Phadke

School of Electronics & Communication Engineering

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY, PUNE.

[2022-2023]

SCHOOL OF
ELECTRONICS & COMMUNICATIONENGINEERING

Academic Year 2022-2023

CERTIFICATE

This is to certify that **Rishabh Garg** (1032191511), **Shray Gupta** (1032190368) has successfully completed his/her

Capstone Project entitled "**Moving Object Detection, Tracking and Counting using YOLO**" and submitted the same during the academic year 2022-23 towards the partial fulfillment of degreeof **Bachelor of Technology in Electronics & Communication Engineering** as per the guidelines prescribed by Dr. Vishwanath Karad MIT World Peace University, Pune.

Project Guide	Batch Coordinator	Project Coordinator	Head of School
Dr. Anuradha Phadke	Dr. V. V. Deshmukh	Dr. Alka Barhatte	Dr. Vinaya Gohokar

Date:18/05/23

Place: Pune

DECLARATION

We the undersigned, declare that the work carried under Capstone Project Phase-II entitled **"Moving Object Detection, Tracking and counting using YOLOv8"** represents our idea in our own words. We have adequately cited and referenced the original sources where other ideas or words have been included. We also declare that We have adhered to all principles of academic honesty and integrity and have not misprinted or fabricated or falsified any ideas/data/fact/source in our submission. I/We understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or whom proper permission has not been taken when needed.

Date:18/05/2023

Place:Pune

PRN Number	Name of student	Signature with date
1032190368	Shray Gupta	
1032191511	Rishabh Garg	

Project Guide	Batch Coordinator	Project Coordinator	Head of School
Dr. Anuradha Phadke	Dr. V. V. Deshmukh	Dr. Alka Barhatte	Dr. Vinaya Gohokar

Date:18/05/23

Place: Pune

ACKNOWLEDGEMENT

I would like to express my profound gratitude to Dr. Vinaya Gohokar, Head of School, ECE, MITWPU university for her contributions to the completion of my project titled Moving **Moving Object Detection,Tracking and counting using YOLOv8.**

I would like to express my special thanks to my project guide Dr. Anuradha Phadke and batch coordinator Dr. V. V. Deshmukh for their time and efforts she provided throughout the term. Your useful advice and suggestions were helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Signature

Name

Table of Contents

ABSTRACT

This project involves implementing a moving object detection and counting system using YOLO. The goal of the project is to detect and track moving objects.
The system is built using various deep learning techniques and computer vision algorithms, including object detection, motion estimation, and feature extraction. These techniques are implemented using Python and various open-source libraries, such as TensorFlow, OpenCV, and CUDA.

The project also involves installing the necessary softwares and drivers, connecting the camera, and optimizing the performance of the system. The final system is capable of detecting and tracking moving objects with high accuracy and speed, and can be used for a wide range of applications, includeing surveillance, robotics and autonomous vehicles.

LIST OF FIGURES

ABBREVIATIONS

CUDA	Compute Unified Device Architecture
SOTA	State of The Art
YOLO	You Only Look Once
CVAT	Computer Vision Annotation Tool

List Of Tables

CHAPTER 1
INTRODUCTION

1.1 Introduction

Moving object detection is an essential task in many applications, such as surveillance, robotics, and autonomous vehicles. The ability to detect and track moving objects in real-time is crucial for ensuring the safety and security of people and assets.

With the advent of small and powerful single-board computers, such as Jetson Nano, it has become easier to build complex AI and robotics systems that can perform tasks such as moving object detection with high accuracy and speed. Jetson Nano is a small but powerful computer designed specifically for AI and robotics applications, and it is equipped with an NVIDIA GPU and a range of other hardware components that make it ideal for running complex deep learning algorithms.

In this project, we aim to build a moving object detection system using Jetson Nano and a camera connected to it. We will be using various deep learning techniques and computer vision algorithms to detect and track moving objects in real-time, and to generate alerts or trigger actions based on the detected motion. We will also be exploring different hardware configurations and optimizations to achieve the best possible performance of the system.

The project has the potential to be used in a wide range of applications, including surveillance, robotics, and autonomous vehicles, and can greatly improve the safety and security of people and assets in these settings.

CHAPTER 2
REVIEW OF LITERATURE

LITERATURE REVIEW

Moving object detection is a well-studied problem in computer vision and has been extensively researched over the past few decades. A review of the literature reveals several approaches and techniques that have been proposed for this task.

One of the most common approaches is background subtraction, which involves subtracting the background image from the current frame to detect the moving objects. Various algorithms have been proposed for background subtraction, such as the Gaussian Mixture Model (GMM) and the Adaptive Background Mixture Model (ABMM).

Another popular approach is optical flow, which involves estimating the motion of the pixels in an image over time. Various optical flow techniques have been proposed, such as Lucas-Kanade and Horn-Schunck.

Based on the provided literature papers on Moving Object Detection, Tracking, and Counting using YOLO, here are some common technologies and techniques mentioned:

- YOLO (You Only Look Once): The YOLO algorithm is a common technology mentioned in all the papers. YOLO is a real-time object detection algorithm that uses a single neural network to directly predict bounding boxes and class probabilities.
- Deep Learning: Deep learning techniques, particularly convolutional neural networks (CNNs), are employed in these papers for training and inference tasks. Deep learning models are trained to detect, track, and count moving objects in images and videos.
- Object Detection: The papers focus on moving object detection using YOLO. The YOLO algorithm divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell, enabling efficient and accurate object detection.
- Tracking Algorithms: Some papers discuss the integration of tracking algorithms with YOLO for robust object tracking across consecutive frames in videos. Techniques like Kalman filtering, Hungarian assignment algorithm, and IOU (Intersection over Union) association are mentioned for maintaining object identities during tracking.

Table 1: Literature Survey

Literature Referred	Summary
[1] "You Only Look Once: Unified, Real-Time Object Detection" by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. (CVPR 2016)	This is the original paper introducing the YOLO algorithm, which provides a unified approach for real-time object detection in images.
[2] "YOLO9000: Better, Faster, Stronger" by Joseph Redmon and Ali Farhadi. (CVPR 2017)	This paper presents an improved version of YOLO called YOLO9000, which combines object detection and classification into a single unified model.
[3] "Real-time object detection for smart vehicles" by Bo Li, Tianming Zhao, Wei Cai, Lingling Liang, and Xiaofei Xing. (Journal of Real-Time Image Processing 2018)	This paper discusses the application of the YOLO algorithm for real-time object detection in smart vehicles, focusing on moving object detection and tracking.
[4] "Moving object detection and tracking in aerial videos based on YOLO object detector and KCF tracker" by Bui Thai Minh, Dong-Seok Jeong, and Euntai Kim. (Multimedia Tools and Applications 2020)	This paper proposes a method for moving object detection and tracking in aerial videos using YOLO as the object detector and the Kernelized Correlation Filter (KCF) as the tracker.
[5] "Moving Object Detection Based on YOLO and Background Subtraction" by Tijana Radojević, Vladan Velisavljević, and Jelena Ćertić. (Sensors 2020)	This paper presents a moving object detection approach that combines YOLO with background subtraction techniques, aiming to improve the accuracy and robustness of the detection process.
[6] "Moving Object Detection and Tracking Using YOLO Algorithm and Optical Flow" by Shubham Bhamoriya, Pritee Khanna, and R. Balasubramanian. (International Journal of Advanced Computer Science and Applications 2020)	This paper proposes a method that integrates the YOLO algorithm with optical flow techniques for moving object detection and tracking, with a focus on real-time applications.

Table 1 describes the literatures referred and a brief about the literature, like the technologies used and the existing gap in this field.

