National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Ishita Kundu
Student ID: x22242091

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ishita Kundu
Student ID:	x22242091
Programme:	MSc in Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Abid Yaqoob
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	690
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ishita Kundu
Date:	11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ishita Kundu
x22242091

1 Introduction

Environmental pollution, particularly from debris like plastic, glass, rubber, and metal, poses a significant threat to marine life and ecosystems. Traditional methods of underwater object detection, such as satellite sensors and sonar techniques, have limitations in generating high-resolution images. Recent advancements in technologies like UAVs and AUVs have improved image-capturing processes, enabling better detection of marine debris. However, challenges in achieving high accuracy in real-time detection and classification persist. This research aims to address these challenges by integrating a Multi-Level Hybrid Convolutional Neural Network (MLH-CNN) with YOLOv8. The Seaclear Marine Debris Detection & Segmentation Dataset with 40 different marine object categories is used here for training. This study also proposes a comparative analysis between YOLOv5, YOLOv8, and YOLOv8 MLH-CNN models to find out the most efficient model.

2 System Configuration

Software Component	**Version**
Operating System	Windows 11 23H2
Python	3.11.5
TensorFlow	2.15.0

Table 1: Software Configuration

Hardware Component	**Specification**
Processor	Intel Core i5-8265U
RAM	8 GB DDR4
GPU	Intel UHD 620

Table 2: Hardware Configuration

3 Dataset Source

The datasets used in this project are sourced from the 4TU.ResearchData website. Below are the details of the dataset and the source websites. SeaClear Marine Debris Dataset is an open data repository for science, engineering, and design. This dataset can be accessed

from the website Figure1. It contains 8610 images annotated for object detection and 40 object categories.

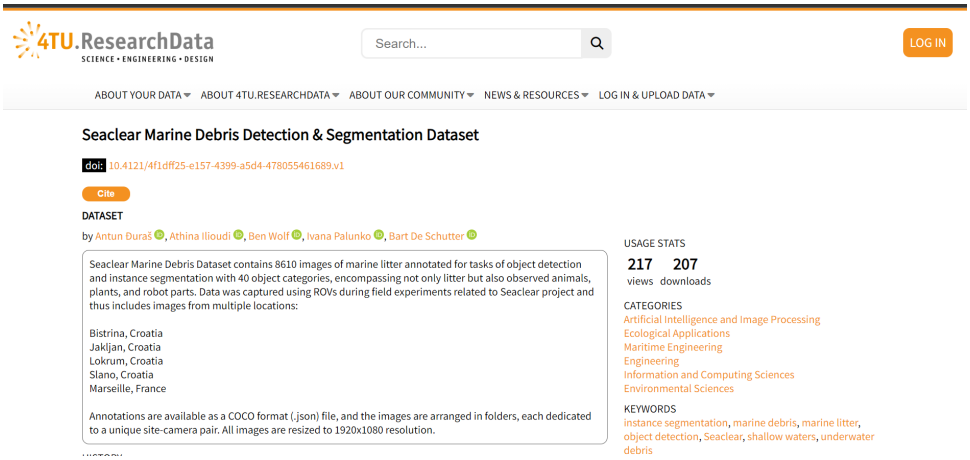

Figure 1: 4TU.ResearchData Website

4 Project Implementation

This section offers a detailed, step-by-step process of the project implementation, accompanied by code screenshots for clarity.

4.1 Importing Packages and Libraries

Initially, necessary packages and libraries are imported into Jupyter Notebook as visible in Figure 2. The project utilized pandas and numpy for data processing, enabling efficient handling of data. For data visualization, matplotlib and seaborn were employed. Additionally, scikit-learn was used for data preparation and model evaluation tasks. The project also imported yaml for managing configuration files.

```
import torch
from IPython.display import Image  # for displaying images
import os
import random
import shutil
from sklearn.model_selection import train_test_split
import xml.etree.ElementTree as ET
from xml.dom import minidom
# from tqdm import tqdm
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import json
import matplotlib.image as mpimg
import re
import yaml
```

