Underwater Object Detection

Anish Kundu, Anishesh Mitra, Biswarupayan Dutta, Bodhdipta Roy, Agniva Bera, Subhajit Chakraborty

7th Semester, CSE, KIIT Deemed University

Minor Project

Introduction

Challenges in Underwater Object Detection

- Underwater environments present unique challenges including poor lighting, color distortion, low contrast, and turbidity that significantly degrade image quality.
- Despite covering 70% of Earth's surface, marine ecosystems lack robust automated monitoring systems compared to terrestrial environments.

Objective

- Develop a YOLOv8-based model for real-time underwater creature detection with optimized training configurations.
- Address challenges like limited computational resources and diverse marine species recognition through efficient architecture and data augmentation.

Key Contributions

- CPU-optimized training pipeline for accessibility with limited GPU resources.
- Comprehensive data augmentation strategy to handle underwater image variations.
- Structured framework with automated validation and model checkpointing.

Literary Review

Key Studies in Object Detection

Ren et al. [6]: Introduced Faster R-CNN, which uses a Region Proposal Network (RPN) to generate object proposals directly from feature maps.
It removed the need for slow external proposal methods and made real-time object detection possible with high accuracy. Used anchor boxes, shared convolutional layers, and end-to-end training for better speed and precision.

Wang & Xiao [7]: Improved Faster R-CNN for underwater object detection.
Replaced VGG16 with Res2Net101 for stronger feature extraction.
Used OHEM to balance positive and negative samples, GIoU for better box regression, and Soft-NMS to reduce missed detections. Added multi-scale training and Mosaic augmentation for better accuracy. Achieved mAP@0.5 = 71.7%, about 3.3% higher than the original Faster R-CNN.

Findings

Faster R-CNN made object detection faster and more accurate.The improved version works better in complex underwater environments. Using Res2Net, OHEM, GIoU, Soft-NMS, and multi-scale training increases detection accuracy and robustness.

Description

The system uses Dataset-4 (Underwater) with YOLOv8n pre-trained weights for transfer learning. Images are preprocessed to 416×416 pixels with normalization and augmentation (mosaic, geometric transforms, HSV adjustments) to simulate underwater conditions. YOLOv8's CSPDarknet backbone with FPN extracts multi-scale features for detection. The model employs anchor-free detection heads and multi-class Softmax for species classification, trained using SGD optimizer with early stopping. Performance is evaluated via mAP50, mAP50-95, and validation loss metrics.

System Model

1.Data Acquisition

- Dataset-4 (Underwater): Contains labeled underwater creature images organized in train/val splits.
- YOLOv8n Pre-trained Weights: Transfer learning from COCO dataset for feature extraction baseline.

2. Preprocessing

- Image Resizing: Standardization to 416×416 pixels for consistent input dimensions.
- Normalization: Pixel value scaling for improved model convergence.
- Data Augmentation: Mosaic composition, geometric transforms (rotation ±10°, translation 20% scaling 50%), horizontal/vertical flipping, and HSV color space adjustments.

3. Feature Extraction

- Backbone Network: YOLOv8 CSPDarknet for hierarchical feature extraction.
- Multi-scale Detection: Feature pyramid network (FPN) for detecting objects at various scales.
- Augmentation Features: Enhanced robustness through synthetic variations mimicking underwater conditions.

4. Detection & Classification

- Object Detection: YOLO architecture with anchor-free detection heads.
- Classification: Multi-class Softmax for creature species identification.
- Training Strategy: SGD optimizer with patience-based early stopping and periodic model checkpointing.

5. Evaluation Metrics

- mAP50: Mean Average Precision at IoU threshold 0.5.
- mAP50-95: Mean Average Precision across IoU thresholds 0.5 to 0.95.
- Validation Loss: Monitoring overfitting and model generalization.

Proposed Mechanism

Key Steps

1. Data Preparation:

- Organize dataset (train/val).
- Load or create data.yaml with class info.

2. Model Configuration:

- Use pre-trained YOLOv8n (CPU-optimized).
- Set image size = 416, batch size = 4, epochs = 5.