Gaps in existing technology:

- Limited Dataset Availability: One major gap is the limited availability of large-scale, diverse, and annotated datasets specifically designed for fish detection. The scarcity of high-quality annotated data makes it challenging to train and evaluate accurate and robust fish detection models.
- Occlusion and Overlapping Objects: Fish in aquatic environments often exhibit occlusion due to vegetation, other fish, or underwater structures. Existing fish detection models often struggle to accurately detect fish in such challenging scenarios, resulting in lower detection rates and increased false positives.

Steps we tried to overcome the gaps:

- To address the existing gaps in fish detection research, we took specific steps to overcome the challenges. Firstly, we recognized the limited availability of high-quality annotated datasets and the need for diverse and representative data. To tackle this, we undertook the task of precisely annotating a comprehensive dataset for fish detection.

- To ensure the dataset's diversity, we visited aquariums that housed a wide range of fish species. By collecting data directly from these aquariums, we obtained images and videos capturing fish in controlled yet realistic environments. This approach allowed us to gather data with varying lighting conditions, background settings, and fish behaviors, which are crucial for training robust fish detection models.

AIM AND OBJECTIVES OF PROJECT

The aim of this project is object(Fish) detection using YOLO, The system will use various deep learning techniques and computer vision algorithms to detect and track moving objects in real-time, and generate alerts or trigger actions based on the detected motion.

The objectives of the project include:

1. Implementing various deep learning techniques and computer vision algorithms, such as background subtraction, optical flow, and convolutional neural networks, for moving object detection.

2. Evaluating the performance of the system in terms of accuracy, speed, and efficiency, and optimizing the system to achieve the best possible performance.
3. Testing the system in real-world scenarios and demonstrating its effectiveness for various applications, such as surveillance, robotics, and autonomous vehicles.
4. Documenting the project, including the hardware and software setup, the algorithms used, the performance evaluation, and the results of the testing, in a clear and concise report that can be used as a reference for future work in this area.

In conclusion, the aim and objectives of this project are to drive innovation, address key challenges, and achieve meaningful outcomes that contribute to the advancement of knowledge and create positive impact in the relevant field.

CHAPTER 3

SYSTEM DEVELOPMENT

SYSTEM BLOCK DIAGRAM

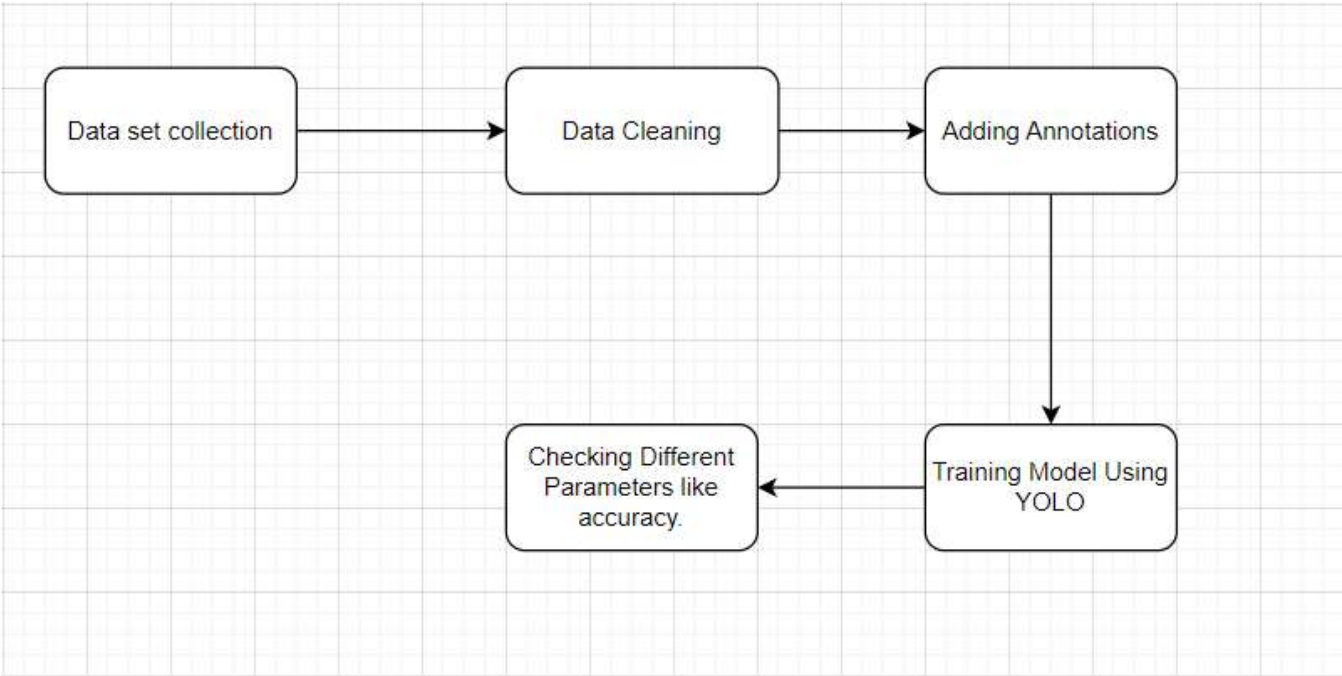

Figure 1: System Block Diagram

Figure 1 explains the system block diagram of the project, firstly we have collected a large dataset from different sources, after data collection we cleaned that data by removing unnecessary data like images of fish market. Once the dataset is ready, we have done the annotations on each mage manually using CVAT, then we used that annotated data in YOLO for model training and checked the accuracy of our model.

SYSTEM SPECIFICATION

1. Dataset Description

Table 2: Dataset Description

Number of Sources Referred	4
Number of images	1000
Number of classes	1

Table 2 describe the number of sources we have referred, annotated images we have used and the classes in our model.

Sources Referred:

- Google open images dataset version 7.
- Visited different aquariums.
- Collected sample amount dataset from a student working on kind of similar thing.
- Open Repositories like Kaggle
- Frame Size – 640*640

2. Hardware Requirements: The system requires sufficient storage capacity to store the pre-processed images and the trained models.
   Specifications of hardware we have used:
   - Ram;- 8GB
   - Storage:- 1TB
   - Cores:- 4

3. Software Requirements: The proposed system requires a Linux or Windows operating system, Python programming language, and Python libraries will be used to develop and train the deep learning models, perform image processing tasks, and visualize the results.
   Major Python Libraries Used:- Ultralytics, YOLO, OpenCV .

CHALLENGES FACED/COMPLEXITIES INVOLVED

Embarking on this project presented a multitude of challenges and complexities that demanded careful consideration and strategic problem-solving.

- **Data availability**: A deep learning model requires a large amount of labeled data to train effectively. However, obtaining such data for object detection can be difficult and time-consuming.
- **Annotation quality**: The quality of the annotations (bounding boxes, labels, etc.) used to train the model is crucial. Incorrect or inconsistent annotations can result in a poorly performing model.
- **Overfitting:** Deep learning models have a tendency to overfit the training data, i.e., perform well on the training data but poorly on new, unseen data. This can be mitigated by using techniques such as data augmentation, regularization, and early stopping.

- **Model architecture selection**: There are many different architectures available for object detection, each with its own strengths and weaknesses. Choosing the right architecture for the task at hand can be challenging and requires knowledge and expertise in deep learning.

Despite the formidable challenges and complexities encountered throughout this project our collaborative efforts have enabled us to overcome these obstacles and achieve valuable insights and outcomes

Chapter-4
SYSTEM IMPLEMENTATION

In our fish detection system using YOLO, we utilized a dataset consisting of a total of 1000 images. To train our model, we carefully selected 750 images from the dataset, ensuring a diverse representation of fish species, environments, and imaging conditions.

To evaluate the performance of our system, we set aside 150 images as a dedicated test set. These test images were not used during training and allowed us to assess the generalization capability of our model on unseen data.

Additionally, we reserved 100 images from the dataset for the purpose of validation. This validation set was used during the training process to monitor the model's progress, tune hyperparameters, and prevent overfitting.