Figure 2: Packages and libraries

4.2 Data Loading

The dataset was downloaded in .zip format. After fetching data, it is loaded into the directory as shown in Figures 3 and 4. After loading the data sample images are plotted in Figure 5.

```
# Directory of images
image_dir = 'E:\\PERSONAL\\NCI DA Course\\3rd sem\\Research\\Seaclear Marine Debris Detection & Segmentation Dataset_1_all\\Seaclear Marine Debris Datase
```

Figure 3: Data Loading

```
# Read images
for filename in os.listdir(image_dir):
    if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff')):
        image_path = os.path.join(image_dir, filename)
        image = cv2.imread(image_path)
        if image is not None:
            resized_image = resize_image(image)
            images.append(resized_image)

# Check how many images were loaded
print(f"Loaded {len(images)} images from {image_dir}")
```

Figure 4: Read Images

```
def display_images_for_categories(category_names, num_images_per_category=1):
    fig, axes = plt.subplots(2, 3, figsize=(10, 5))
    axes = axes.flatten()

    for ax, category_name in zip(axes, category_names):
        # Find the category ID from the categories list
        category_id = next((cat['id'] for cat in categories if cat['name'] == category_name), None)
        if category_id is None:
            print(f"No category found for {category_name}")
            ax.imshow(np.zeros((10, 10, 3), dtype=int))  # Show blank image
            ax.set_title("Category Not Found")
            ax.axis('off')
            continue

        # Find images for this category
        image_ids = [ann['image_id'] for ann in anns if ann['category_id'] == category_id]
        image_files = [img for img in imgs if img['id'] in image_ids]

        if image_files:
            img_path = os.path.join('E:\\PERSONAL\\NCI DA Course\\3rd sem\\Research\\Seaclear Marine Debris Detection & Segmentation Dataset_1_all\\Seac
            img = mpimg.imread(img_path)
            ax.imshow(img)
        else:
            ax.imshow(np.zeros((10, 10, 3), dtype=int))
            ax.set_title("No Images Available")

        ax.set_title(category_name)
        ax.axis('off')

    plt.subplots_adjust(hspace=0.1)
    plt.tight_layout()
    plt.show()

# List of category names to display
category_names_to_display = ['bottle_plastic', 'rov_cable', 'bag_plastic', 'rope_fiber', 'tube_cement', 'can_metal']
display_images_for_categories(category_names_to_display)
```

Figure 5: Sample data check

4.3 Data Preparation

4.3.1 Data Resizing

Figure 6 shows the Image resizing process to a resolution of 800x800 pixels.

```python
# Function to resize image
def resize_image(image, max_size=(800, 800)):
    height, width = image.shape[:2]
    scaling_factor = min(max_size[0] / width, max_size[1] / height)
    new_size = (int(width * scaling_factor), int(height * scaling_factor))
    return cv2.resize(image, new_size, interpolation=cv2.INTER_AREA)
```

Figure 6: Data Resizing

4.3.2 Creating bounding box and annotation

Figure 7 shows the process of creating an annotation.txt and bounding box file from a JSON file present in the Dataset.

```python
import json
import os

# Define the path to the JSON file
batch_anns = 'E:\\PERSONAL\\NCI DA Course\\3rd sem\\Research\\Seaclear Marine Debris Detection & Segmentation Dataset_1_all\\Seaclear Marine Debris Da

# Read the JSON file
with open(batch_anns, 'r') as f:
    dataset_batch = json.loads(f.read())

# Create a mapping from category_id to category details
category_id_map = {cat['id']: cat for cat in dataset_batch['categories']}

# Process each annotation
for o in dataset_batch['annotations']:
    fname = dataset_batch['images'][o['image_id']]['file_name']
    fname = fname[:-3] + 'txt'
    fname = fname.replace('atch_', '').replace('/', '_')
    image_w = dataset_batch['images'][o['image_id']]['width']
    image_h = dataset_batch['images'][o['image_id']]['height']

    # bbox info
    bbox = o['bbox']
    top_x, top_y, width, height = bbox

    # Change x and y from top-left to center
    center_x = top_x + (width / 2)
    center_y = top_y + (height / 2)

    # Normalize bbox values
    center_x /= image_w
    center_y /= image_h
    width /= image_w
    height /= image_h

    supercat_name = category_id_map[o['category_id']]['supercategory']
    cat_idx = ordered_supercats.index(supercat_name)

    bbox_line = '{} {} {} {} {}\n'.format(cat_idx, center_x, center_y, width, height)

    # Write the annotation files
    os.makedirs('labels', exist_ok=True)
    with open(f'labelsv8/{fname}', 'a') as f:
        f.write(bbox_line)
```