3. Data Augmentation:

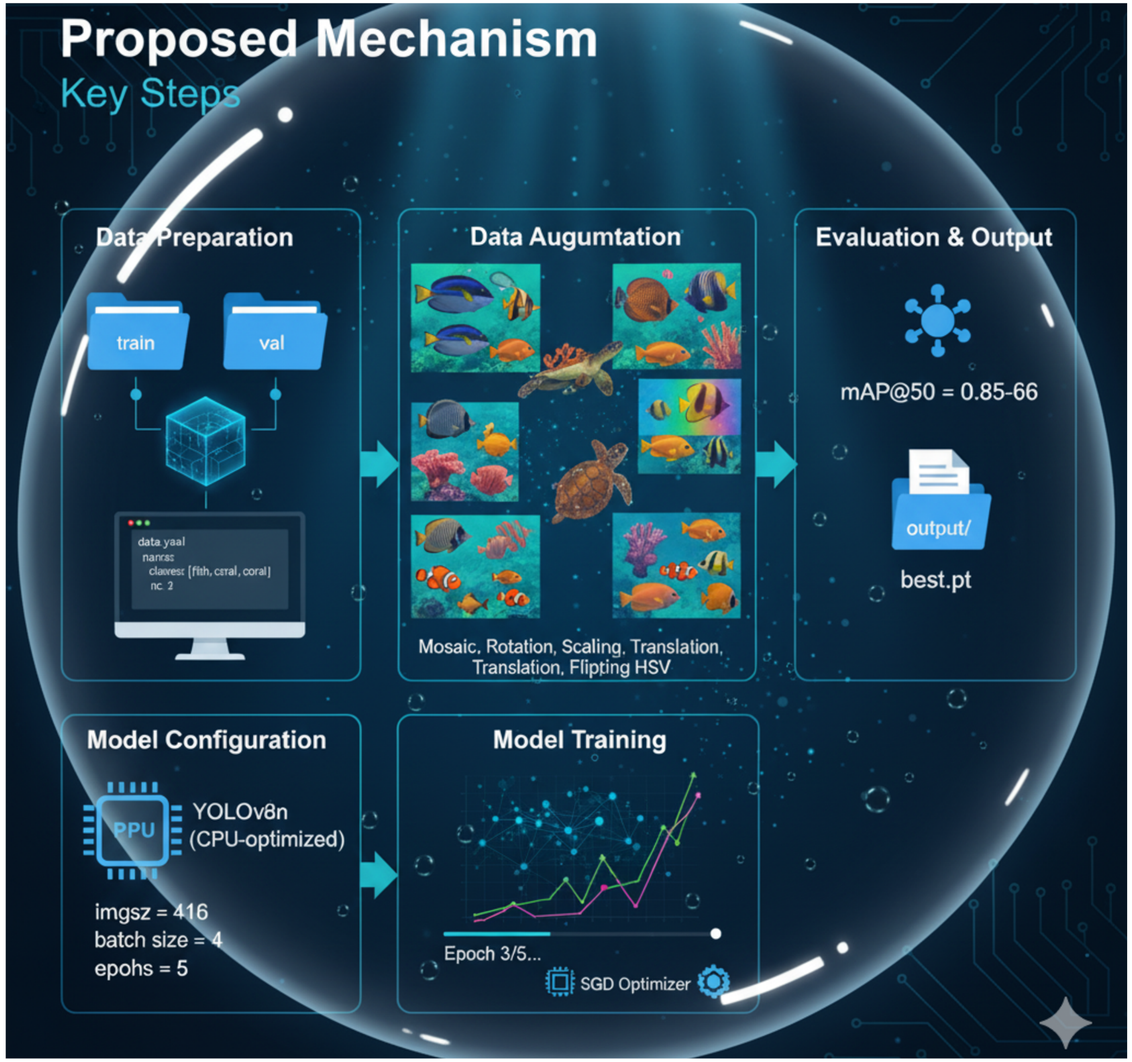
- Apply Mosaic, rotation, scaling, translation, flipping, and HSV adjustments.