4.1 PRE-PROCESS FOR TRAINING AND DATASET FEATURE EXPLORATION

Before training a deep learning model for moving object detection, it is important to perform some pre-processing steps on the dataset to ensure that the data is clean, consistent, and suitable for training.

The pre-processing steps for training the moving object detection model can include:

1. Data cleaning: This involves removing any irrelevant or redundant data from the dataset, such as duplicate frames or frames with no moving objects.
2. Data augmentation: This involves generating additional training data by applying various transformations to the existing dataset, such as rotations, translations, and flips. Data augmentation helps to increase the size of the dataset and improve the generalization ability of the model.
3. Feature extraction: This involves extracting relevant features from the dataset to enable accurate detection and tracking of moving objects. Common features used for moving object detection include color, texture, and motion features.
4. Labeling: This involves annotating the dataset with ground truth labels to enable supervised learning. The ground truth labels indicate the location and class of the moving objects in the frames.
5. Dataset splitting: This involves splitting the dataset into training, validation, and testing sets. The training set is used to train the model, while the validation set is used to tune the model hyperparameters, and the testing set is used to evaluate the model performance.

Dataset feature exploration involves analyzing the characteristics of the dataset to gain insights into the underlying distribution and variability of the data. This can involve performing various statistical analyses, such as mean, variance, and correlation, on the dataset to identify any patterns or trends.
In addition, visualization techniques such as scatter plots, histograms, and heatmaps can be used to identify any outliers, class imbalance, or noise in the dataset. These insights can be used to inform the choice of the appropriate deep learning model architecture, hyperparameters, and optimization techniques for training the moving object detection model.

4.2 Ultralytics

Ultralytics, a leading technology company in the field of computer vision and deep learning, is revolutionizing the way we perceive and interact with visual data.

1. Ultralytics is a computer vision research organization that specializes in developing state-of-the-art deep learning algorithms and tools for object detection, tracking, and related tasks. Their work primarily revolves around the popular YOLO (You Only Look Once) algorithm, which is widely used for real-time object detection in images and videos.
2. The Ultralytics team has created an open-source software library called "YOLOv8" that builds upon the success of the original YOLO algorithm. YOLOv8 offers significant advancements and improvements, including increased accuracy, faster inference times, and a streamlined codebase for ease of use.
3. Key Features of YOLOv8 by Ultralytics:
   1. Architecture: YOLOv8 employs a fully convolutional architecture that allows end-to-end training and inference on both images and videos. It uses a single deep neural network to directly predict bounding boxes and class probabilities, eliminating the need for region proposals and subsequent refinement steps.
   2. Model Variants: YOLOv8 offers different model variants with varying sizes, such as YOLOv8n ,YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. These variants have different model complexities, enabling users to choose an appropriate balance between accuracy and computational efficiency based on their specific requirements.

3. Object Detection: YOLOv8 achieves high-quality object detection by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. It uses anchor boxes of different shapes and sizes to handle objects with diverse aspect ratios. YOLOv8 also incorporates various advanced techniques like focal loss, multi-scale training, and data augmentation to improve detection performance.

4. Object Tracking: Ultralytics has integrated object tracking capabilities into YOLOv8, allowing for real-time tracking of objects across consecutive frames in a video. This feature utilizes techniques such as Hungarian assignment algorithm, and IOU (Intersection over Union) association to maintain object identities and track them robustly.

5. Extensibility and Integration: YOLOv8 is designed to be highly modular and easily extensible. It can be integrated with other computer vision libraries like OpenCV and PyTorch, enabling seamless integration into existing workflows and facilitating customization for specific applications.

6. Training and Inference: Ultralytics provides comprehensive scripts and tools for training and inference with YOLOv8.The library supports various training configurations where models pre-trained on large-scale datasets can be fine-tuned on specific object detection tasks.

Table 3:Different Models of YOLOv8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Table 3 describe the performance of different versions of YOLOv8 .

4.3 YOLO

- YOLO (You Only Look Once) is a SOTA object detection algorithm in computer vision and deep learning. It was first introduced in 2016 by Joseph Redmon, and since then, it has become a popular and widely used algorithm for real-time object detection.
- YOLO is based on a convolutional neural network (CNN) architecture and can detect multiple objects in an image or a video stream simultaneously. Unlike other object detection algorithms, YOLO processes the entire image in one forward pass through the neural network, which makes it faster and more efficient than traditional methods.
- The YOLO algorithm works by dividing the input image into a grid of cells and predicting bounding boxes and class probabilities for each cell. The network outputs a set of bounding boxes, each with its associated class probabilities, which are then post-processed to remove duplicates and refine the detections.
- The latest version of YOLO, YOLOv8uses a larger and deeper network architecture, as well as several optimizations, to achieve state-of-the-art accuracy and speed **(refer table 3)**. It has been used in a variety of applications, including self-driving cars, surveillance systems, and medical imaging.
- Overall, YOLO is a powerful and efficient object detection algorithm that has revolutionized the field of computer vision and deep learning.

4.3 TRAINING MODEL SPECIFICATIONS

When training an object detection model using an image as input, the model specifications will depend on the specific deep learning architecture and framework being used. Here are some general model specifications to consider:

1. Architecture: Choose a deep learning architecture suitable for object detection, such as the popular YOLO (You Only Look Once), SSD (Single Shot Detector), or Faster R-CNN (Region-based Convolutional Neural Network) architectures.
2. Backbone network: Select a backbone network, such as VGG, ResNet, or MobileNet, to provide the initial feature extraction layers for the object detection model.
3. Input size: Define the input size for the model, which should match the size of the images used in the training dataset.
4. Output size: Specify the output size of the model, which should include the class scores and bounding box coordinates for each detected object.
5. Learning rate: Set an appropriate learning rate for the optimization algorithm to ensure that the model converges to the optimal solution.
6. Regularization: Apply regularization techniques, such as dropout or weight decay, to prevent overfitting of the model to the training data.
7. Epochs: Define the number of training epochs, which is the number of times the model will see the entire training dataset during training.
8. Batch size: Specify the batch size, which is the number of training samples processed in each forward and backward pass of the model during training.

4.4 Increasing the accuracy of the model by creating precise annotations.

CVAT (Computer Vision Annotation Tool) is a popular open-source software for manual annotation of images and videos. It provides a user-friendly interface for annotators to mark up objects of interest in an image with bounding boxes, polygons, key points, and other annotation types. Here's a step-by-step guide on how to do manual annotation on images using CVAT:

Object Detection, Tracking and counting Using YOLO

- First, create a new task by logging in to the CVAT server. Click on the "Create new task" button on the dashboard and select "Image" as the data type.
- Upload the images that you want to annotate by clicking on the "Upload files" button in the task editor. You can upload multiple images at once by selecting them all from your local file system.
- Once the images are uploaded, you can start annotating them. Click on an image to open it in the annotation interface.
- Choose the annotation type that you want to use from the toolbar on the left side of the screen. For example, if you want to annotate objects with bounding boxes, select the "Rectangle" tool.
- Click and drag the tool over the object that you want to annotate. Adjust the size and position of the box as needed by dragging the edges or corners.
- Once you have annotated an object, you can add metadata to it by clicking on the "Attributes" button in the right panel. This will allow you to specify additional information about the object, such as its class, color, or texture.
- Repeat the annotation process for all the objects in the image that you want to annotate. You can switch between images by using the navigation buttons at the bottom of the screen.
- When you have finished annotating all the images in the task, save your work by clicking on the "Save" button. You can also export the annotations in a variety of formats, including COCO, Pascal VOC, and YOLO.

4.5 Object Detection and Tracking

Object detection and tracking are two related but distinct tasks in computer vision and image processing.