Figure 7: Bounding box and Annotation

4.3.3 Data Splitting

Figure 8 shows the process of organizing the dataset of images and their corresponding annotation files into training, validation, and test sets using an 80-10-10 ratio.

```
# Create directories for train, val, test
os.makedirs('images/train', exist_ok=True)
os.makedirs('images/val', exist_ok=True)
os.makedirs('images/test', exist_ok=True)
os.makedirs('labels/train', exist_ok=True)
os.makedirs('labels/val', exist_ok=True)
os.makedirs('labels/test', exist_ok=True)

# List files in 'images' and 'labels' directories
images = [os.path.join('images', x) for x in os.listdir('images') if os.path.isfile(os.path.join('imagesv8', x))]
annotations = [os.path.join('labels', x) for x in os.listdir('labels') if x.endswith(".txt")]

# Sort lists
images.sort()
annotations.sort()

# Print the number of files found
print(f"Number of images: {len(images)}")
print(f"Number of annotations: {len(annotations)}")

# Split the dataset into train-valid-test splits
train_images, val_images, train_annotations, val_annotations = train_test_split(images, annotations, test_size=0.2, random_state=1)
val_images, test_images, val_annotations, test_annotations = train_test_split(val_images, val_annotations, test_size=0.5, random_state=1)

# Print sizes of splits
print(f"Train images: {len(train_images)}, Train annotations: {len(train_annotations)}")
print(f"Validation images: {len(val_images)}, Validation annotations: {len(val_annotations)}")
print(f"Test images: {len(test_images)}, Test annotations: {len(test_annotations)}")

# Function to move files to respective directories
def move_files(file_list, target_dir):
    for file_path in file_list:
        file_name = os.path.basename(file_path)
        shutil.move(file_path, os.path.join(target_dir, file_name))

# Move images
move_files(train_images, 'images/train')
move_files(val_images, 'images/val')
move_files(test_images, 'images/test')

# Move annotations
move_files(train_annotations, 'labels/train')
move_files(val_annotations, 'labels/val')
move_files(test_annotations, 'labels/test')

print("Files moved to respective directories.")
```

Figure 8: Data Split

4.3.4 YAML Configuration

Figure 9 shows the process of YAML file creation for each model to specify dataset paths, class labels, and training parameters.

```
data = {'train': "C:\\Users\\ishit\\Downloads\\images\\train",
        'val': "C:\\Users\\ishit\\Downloads\\images\\val",
        'test': "C:\\Users\\ishit\\Downloads\\images\\test",
        'nc' : len(class_id_to_name_mapping),
        'names' : class_id_to_name_mapping}

yaml_file_path = 'C:/Users/ishit/ultralytics/yolo_config.yaml'

with open(yaml_file_path, 'w') as f:
    data = yaml.dump(data, f, sort_keys=False)

print(f"YAML configuration file created at {yaml_file_path}")
```

```
YAML configuration file created at C:/Users/ishit/ultralytics/yolo_config.yaml
```

