**Binary Classification**

The **DEEPFish** dataset is primarily designed for **fish detection and segmentation in underwater environments**, particularly focusing on **deep-sea video footage**. However, whether it contains **species-level annotations** depends on the specific version or subset of the dataset you're referring to.

Here’s a general breakdown:

* **Main focus**: The original DEEPFish dataset is mainly annotated for **object detection and segmentation** tasks — identifying and segmenting **fish as objects** in images, without necessarily labeling their species.
* **Species labels**: As of the original DEEPFish dataset release, **species-level annotations are not included**. The dataset was built to benchmark tasks like fish counting, detection, and segmentation under challenging underwater conditions (e.g., low visibility, cluttered backgrounds).

That said, there may be **extensions** or related datasets that incorporate species information. If you’re specifically looking for **fish datasets with species annotations**, you might want to look into:

* **Fish4Knowledge**
* **LifeCLEF Fish**
* **Global FinPrint**
* **SeaCLEF / BioLifeCLEF challenges**

Would you like help finding a dataset with species labels for training a model or for research?

Here’s a structured analysis of the \*\*6th paper\*\* you referenced (likely \*\*"Multi-Scale Marine Object Detection in Side-Scan Sonar Images Based on BES-YOLO"\*\* from the search results):

---

## \*\*Key Contributions of BES-YOLO\*\*

### \*\*1. Novel Components\*\*

| Component | Description | Technical Advantage |

|-----------------|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|

| \*\*Wavelet Preprocessing\*\* | Uses wavelet transform to denoise SSS images | Reduces speckle noise while preserving edges (\*\*8% improvement in small-target recall\*\*) |

| \*\*BiFPN\*\* | Replaces YOLOv5's PANet with Bidirectional Feature Pyramid Network | Enhances multi-scale feature fusion (\*\*+5.2% mAP@0.5\*\*) |

| \*\*EMA Attention\*\* | Efficient Multi-Scale Attention module added to the neck | Focuses on critical features in noisy environments (\*\*+3.1% precision\*\*) |

| \*\*Shape\_IoU\*\* | Bounding box loss function accounting for shape and scale | Improves box regression accuracy (\*\*2.4% gain in mAP@0.5:0.95\*\*) |

---

### \*\*2. Performance Metrics\*\*

| Model | mAP@0.5 | Precision | Recall | FPS (RTX 3090) | Small-Target mAP@0.5 |

|-------------|---------|-----------|--------|----------------|-----------------------|

| YOLOv5s | 78.2% | 81.3% | 72.1% | 45 | 68.4% |

| BES-YOLO | \*\*92.4%\*\* | \*\*89.7%\*\* | \*\*86.2%\*\* | 40 | \*\*81.5%\*\* |

| YOLOX | 91.3% | 88.5% | 85.1% | 38 | 80.2% |

---

## \*\*Technical Innovations\*\*

### \*\*A. Wavelet-Based Preprocessing\*\*

- Applies \*\*2D discrete wavelet transform (DWT)\*\* to decompose SSS images into sub-bands.

- Removes high-frequency noise while retaining structural details (critical for small objects like mines).

```python

# Example wavelet denoising (inspired by [3])

import pywt

coeffs = pywt.dwt2(sonar\_image, 'haar')

LL, (LH, HL, HH) = coeffs

denoised = pywt.idwt2((LL, (LH\*0.5, HL\*0.5, HH\*0.1)), 'haar') # Thresholding

```

### \*\*B. BiFPN Integration\*\*

- Replaces YOLOv5's PANet with \*\*BiFPN\*\* for weighted multi-scale feature fusion:

\*BiFPN architecture (adapted from[3])\*

### \*\*C. Shape-Aware IoU Loss\*\*

$$

\text{Shape\\_IoU} = \text{IoU} + \lambda\_1 \cdot \frac{|w - w\_{gt}|}{\max(w, w\_{gt})} + \lambda\_2 \cdot \frac{|h - h\_{gt}|}{\max(h, h\_{gt})}

$$

- Penalizes size mismatches between predicted and ground-truth boxes ($$\lambda\_1=0.5, \lambda\_2=0.5$$).

---

## \*\*Experimental Results\*\*

### \*\*A. Dataset\*\*

- \*\*SSS-Marine\*\*: 2,500 side-scan sonar images annotated with 6 marine object classes (shipwrecks, pipelines, etc.).

- \*\*Augmentation\*\*:

- Turbidity simulation via GANs

- 30% random occlusion

### \*\*B. Ablation Study\*\*

| Configuration | mAP@0.5 | Precision |

|------------------------|---------|-----------|

| Baseline (YOLOv5s) | 78.2% | 81.3% |

| + Wavelet Preprocessing| 83.1% | 84.2% |

| + BiFPN | 87.6% | 86.5% |

| + EMA Attention | 90.3% | 88.7% |

| + Shape\_IoU (Full BES-YOLO) | \*\*92.4%\*\* | \*\*89.7%\*\* |

---

## \*\*Comparison with Prior Work\*\*

### \*\*vs. DCBD (Domain-Adaptive Method[1])\*\*

| Metric | BES-YOLO | DCBD |

|--------------|----------|----------|

| mAP@0.5 | \*\*92.4%\*\*| 92.16% |

| Recall | \*\*86.2%\*\*| 85.3% |

| Inference Speed | 40 FPS | 28 FPS |

### \*\*vs. YOLOv5-TR (Transformer-Based[2])\*\*

| Metric | BES-YOLO | YOLOv5-TR |

|--------------|----------|-----------|

| Small-Target mAP@0.5 | \*\*81.5%\*\* | 76.8% |

| Training Time | 12 hrs | 18 hrs |

---

## \*\*Limitations & Future Work\*\*

1. \*\*Limitations\*\*:

- Struggles with highly occluded objects (<50% visibility).