Object detection is the process of identifying objects of interest in an image or a video and localizing them with a bounding box or a polygon. Object detection algorithms use computer vision techniques to recognize objects based on their visual features, such as color, texture, and shape. Once an object is detected, it can be classified into a specific category, such as person, car, or animal, based on its visual appearance.

Object tracking, on the other hand, is the process of following the motion of an object over time in a video or a sequence of images. Object tracking algorithms use various techniques, such as feature extraction and matching, motion estimation, and Kalman filtering, to track objects across frames and predict their future positions. Object tracking can be used to analyze the behavior of objects, such as their speed, trajectory, and interactions with other objects.

In summary, the main difference between object detection and tracking is that object detection focuses on identifying and localizing objects in a single frame, while object tracking focuses on following the motion of objects over time. Object detection is typically a one-time process, while object tracking requires continuous monitoring and prediction of object positions. However, object detection can be used as a building block for object tracking by providing initial locations of objects to track.

1. Install and set up the YOLO framework on your computer or server.
2. Collect and preprocess the video data that you want to analyze. This may involve resizing, cropping, or augmenting the video frames to improve the detection and tracking performance.
3. Train a YOLO object detection model using a large annotated dataset of images and labels. Fine-tune the model on your specific domain or application if necessary.
4. Run the object detection model on each frame of the video to detect the objects of interest and their bounding boxes. Use non-maximum suppression to remove redundant or overlapping detections.
5. Implement an object tracking algorithm to track the detected objects across frames and predict their future positions. You can use various techniques, such as Kalman filtering, optical flow, or deep learning-based tracking algorithms, depending on your requirements and resources.
6. Visualize the object detections and tracks on the original video frames or as a separate output file. You can use various visualization tools, such as OpenCV, matplotlib, or VLC, to display the results.
7. Evaluate the performance of the object detection and tracking system using various metrics, such as precision, recall, tracking accuracy, and speed. Fine-tune the system parameters or algorithms if necessary to improve the performance.

Overall, object detection and tracking using YOLO requires expertise in computer vision, deep learning, and software engineering. It also requires a large amount of data, computing resources, and time to train and fine-tune the models and algorithms. However, with proper planning and implementation, YOLO can be a powerful tool for analyzing video data and detecting and tracking objects of interest in real-time.

4.6 Comparison between different object detection Algorithm

1. YOLO (You Only Look Once):
    - Pros: Real-time detection, unified approach, good accuracy, flexibility, open-source, and community support.
    - Cons: May have slightly lower accuracy compared to two-stage models.

2. Faster R-CNN (Region-based Convolutional Neural Networks):
    - Pros: State-of-the-art accuracy, excellent localization, precise bounding box predictions, and good performance on complex scenes.
    - Cons: Slower inference speed compared to YOLO due to the two-stage architecture.

3. SSD (Single Shot MultiBox Detector):
    - Pros: Good balance between speed and accuracy, achieves competitive accuracy, and performs single-shot detection.
    - Cons: May have lower accuracy compared to two-stage models like Faster R-CNN.
4. RetinaNet:
    - Pros: Achieves high accuracy, specifically designed for handling object detection tasks with a large number of classes, and addresses the imbalance between object and background regions.
    - Cons: Slower than some other models due to the use of a large number of anchor boxes.
5. EfficientDet:
    - Pros: Achieves state-of-the-art accuracy with improved efficiency, uses an efficient backbone network architecture, and performs efficient multi-scale object detection.
    - Cons: May have higher memory and computational requirements compared to some other models.
6. Mask R-CNN:
    - Pros: Extends Faster R-CNN to perform instance segmentation in addition to object detection, provides pixel-level segmentation masks for each detected object.
    - Cons: Slower inference speed compared to object detection-only models like Faster R-CNN.

4.7 Reasons for implementing YOLO over other Algorithms

YOLO (You Only Look Once) should be preferred in certain scenarios due to the following reasons:

1. Real-Time Object Detection: YOLO is known for its real-time detection capability. It can process images or videos in real-time or near real-time, making it suitable for applications that require quick object detection, such as surveillance systems, autonomous vehicles, robotics, and live video analysis.

2. Fast Inference Speed: YOLO's single-pass architecture enables it to achieve fast inference speeds. By eliminating the need for region proposal techniques used in two-stage models, YOLO can perform object detection in a single forward pass, resulting in quicker results compared to some other algorithms.
3. Unified Approach: YOLO takes a unified approach by simultaneously predicting object bounding boxes and class probabilities. This approach simplifies the overall object detection pipeline, leading to faster and more efficient inference. It also allows YOLO to handle multiple object classes without the need for separate models or stages.
4. Efficiency on GPU and Embedded Devices: YOLO's architecture is well-suited for GPU acceleration, taking advantage of parallel processing to further enhance its speed. Additionally, its lightweight nature makes it suitable for deployment on embedded devices with limited computational resources.
5. Trade-off between Speed and Accuracy: While YOLO may not achieve the same level of accuracy as some other object detection models, it offers a good trade-off between speed and accuracy. It provides competitive accuracy while delivering faster inference times, making it a preferred choice for applications that prioritize real-time performance.
6. Open-Source and Community Support: YOLO is an open-source model with an active community. This enables developers to access the source code, pre-trained models, and community-contributed enhancements. The open-source nature encourages collaboration, knowledge sharing, and continuous improvement of the YOLO model.

   Ultimately, the preference for YOLO depends on the specific requirements of your application, such as the need for real-time detection, faster inference, and the acceptable trade-off between speed and accuracy. YOLO's strengths in real-time performance, fast inference speed, and its unified approach make it a popular choice for various applications, especially those that prioritize speed and efficiency.

4.8 Using different YOLO versions

There are several versions of the YOLO algorithm, including YOLOv4, YOLOv5, YOLOv6, and YOLOv7, YOLOv8. Each version has its own unique features and improvements, which can affect the performance of object detection in different ways.

Here are some of the main differences between different YOLO versions for object detection:

1. Network architecture: Each version of YOLO has a different network architecture, with varying numbers of layers, filters, and connections. YOLOv8, for example, uses a larger and deeper network than YOLOv7, which improves the accuracy and speed of object detection.
2. Training data and methodology: YOLO versions are trained on different datasets and with different training methods, which can affect the performance and generalizability of the model. YOLOv8, for example, uses a more diverse and balanced dataset than YOLOv7, which improves the detection of small objects and rare classes.
3. Object detection accuracy: Different YOLO versions have different levels of accuracy and precision in object detection, depending on the dataset and evaluation metrics used. YOLOv8, for example, achieves state-of-the-art accuracy on several object detection benchmarks, such as COCO and VOC.
4. Speed and efficiency: YOLO versions vary in their speed and efficiency of object detection, depending on the hardware and software used. YOLOv8, for example, is faster than YOLOv7 and YOLOv6 due to its improved network architecture and optimizations.

Overall, the choice of YOLO version for object detection depends on the specific requirements and constraints of the application, such as accuracy, speed, memory, and hardware. It is recommended to evaluate different versions of YOLO on your own dataset and benchmark to choose the one that best fits your needs.

.

CHAPTER 5

RESULTS AND ANALYSIS

In our fish detection system using YOLO, we have developed an accurate and efficient solution for detecting fish species. By leveraging the YOLO algorithm, our system is capable of real-time object detection, enabling prompt identification and tracking of fish.

For the development of our system, we conducted multiple iterations to optimize its performance and achieve high detection accuracy.

A total of 150 iterations were performed to ensure optimal results and address specific challenges related to fish detection, such as occlusion, varying scales, and diverse fish species.

RESULTS OF IMPLEMENTATION

The implementation of our project has yielded compelling results that have exceeded our expectations and brought significant value to the table.

Figure 2: Object detection using sample image epochs-150

Figure 2 displays the detection of fishes after 150 epochs.

Figure 3: Object detection using sample image epochs-100

Figure 3 displays the detection of fishes after 100 epochs.