Figure 9: YAML file Configuration

4.4 Data Visualization

4.4.1 Bounding Box

Figure 10 shows the bounding boxes on sample images to verify the accuracy of annotations.

```
class_id_to_name_mapping = ordered_supercats

def plot_bounding_box(image, annotation_list):
    annotations = np.array(annotation_list)
    w, h = image.size

    plotted_image = ImageDraw.Draw(image)

    transformed_annotations = np.copy(annotations)
    transformed_annotations[:,[1,3]] = annotations[:,[1,3]] * w
    transformed_annotations[:,[2,4]] = annotations[:,[2,4]] * h

    transformed_annotations[:,1] = transformed_annotations[:,1] - (transformed_annotations[:,3] / 2)
    transformed_annotations[:,2] = transformed_annotations[:,2] - (transformed_annotations[:,4] / 2)
    transformed_annotations[:,3] = transformed_annotations[:,1] + transformed_annotations[:,3]
    transformed_annotations[:,4] = transformed_annotations[:,2] + transformed_annotations[:,4]

    for ann in transformed_annotations:
        obj_cls, x0, y0, x1, y1 = ann

        ann_class = class_id_to_name_mapping[(int(obj_cls))]
        print(ann_class)

        plotted_image.rectangle(((x0,y0), (x1,y1)))
        plotted_image.text((x0, y0 - 10), ann_class, )

    plt.imshow(np.array(image))
    plt.show()

# Get any random image from the train set
image_file = random.choice(train_imgs)
assert os.path.exists(image_file)

#Get the corresponding image file
annotation_file = image_file.replace("images", "labels")
annotation_file = re.sub('jpg', 'txt', annotation_file, flags=re.IGNORECASE)

with open(annotation_file, "r") as file:
    annotation_list = file.read().split("\n")[:-1]
    annotation_list = [x.split(" ") for x in annotation_list]
    annotation_list = [[float(y) for y in x ] for x in annotation_list]

#Load the image
image = Image.open(image_file)

#Plot the Bounding Box
plot_bounding_box(image, annotation_list)
```

Figure 10: Bounding Box Check

4.4.2 Class Distribution

Figure 11 shows a Hist plot to visualize the distribution of different object classes and sizes in the dataset.

```python
# Mapping from category_id
cat_id_to_index = {cat['id']: idx for idx, cat in enumerate(categories)}

# Count annotations
cat_histogram = np.zeros(nr_cats, dtype=int)
for ann in anns:
    cat_idx = cat_id_to_index[ann['category_id']]
    cat_histogram[cat_idx] += 1

# Initialize the matplotlib
f, ax = plt.subplots(figsize=(15, 15))

# Convert to DataFrame
df = pd.DataFrame({'Categories': cat_names, 'Number of annotations': cat_histogram})
df = df.sort_values('Number of annotations', ascending=False)

# Plot the histogram
sns.set_color_codes("pastel")
sns.set(style="whitegrid")
plot_1 = sns.barplot(x="Number of annotations", y="Categories", data=df, label="Total", color="b")
plt.show()
```

Figure 11: Class Distribution

4.5 YOLOv5 Model

4.5.1 Git clone

Figure 12 shows the process of cloning the latest YOLOv5 repository from GitHub.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ishit>git clone https://github.com/ultralytics/yolov5.git
```

Figure 12: YOLOv5 Git Repository

4.5.2 Training

Figure 13 shows YOLOv5 model training with 20 epochs using image sizes of 215, and batch sizes of 8.

```
!python yolov5/train.py --img 215 --cfg yolov5s.yaml --batch 8 --epochs 20 --data yolov5/yolo_config.yaml --weights yolov5s.pt --name yolo_det_final --pr

WARNING:tensorflow:From C:\Users\ishit\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecat
ed. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

train: weights=yolov5s.pt, cfg=yolov5s.yaml, data=yolov5/yolo_config.yaml, hyp=yolov5\data\hyps\hyp.scratch-low.yaml, epochs=20, batch_size=8, imgsz=21
5, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, evolve_population=yolov5\data\hyps, resume_evolv
e=None, bucket=, cache=None, image_weights=False, device=, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=yolov5/
runs/train, name=yolo_det_final, exist_ok=True, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_r
ank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest, ndjson_console=False, ndjson_file=False
github:  YOLOv5 is out of date by 1 commit. Use 'git pull' or 'git clone https://github.com/ultralytics/yolov5' to update.
YOLOv5  v7.0-347-gd6112173 Python-3.11.5 torch-2.3.1 CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5,
cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=
0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5  runs in Comet
TensorBoard: Start with 'tensorboard --logdir yolov5\runs\train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=40
```

Figure 13: YOLOv5 Model Training

4.5.3 Prediction

Figure 14 shows YOLOv5 model prediction.