4. Model Training:

- Train model using YOLO framework.
- Use SGD optimizer and pretrained weights.
- Validate during training and save checkpoints.

5. Evaluation & Output:

- Compute mAP@50 and mAP@50–95.
- Save best model in output folder.



Workflow

Configuration & Setup:
Load parameters (model, epochs, batch size, augmentations) and check system RAM/device.

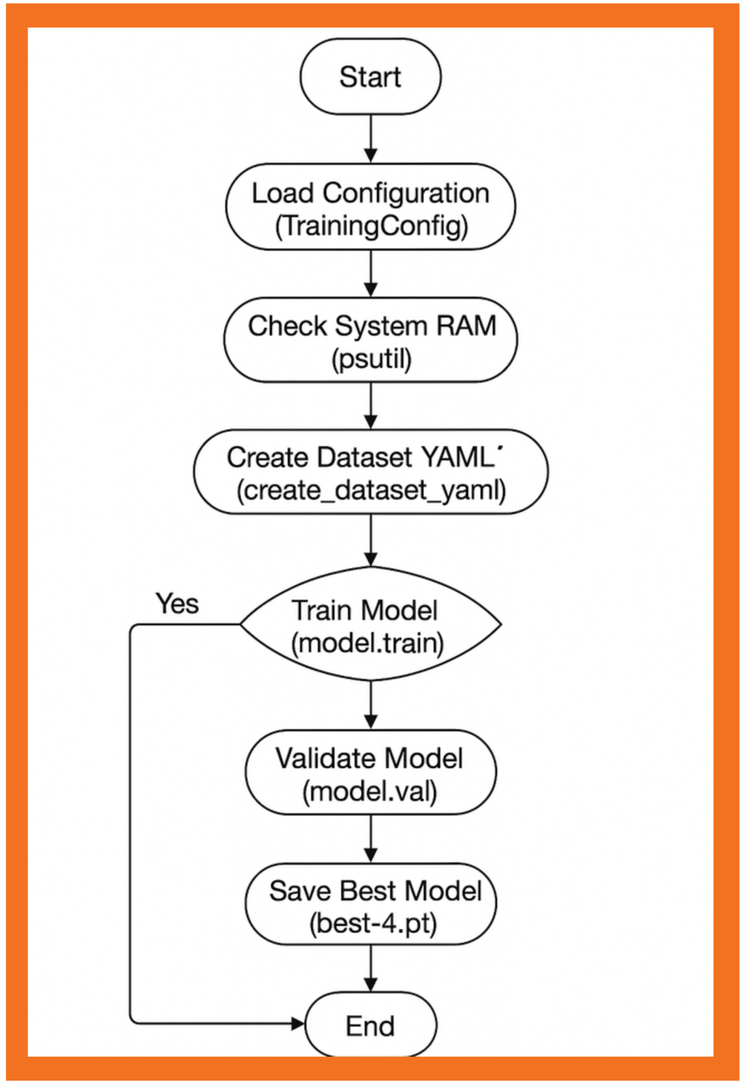
Dataset Preparation:
Find data.yaml, update paths, and create underwater_data.yaml.

Model Initialization:
Load pre-trained YOLOv8n model for CPU.

Training:
Train with augmentations using SGD optimizer; save checkpoints.

Validation:
Evaluate with mAP50 and mAP50-95 metrics.

Output:
Save best weights as best-4.pt — final trained model ready for detection.



Details of Work

• **Tools Used:** Python, Ultralytics YOLOv8, PyTorch, YAML, psutil, pathlib.

Model Architecture

- **Model:** YOLOv8s
- **Training Input:** 640×640 images.
- **Dataset:** Custom underwater dataset (Dataset-4) with train/ and val/ splits.
- **Loss Function:** Combination of objectness, classification, and bounding box regression loss.
- **Optimization:** Mini Batch Stochastic Gradient Descent (BGD).
- **Augmentation:** Mosaic (1.0), Rotation (±10°), Translation (0.2), Scaling (0.5), Horizontal/Vertical Flip (0.5), HSV adjustments.