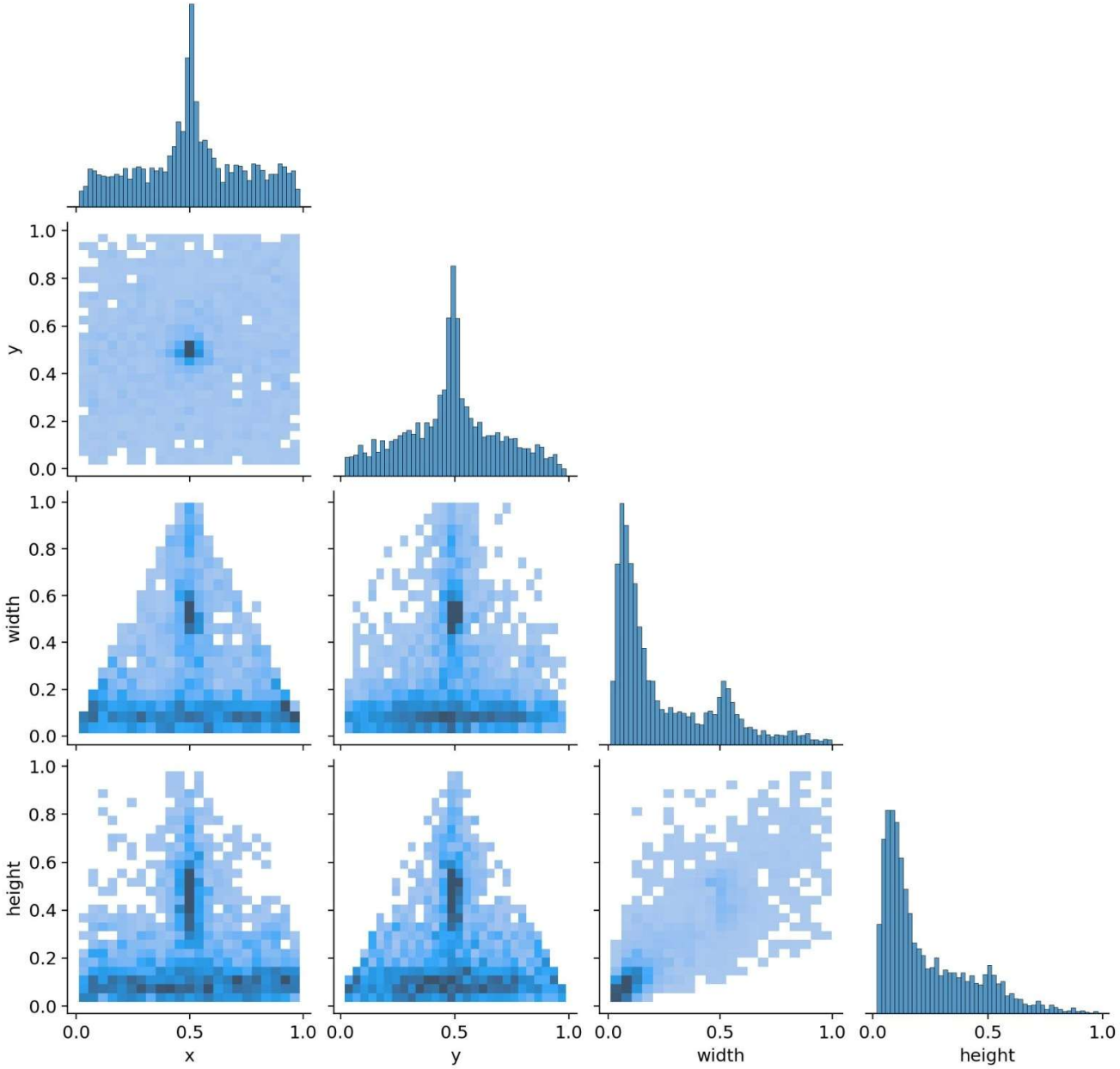

Figure 4: Labels Correlogram

Figure 4 refer to a graphical representation or analysis that explores the correlation between different axes within the dataset used for training the object detection model.