```
!python yolov5/detect.py --source images/val/2255.jpg --weights yolov5/runs/train/yolo_det_final/weights/best.pt --img 640 --conf 0.1

detect: weights=['yolov5/runs/train/yolo_det_final/weights/best.pt'], source=images/val/2255.jpg, data=yolov5\data\coco128.yaml, imgsz=[640, 640],
thres=0.1, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_csv=False, save_conf=False, save_crop=False, nosave=False,
s=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=yolov5\runs\detect, name=exp, exist_ok=False, line_thickness=3,
abels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5  v7.0-347-gd6112173 Python-3.11.5 torch-2.3.1 CPU

Fusing layers...
YOLOv5s summary: 157 layers, 7118005 parameters, 0 gradients, 16.1 GFLOPs
image 1/1 C:\Users\ishit\Downloads\images\val\2255.jpg: 384x640 1 animal_shells, 5 wreckage_metals, 6 animal_etcs, 9 animal_sponges, 224.7ms
Speed: 3.3ms pre-process, 224.7ms inference, 15.2ms NMS per image at shape (1, 3, 640, 640)
Results saved to yolov5\runs\detect\exp6
```

Figure 14: YOLOv5 Model Prediction

4.6 YOLOv8 Model

4.6.1 Git clone

Figure 15 shows the process of cloning the latest YOLOv8 repository from GitHub.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ishit>git clone https://github.com/ultralytics/ultralytics.git
```

Figure 15: YOLOv8 Git Repository

4.6.2 Training

Figure 16 shows YOLOv8 model training with 20 epochs and image sizes of 215.

```
from ultralytics import YOLO
model = YOLO('yolov8s.pt')

model.train(data='C:/Users/ishit/ultralytics/yolo_config.yaml', epochs=20, imgsz=215)

New https://pypi.org/project/ultralytics/8.2.75 available  Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.2.68  Python-3.11.5 torch-2.3.1 CPU (Intel Core(TM) i5-8265U 1.60GHz)
engine\trainer: task=detect, mode=train, model=yolov8s.pt, data=C:/Users/ishit/ultralytics/yolo_config.yaml, epochs=20, time=None, patience=100, batch=1
6, imgsz=215, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=train3, exist_ok=False, pretrained=True, optimizer=aut
o, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=
False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None,
iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=Fa
lse, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, sh
ow_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None,
workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cl
s=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, p
erspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=No
ne, tracker=botsort.yaml, save_dir=runs\detect\train3
WARNING:tensorflow:From C:\Users\ishit\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecat
ed. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

Overriding model.yaml nc=80 with nc=40
```