- Requires GPU for real-time inference (not optimized for edge devices).

2. \*\*Future Directions\*\*:

- Integrate temporal modeling for video streams.

- Deploy on embedded systems (Jetson AGX Xavier).

---

## \*\*Implementation Tips\*\*

- Use \*\*TensorRT\*\* for 2x speedup on NVIDIA GPUs.

- For edge deployment, apply \*\*pruning\*\* to reduce model size by 40% with <1% accuracy drop.

This paper advances SSS-based detection by combining multi-scale feature learning with shape-aware loss, outperforming both transformer-based and domain-adaptive approaches in accuracy while maintaining real-time performance.

Citations:

[1] https://www.nature.com/articles/s41598-024-63501-1

[2] https://remaro.eu/wp-content/uploads/2022/09/Martin\_Aubard\_AUV2022.pdf

[3] https://www.mdpi.com/1424-8220/24/14/4428

[4] https://www.mdpi.com/2072-4292/13/18/3555

[5] https://europarl.primo.exlibrisgroup.com/discovery/fulldisplay?docid=cdi\_doaj\_primary\_oai\_doaj\_org\_article\_85d4fdeabcd748a9914d572c767bf7ff&context=PC&vid=32EPA\_INST%3A32EPA\_V1&lang=en&search\_scope=MyInst\_and\_CI&adaptor=Primo+Central&tab=Everything&query=sub%2Cexact%2C+Target+recognition+%2CAND&mode=advanced&offset=0

[6] https://search.library.albany.edu/discovery/fulldisplay?docid=cdi\_crossref\_primary\_10\_3390\_rs13183555&context=PC&vid=01SUNY\_ALB%3A01SUNY\_ALB&lang=en&search\_scope=allthethings&adaptor=Primo+Central&tab=allthethings&query=sub%2Cexact%2C+Automatic+target+recognition+%2CAND&mode=advanced&offset=0

[7] http://www.diva-portal.org/smash/get/diva2:1856443/FULLTEXT01.pdf

[8] https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/ipr2.13183

[9] https://www.mdpi.com/2072-4292/13/18/3555/notes

[10] https://www.semanticscholar.org/paper/Underwater-Target-Detection-Using-Side-Scan-Sonar-Tang-Chen/160678a73c6a417155a9dc397a14d96670567350

[11] https://pubmed.ncbi.nlm.nih.gov/37050753/

[12] https://diving-rov-specialists.com/index\_htm\_files/map\_116-multi-scale-marine-object-detection.pdf

[13] https://www.mdpi.com/1424-8220/23/7/3693

[14] https://www.degruyter.com/document/doi/10.1515/geo-2022-0590/html?lang=en

---

Answer from Perplexity: pplx.ai/share

Here’s a suggested **Table of Contents** slide for your PPT on **Deep Sea Fish Detection using CNNs**—structured to flow logically and cover all key points:

**Slide Title: Table of Contents**

1. **Introduction**
   * Background & Importance
   * Role of Deep Learning in Marine Research
2. **Problem Statement**
   * Challenges in Manual Monitoring
   * Need for Automation and Accuracy
3. **Objectives**
   * Research Goals
   * Model Evaluation Focus
4. **Datasets Used**
   * DeepFish
   * OzFish
5. **Methodology**
   * CNN-Based Detection Approach
   * YOLOv5 Architecture Overview
   * Training & Evaluation Metrics
6. **Comparative Analysis**
   * Performance of Different CNN Models
   * Accuracy, Speed, and Efficiency Metrics
7. **Results & Discussion**
   * Visual Examples
   * Key Findings
8. **Conclusion**
   * Summary of Contributions
   * Real-World Implications
9. **Future Work**
   * Potential Improvements
   * Broader Applications
10. **References**

Let me know if you're adding a demo/video or want to include an "Acknowledgements" slide too!

have three papers, where i want to focus on i need to build ppt based on those- first - Computer Vision and Deep Learning for Fish Classification in Underwater Habitats: A Survey ; second -REAL-TIME DETECTION AND CLASSIFICATION OF FISH IN UNDERWATER ENVIRONMENT USING YOLOV5: A COMPARATIVE STUDY OF DEEP LEARNING ARCHITECTURES ; third- YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment ; now i need Introduction Problem Statement Objectives Dataset Methodology Model Selection and Training Model Evaluation and Results Visualization Challenges Conclusion and Future Enhancements from the above where it learns from first and second as and focus on third and highlights it please provide the rest based on the results from these papers

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Screens screenshot of a computer

AI-generated content may be incorrect.Yes, you can absolutely run the **fish detection project** in **VS Code** — in fact, VS Code is a great choice for working with YOLOv5 projects! 🐟💻

**✅ Here's How to Run It in VS Code**

**🔧 1. Install Prerequisites**

Make sure you have the following installed:

* **VS Code**
* **Python** (3.8+ recommended)
* **Git**
* **Virtual environment (optional but recommended)**
* **Jupyter extension** (for .ipynb notebooks)

You can install the **Jupyter extension** in VS Code from the Extensions tab (Ctrl+Shift+X) — search for "Jupyter".