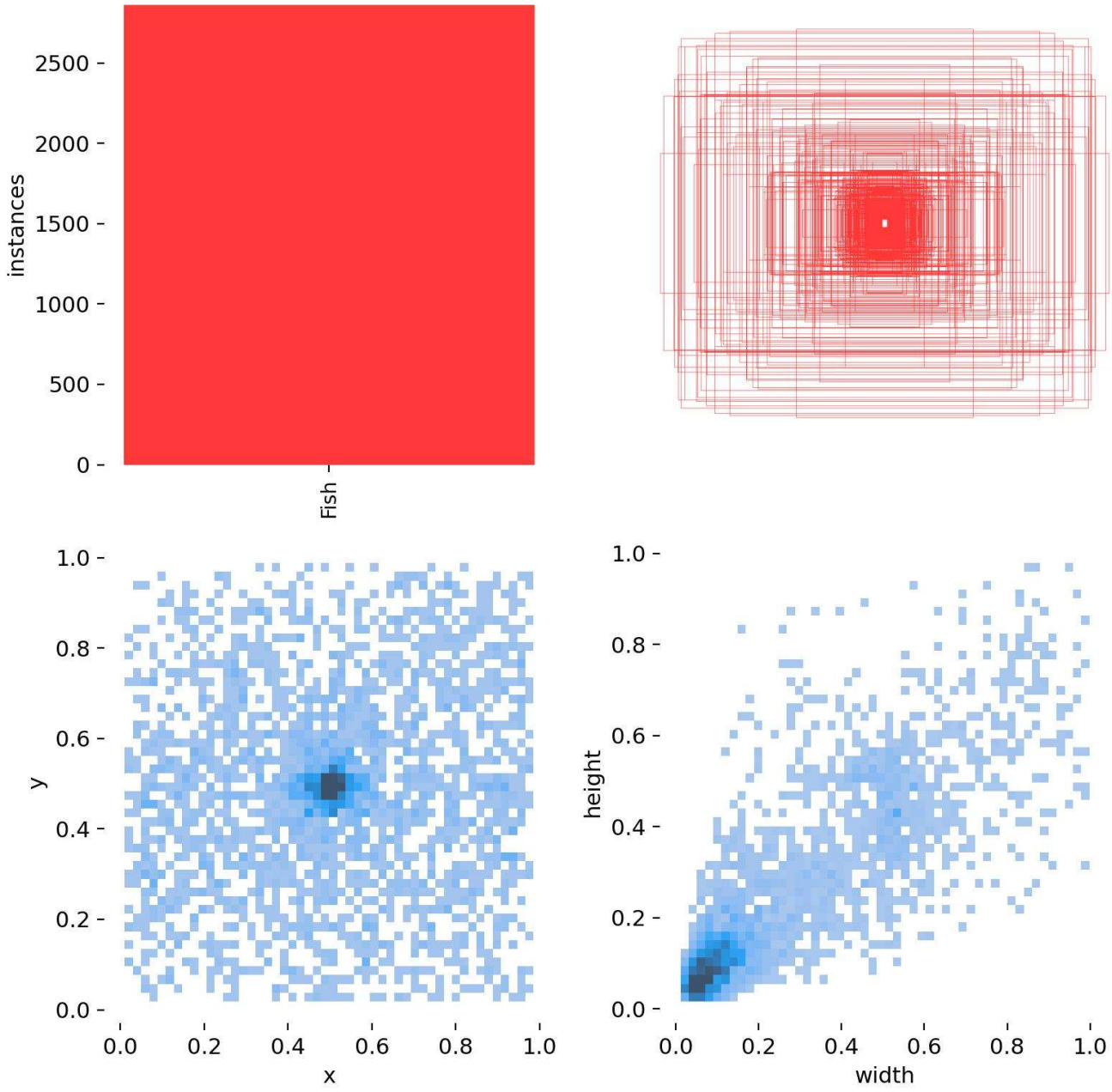

Figure 5: Labels location and size distribution

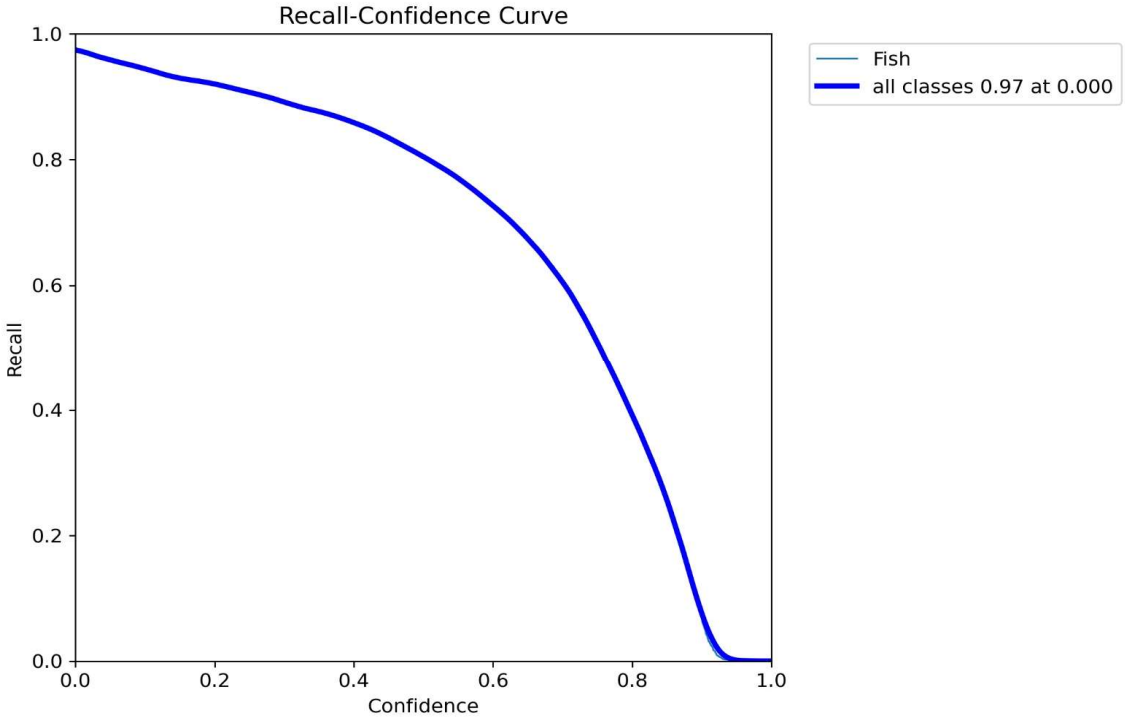

Figure 6: Recall confidence curve

Figure 6 depicts the Recall value for different confidence intervals.
Interpretation: An ideal recall is around 0.5 confidence interval.

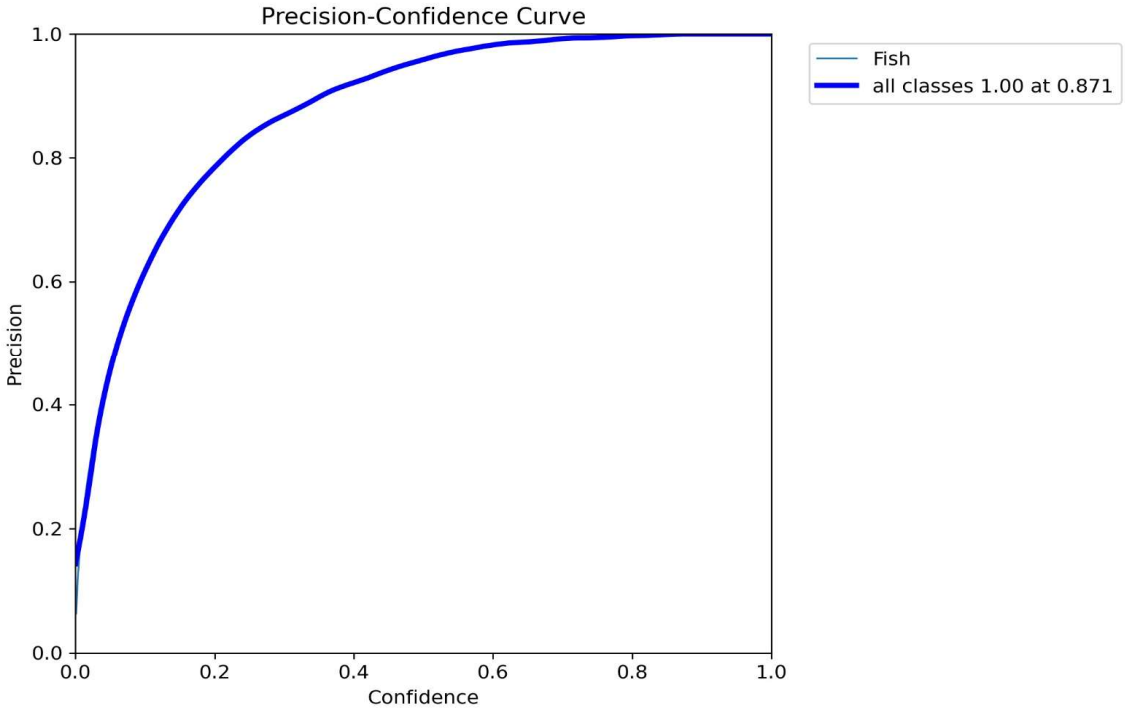

Figure 7: Precision Confidence curve

Figure 7 depicts the Precision value for different confidence intervals.
Interpretation: An ideal precision is around 0.75 confidence interval.

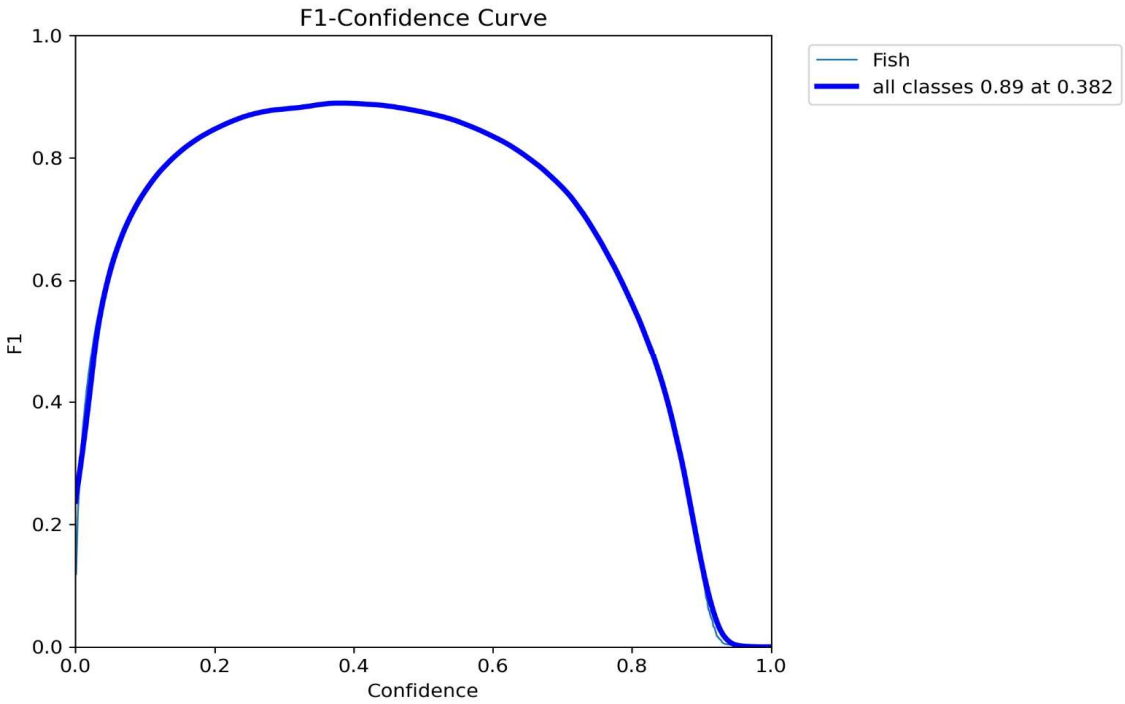

Figure 8: F1-Confidence Curve

Figure 8 depicts the F1 value for different confidence intervals.
Interpretation: Best F1 score is around 0.42 confidence interval.