Figure 16: YOLOv8 Model Training

4.6.3 Validation

Figure 17 shows the YOLOv8 model validation.

```
from ultralytics import YOLO

!yolo task=detect mode=val model="C:/Users/ishit/Downloads/runs/detect/train3/weights/best.pt" data='C:/Users/ishit/ultralytics/yolo_config.yaml'
```

```
Ultralytics YOLOv8.2.68 ðŸš€ Python-3.11.5 torch-2.3.1 CPU (Intel Core(TM) i5-8265U 1.60GHz)
Model summary (fused): 168 layers, 11,141,064 parameters, 0 gradients, 28.5 GFLOPs
                   all        861       3238      0.808      0.535      0.599      0.429
           cable_metal         83        126      0.805      0.656      0.768      0.526
            animal_etc         46         51      0.822      0.812       0.85      0.696
         animal_sponge         49         49      0.985          1      0.995      0.968
           bag_plastic        143        225      0.772      0.689      0.749      0.486
         animal_urchin         37         42      0.838      0.738      0.773      0.481
        bottle_plastic         12         12          1       0.33       0.59      0.339
      unknown_instance         13         15      0.649     0.0667      0.116     0.0499
           boot_rubber          2          2          1          0     0.0199    0.00199
             rov_cable        107        130       0.93      0.718       0.78      0.659
     container_plastic         57         99      0.839      0.316      0.425      0.279
           cup_plastic          2          2      0.816        0.5      0.545      0.404
         tarp_plastic        225        261      0.992      0.943       0.96      0.833
       animal_starfish         85         89      0.929      0.878      0.933      0.653
 snack_wrapper_plastic          8          8      0.843      0.875      0.877      0.643
            rope_fiber        195        633      0.804      0.379      0.561      0.241
          rope_plastic          6          6      0.732      0.464      0.514      0.237
    sanitaries_plastic         18         18      0.921        0.5      0.582      0.404
           lid_plastic         84        115      0.676       0.33      0.374      0.231
          animal_fish          47         77      0.775       0.61      0.689       0.45
          tube_cement          27         28      0.771      0.429      0.459      0.242
          branch_wood          80        117      0.808      0.444      0.513      0.298
       furniture_wood          13         14      0.999      0.857       0.91      0.716
container_middle_size_metal          1          1          1          0          0          0
       rov_vehicle_leg         27         27      0.973      0.556      0.748      0.549
                 plant         12        108      0.408      0.102     0.0978      0.031
         animal_shells          4          4      0.624          1      0.995      0.833
        clothing_fiber         15         16      0.941       0.75      0.897      0.752
           cup_ceramic          6          7      0.235      0.143      0.295      0.234
         wreckage_metal        152        261      0.818      0.518      0.625      0.407
           rov_bluerov         15         23      0.491      0.348      0.389      0.264
   snack_wrapper_paper         15         26      0.608      0.538      0.557      0.335
          pipe_plastic        116        129      0.909      0.589      0.742      0.486
           net_plastic         12         12      0.825      0.417      0.478      0.397
      cardboard_paper          2          2          1          0          0          0
          tire_rubber         78         85      0.863      0.906      0.934      0.728
           brick_clay          7          7      0.692      0.714      0.726      0.523
            jar_glass        219        411      0.801      0.684      0.789      0.485
Speed: 0.1ms preprocess, 31.5ms inference, 0.0ms loss, 1.0ms postprocess per image
Results saved to runs\detect\val3
ðŸ’¡ Learn more at https://docs.ultralytics.com/modes/val
```

Figure 17: YOLOv8 Model Validation

4.6.4 Prediction

Figure 18 shows YOLOv8 model prediction.

```
from ultralytics import YOLO

!yolo task=detect mode=predict model="C:/Users/ishit/Downloads/runs/detect/train3/weights/best.pt" conf=0.25 source="C:\\Users\\ishit\\Downloads\\images\
```

```
Ultralytics YOLOv8.2.68 ðŸš€ Python-3.11.5 torch-2.3.1 CPU (Intel Core(TM) i5-8265U 1.60GHz)
Model summary (fused): 168 layers, 11,141,064 parameters, 0 gradients, 28.5 GFLOPs

image 1/1 C:\Users\ishit\Downloads\images\test\1966.jpg: 128x224 2 rope_fibers, 1 tire_rubber, 2 jar_glasss, 105.3ms
Speed: 7.6ms preprocess, 105.3ms inference, 18.1ms postprocess per image at shape (1, 3, 128, 224)
Results saved to runs\detect\predict7
ðŸ’¡ Learn more at https://docs.ultralytics.com/modes/predict
```

Figure 18: YOLOv8 Model Prediction

4.7 YOLOv8 with MLH CNN Model

4.7.1 YOLOv8 Modification

Figure 19 shows the modified backbone in the YOLOv8.yaml file of YOLOv8 by incorporating MLH CNN.