Performance Metrics

Initial Training (YOLOv8n):

Epochs = 50, Batch = 8, GPU-only.

Metrics after Validation:

• mAP@50 = 87.5 %

• mAP@50–95 = 65%

Optimized Run (Enhanced Dataset):

Improved precision and recall for compound underwater objects (e.g., coral clusters, multiple fish).

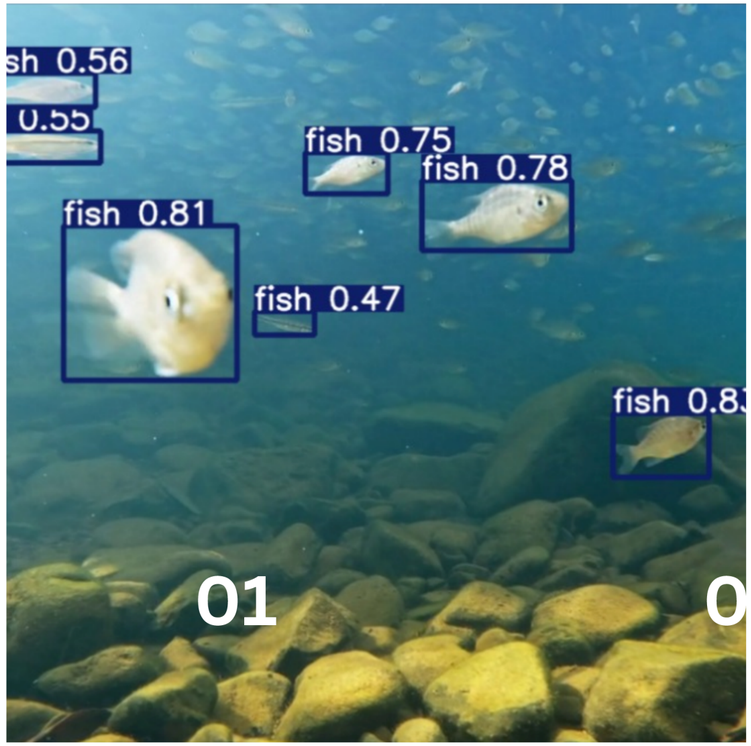
Key Metrics:

• Precision = A %

• Recall = B %

• Accuracy = 87.5 %

Results



Fish Detection confidence

Video Detection with confidence

Future Works
(Improvements & Extensions)

Model Optimization:
- Implement GPU or TPU training to improve speed and accuracy.

Dataset Expansion:
- Add more underwater images with diverse lighting, turbidity, and species.

Real-Time Deployment:
- Integrate the model into underwater drones or live video feeds for real-time detection.

Model Comparison:
- Compare YOLOv8 with other models like Faster R-CNN, EfficientDet, or YOLOv9.

Transfer Learning & Fine-Tuning:
- Use transfer learning with domain-specific pre-trained models for better results.

Post-Processing Improvements:
- Enhance bounding box accuracy using non-max suppression tuning or ensemble methods.

Web or Mobile Application:
- Develop a simple interface to upload underwater images or videos for automatic detection.

Analysis and Conclusion

Analysis and Conclusion

- Developed a YOLOv8-based underwater object detection system with strong performance on Dataset-4.
- Optimized for CPU training using data augmentation, improving accuracy and generalization under varied underwater conditions.

Summary

- Enhances underwater exploration, aiding marine research, environmental monitoring, and object recognition.
- Demonstrates that lightweight YOLO models can perform efficiently even with limited computational resources.

Impact

- Supports real-world applications such as underwater robotics, marine biodiversity tracking, and pollution detection.
- Provides a foundation for future advancements in real-time, high-accuracy underwater detection systems.

THANK YOU!

Anish Kundu 22052797

Anishesh Mitra 22052884

Biswarupayan Dutta 22053152

Bodhdipta Roy 22052976

Agniva Bera 22052964

Subhajit Chakraborty 22052249