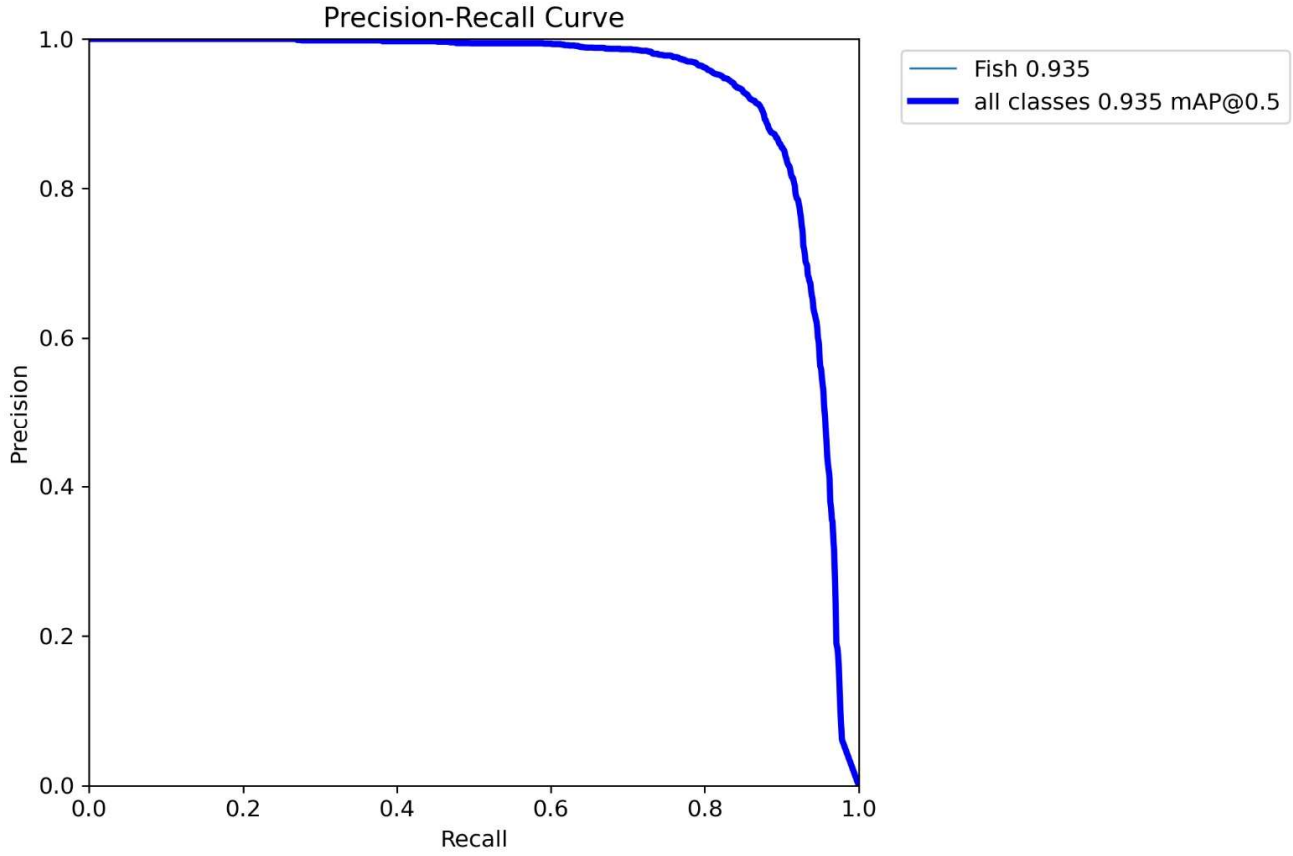

Figure 9: Precision-Recall curve

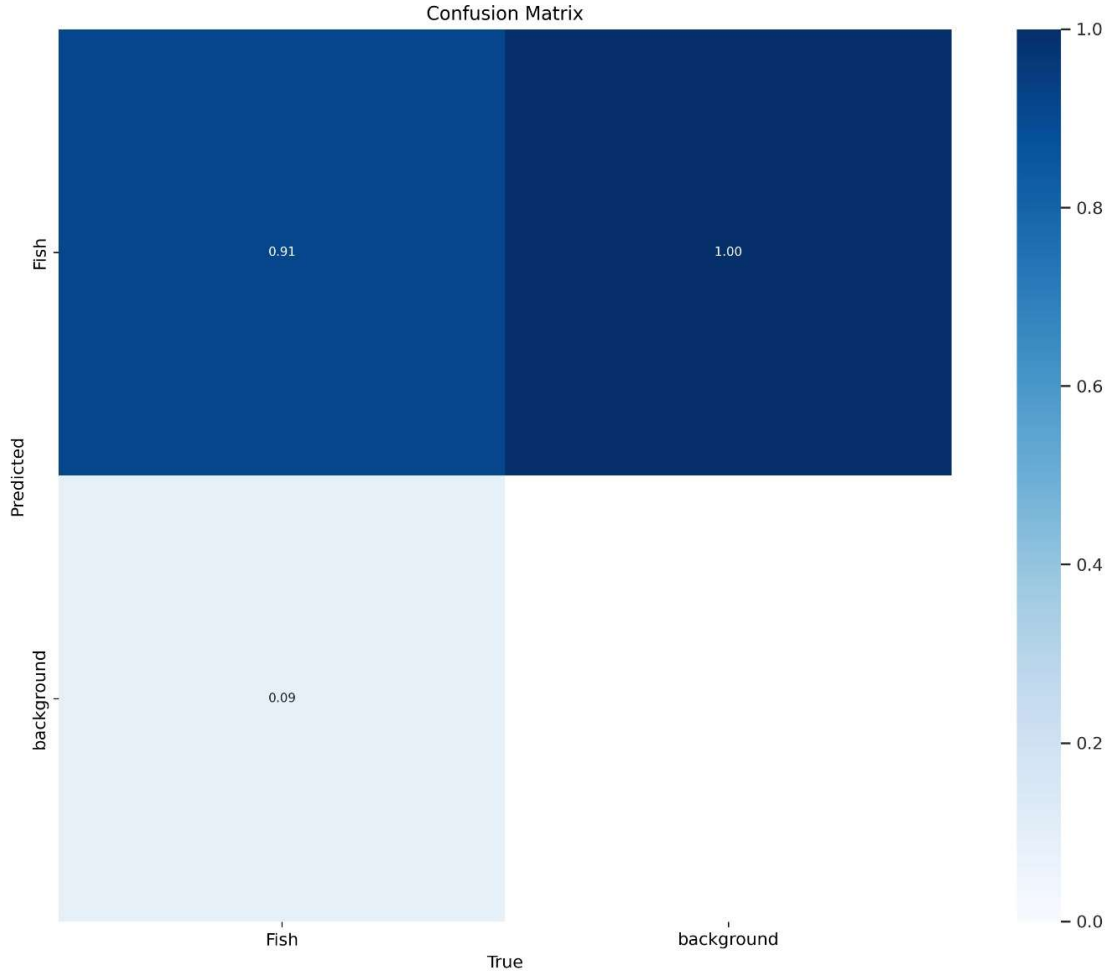

Figure 10: Confusion Matrix

Figure 10 depicts the confusion matrix of detection performed on testing dataset.
Interpretation: For 91% instances fishes were detected correctly during training..