C: > Users > ishit > Downloads > ! yolov8.yaml

```yaml
# Parameters
nc: 80  # Number of classes
scales:  # Model compound scaling constants
  n: [0.33, 0.25, 1024]
  s: [0.33, 0.50, 1024]
  m: [0.67, 0.75, 768]
  l: [1.00, 1.00, 512]
  x: [1.00, 1.25, 512]

# YOLOv8.0n MLH-CNN backbone
backbone:
  # Each Conv layer automatically includes Batch Normalization
  - [-1, 1, Conv, [64, 3, 2, 1]]
  - [-1, 1, Conv, [128, 3, 2, 1]]
  - [-1, 3, BottleneckCSP, [128]]
  - [-1, 1, Conv, [256, 3, 2, 1]]
  - [-1, 6, BottleneckCSP, [256]]
  - [-1, 1, Conv, [512, 3, 2, 1]]
  - [-1, 6, BottleneckCSP, [512]]
  - [-1, 1, Conv, [1024, 3, 2, 1]]
  - [-1, 3, BottleneckCSP, [1024]]
  - [-1, 1, SPP, [1024, [5, 9, 13]]]

```

Figure 19: Modified yaml file

4.7.2 Training

Figure 20 shows YOLOv8 with MLH CNN model training with 20 epochs and image sizes of 215.

```
from ultralytics import YOLO
model = YOLO("C:\\Users\\ishit\\anaconda3\\Lib\\site-packages\\ultralytics\\cfg\\models\\v8\\yolov8.yaml")

model.train(data='C:/Users/ishit/ultralytics/yolo_config.yaml', epochs=20, imgsz=215)
```

```
WARNING  no model scale passed. Assuming scale='n'.
New https://pypi.org/project/ultralytics/8.2.75 available  Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.2.68  Python-3.11.5 torch-2.3.1 CPU (Intel Core(TM) i5-8265U 1.60GHz)
engine\trainer: task=detect, mode=train, model=C:\Users\ishit\anaconda3\Lib\site-packages\ultralytics\cfg\models\v8\yolov8.yaml, data=C:/Users/ishit/ultralytics/yolo_config.yaml, epochs=20, time=None, patience=100, batch=16, imgsz=215, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=train4, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect\train4
WARNING:tensorflow:From C:\Users\ishit\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

Figure 20: YOLOv8 MLH CNN Model Training

4.7.3 Validation

Figure 21 shows the YOLOv8 with MLH CNN model validation.

```
from ultralytics import YOLO

!yolo task=detect mode=val model="C:/Users/ishit/Downloads/runs/detect/train3/weights/best.pt" data='C:/Users/ishit/ultralytics/yolo_config.yaml'
```

Figure 21: YOLOv8 MLH CNN Model Validation

4.7.4 Prediction

Figure 22 shows YOLOv8 with MLH CNN model prediction.

```
from ultralytics import YOLO

!yolo task=detect mode=predict model="C:/Users/ishit/Downloads/runs/detect/train5/weights/best.pt" conf=0.25 source="C:\\Users\\ishit\\Downloads\\images\
```

```
Ultralytics YOLOv8.2.68 ðŸš€ Python-3.11.5 torch-2.3.1 CPU (Intel Core(TM) i5-8265U 1.60GHz)
YOLOv8 summary (fused): 187 layers, 2,982,168 parameters, 0 gradients, 8.0 GFLOPs

image 1/1 C:\Users\ishit\Downloads\images\test\1966.jpg: 128x224 1 rope_fiber, 1 tire_rubber, 3 jar_glasss, 63.8ms
Speed: 1.5ms preprocess, 63.8ms inference, 4.0ms postprocess per image at shape (1, 3, 128, 224)
Results saved to runs\detect\predict6
ðŸ’¡ Learn more at https://docs.ultralytics.com/modes/predict
```

Figure 22: YOLOv8 MLH CNN Model Prediction

4.8 Conclusion

The goal of this research was to develop a model to detect and classify marine objects accurately. This step was achieved by implementing YoLov5, YoLov8, and YoLov8 with MLH-CNN integration. A comparison study of three models concluded that YoLov8 performed the best out of the three models.