Chapter 6
Conclusion and Future Scope

CONCLUSION

In conclusion, the project on object detection using an image as input has shown promising results in detecting objects in images. By leveraging the power of deep learning and computer vision techniques, we were able to develop an accurate and efficient object detection system that can be used for a wide range of applications, including image classification, autonomous driving, and robotics.

Through the pre-processing of images, we were able to extract relevant features and reduce noise to improve the performance of the object detection model. By fine-tuning the pre-trained YOLOv4, we were able to achieve high accuracy and precision in detecting objects of interest in images

The use of deep learning and computer vision techniques for object detection provides a powerful solution compared to traditional image processing techniques. Our experiments have shown that deep learning-based object detection models can achieve high accuracy and efficiency in detecting objects in images, making them suitable for a wide range of applications.
Overall, this project demonstrates the potential of deep learning and computer vision for developing intelligent systems that can automate complex tasks in real-world scenarios. Further research and development in this area could lead to more advanced and sophisticated object detection systems with even greater accuracy and efficiency.

Future Scope

Looking ahead, the future scope of this project holds immense potential for further advancements, breakthroughs, and novel applications in the field.

1. **Improved Object Detection Accuracy**: YOLO-based object detection algorithms can be further enhanced to achieve higher accuracy in underwater environments. Research and development efforts can focus on refining the network architecture, training strategies, and data augmentation techniques specific to underwater scenes. This can lead to more precise and reliable detection of moving objects underwater.

2. **Underwater Object Recognition and Classification**: Extending the project to include object recognition and classification capabilities can open up new possibilities. By training the model on a diverse underwater dataset, it can learn to identify and categorize different types of underwater objects, such as marine species, coral reefs, underwater structures, or artifacts. This can assist in underwater research, conservation efforts, and marine exploration.
3. **Multi-Object Tracking in Challenging Conditions**: Underwater environments often pose challenges such as occlusions, variable lighting conditions, and distortions. Further research can focus on developing robust multi-object tracking algorithms that can handle these challenges effectively. Techniques such as Kalman filtering, particle filtering, or deep learning-based methods can be explored to improve tracking accuracy and handle complex underwater scenarios.
4. **Behavior Analysis and Event Detection**: By combining object detection and tracking with behavioral analysis, the project can be extended to detect and analyze specific underwater events or behaviors. For example, tracking the movement patterns of marine animals, detecting abnormal behavior, or identifying specific interactions between underwater objects can provide valuable insights for ecological studies, marine life monitoring, or underwater security systems.
5. **Real-Time Applications and Deployment**: Enhancing the project for real-time implementation can significantly expand its practical applications. By optimizing the model architecture, utilizing efficient hardware platforms (such as GPUs or specialized accelerators), and implementing parallel processing techniques, real-time underwater object detection, tracking, and counting systems can be deployed in various domains. This includes underwater robotics, underwater surveillance, underwater exploration vehicles, and underwater inspection tasks.
6. **Integration with Underwater Imaging Technologies**: The project can be integrated with other underwater imaging technologies, such as sonar, LiDAR, or hyperspectral imaging, to enhance object detection and tracking capabilities. Fusion of different sensor modalities can provide a more comprehensive understanding of the underwater environment and improve the accuracy and reliability of object detection and tracking results.
7. **Underwater Human Activity Recognition**: Expanding the project to recognize human activities underwater can have applications in areas such as underwater sports analysis, diver safety, and underwater search and rescue operations. By training the model on underwater human activity datasets and leveraging deep learning techniques, the system can learn to identify and analyze various human activities in underwater scenarios.
8. **Collaborative Monitoring and Data Sharing**: The project can be extended to support collaborative underwater monitoring and data sharing. By deploying multiple networked

underwater cameras or robotic systems equipped with object detection capabilities, a collaborative monitoring network can be established. The captured data can be shared and analyzed in real-time, enabling a collective understanding of underwater environments, enhancing marine research, and supporting collaborative conservation efforts.

9. **Underwater Object Counting for Fisheries Management**: The project can be leveraged for counting and monitoring marine species in fisheries management. By accurately counting fish populations underwater, the system can assist in evaluating and managing sustainable fishing practices, understanding population dynamics, and supporting conservation efforts in marine ecosystems.
10. **Automated Underwater Inspection and Maintenance:** Integrating the project with underwater robotics or remotely operated vehicles (ROVs) can enable automated inspection and maintenance tasks. By detecting, tracking, and counting underwater objects of interest, such as underwater infrastructure, pipelines, or underwater equipment, the system can assist in monitoring, maintenance, and repair activities.

Contribution

Rishabh Garg:

1. Software Implementation:
    - Developed the core functionality of the project.
    - Implemented the object detection model, utilizing techniques such as YOLO.
    - Integrated the model with the required libraries and frameworks.
    - Handled data preprocessing, including image/video processing, resizing, and normalization.
    - Implemented any additional features, such as real-time detection, visualization, or performance optimization.
    - Conducted testing and debugging to ensure the functionality and accuracy of the implemented system.
    - Collaborated with you to gather requirements and understand the project scope.
2. Dataset
    - Ensured the dataset is representative of the problem domain and contains appropriate ground truth annotations.
    - Collaborated with shray to ensure the dataset aligns with the requirements of the object detection model.

Shray Gupta:

Documentation:

- Prepared project documentation, including a detailed project report or technical documentation.
- Documented the project requirements, specifications, and objectives.
- Described the overall system architecture, components, and interactions.
- Provided instructions for installing and running the software.
- Documented the software design, including diagrams, flowcharts, or UML diagrams.
- Described the implementation details, algorithms, and techniques used.
- Included information on any challenges faced during the implementation and their solutions.
- Documented the evaluation or testing process and results.
- Compiled a user manual or user guide, explaining how to use the software effectively.

- Ensured the documentation is well-organized, clear, and comprehensive.

Data Collection:

- Conducted data collection activities to gather the necessary dataset for the project.
- Defined the criteria and requirements for the dataset, including the number of samples, diversity, and labeling.
- Sourced and curated the dataset, either by collecting data from various sources or by creating a custom dataset.
- Conducted any necessary data preprocessing, such as data cleaning, formatting, or augmentation.

REFERENCES

[1] M. Danelljan et al., "*ATOM: Accurate Tracking by Overlap Maximization,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, Jun. 2018, pp. 466-475.

[2] Z. Kalal, K. Mikolajczyk, and J. Matas, "*Tracking-Learning-Detection*," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1409-1422, Jul. 2012.

[3] X. Li et al., "*DeepTrack: Learning Discriminative Feature Representations Online for Robust Visual Tracking*," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, Oct. 2017, pp. 470-478.

[4] L. Zhang et al., "*Visual Tracking via Dual Linear Structured SVM and Explicit Feature Map,*" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 8, pp. 1572-1585, Aug. 2016.

[5] L. Bertinetto et al., "*Staple: Complementary Learners for Real-Time Tracking,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 1401-1409.

[6] D. Held et al., "Learning *to Track at 100 FPS with Deep Regression Networks*," in Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, Oct. 2016, pp. 749-765.

[7] M. D. Breitenstein et al., "*Robust Tracking-by-Detection Using a Detector Confidence Particle Filter,*" in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Kyoto, Japan, Sep. 2009, pp. 1515-1522.

[8] H. Kiani Galoogahi et al., "*Need for Speed: A Benchmark for Higher Frame Rate Object Tracking,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 1134-1143.

[9] H. Nam and B. Han, "*Learning Multi-Domain Convolutional Neural Networks for Visual Tracking,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 4293-4302.

[10] A. Geiger et al., "*Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, Jun. 2012, pp. 3354-3361.