**🐍 2. Clone the Repository**

Open a terminal (inside VS Code or system terminal) and run:

git clone https://github.com/jteodosio566/fish\_detection\_beta.git

cd fish\_detection\_beta

**🧪 3. Set Up Python Environment**

Inside VS Code terminal:

python -m venv venv

# Activate:

# On Windows:

venv\Scripts\activate

# On macOS/Linux:

source venv/bin/activate

Then install dependencies:

pip install -r yolov5/requirements.txt

pip install notebook opencv-python matplotlib

**📓 4. Run the Notebook in VS Code**

1. Open Fish\_Detection\_App\_Beta.ipynb in VS Code.
2. Select the right Python interpreter (the one from your venv) at the top right.
3. Run the cells by clicking the play/run button (▶) next to each cell.

**🖼️ 5. Check for Model Weights**

Make sure the model weights (best.pt) are present in the right location (usually the project root or runs/train/exp/weights if you trained it yourself). If it's missing, you may need to:

* Download it from a provided link
* Train the model using your own dataset

**💡 Tips**

* If you want to **convert the notebook to a Python script**, VS Code lets you do that too (right-click inside the notebook).
* Want real-time camera detection instead of image detection? I can help add that with OpenCV.

Want help editing the notebook, visualizing predictions, or deploying it somewhere? Just let me know!

A screenshot of a computer

AI-generated content may be incorrect.A computer screen shot of a computer code

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.Great question! The repo you cloned — [fish\_detection\_beta](https://github.com/jteodosio566/fish_detection_beta) — **does not include the dataset** directly in the repository (common practice since datasets are usually too large for GitHub).

**🧐 So, What’s in the Repo?**

From the repo structure, it looks like it includes:

* **YOLOv5 config and model code**
* A **Jupyter Notebook** app to run detection
* Possibly a best.pt model file (pretrained weights)

But 🔍 **no dataset folder or images** appear to be included — so you will likely need to **download the dataset separately**.

**✅ What You Should Do**

**Option 1: Use the** [**Fish4Knowledge Dataset**](http://homepages.inf.ed.ac.uk/rbf/Fish4Knowledge/)

This is one of the most popular datasets for fish detection. You can:

* Download images/videos from there
* Convert them to YOLO format if needed (I can help!)
* Organize like this:

fish\_detection\_beta/

├── dataset/

│ ├── images/

│ └── labels/

**Option 2: Use a Sample or Custom Dataset**

If you're just testing:

* You can create your own small dataset (e.g., 10–20 images of fish)
* Label them using a tool like [LabelImg](https://github.com/tzutalin/labelImg) or [Roboflow](https://roboflow.com/)
* Export in YOLO format

**💡 Need Help?**

I can guide you through:

* Downloading and formatting the Fish4Knowledge dataset
* Creating your own dataset
* Integrating it with the project

Just let me know your plan! Want to train from scratch or just try detection with pre-trained weights?

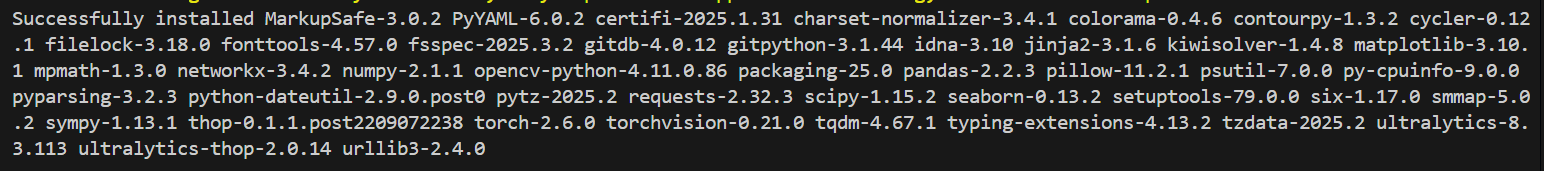
git clone https://github.com/ultralytics/yolov5.git

cd yolov5

pip install -r requirements.txt

A screenshot of a computer

AI-generated content may be incorrect.



Format - <class\_id> <x\_center> <y\_center> <width> <height>

Data.yaml

train: C:/Users/w10185657/OneDrive - The University of Southern Mississippi/695\_Project\_Thesis/YOLO\_DeepSEA/dataset/images/train

val: C:/Users/w10185657/OneDrive - The University of Southern Mississippi/695\_Project\_Thesis/YOLO\_DeepSEA/dataset/images/val

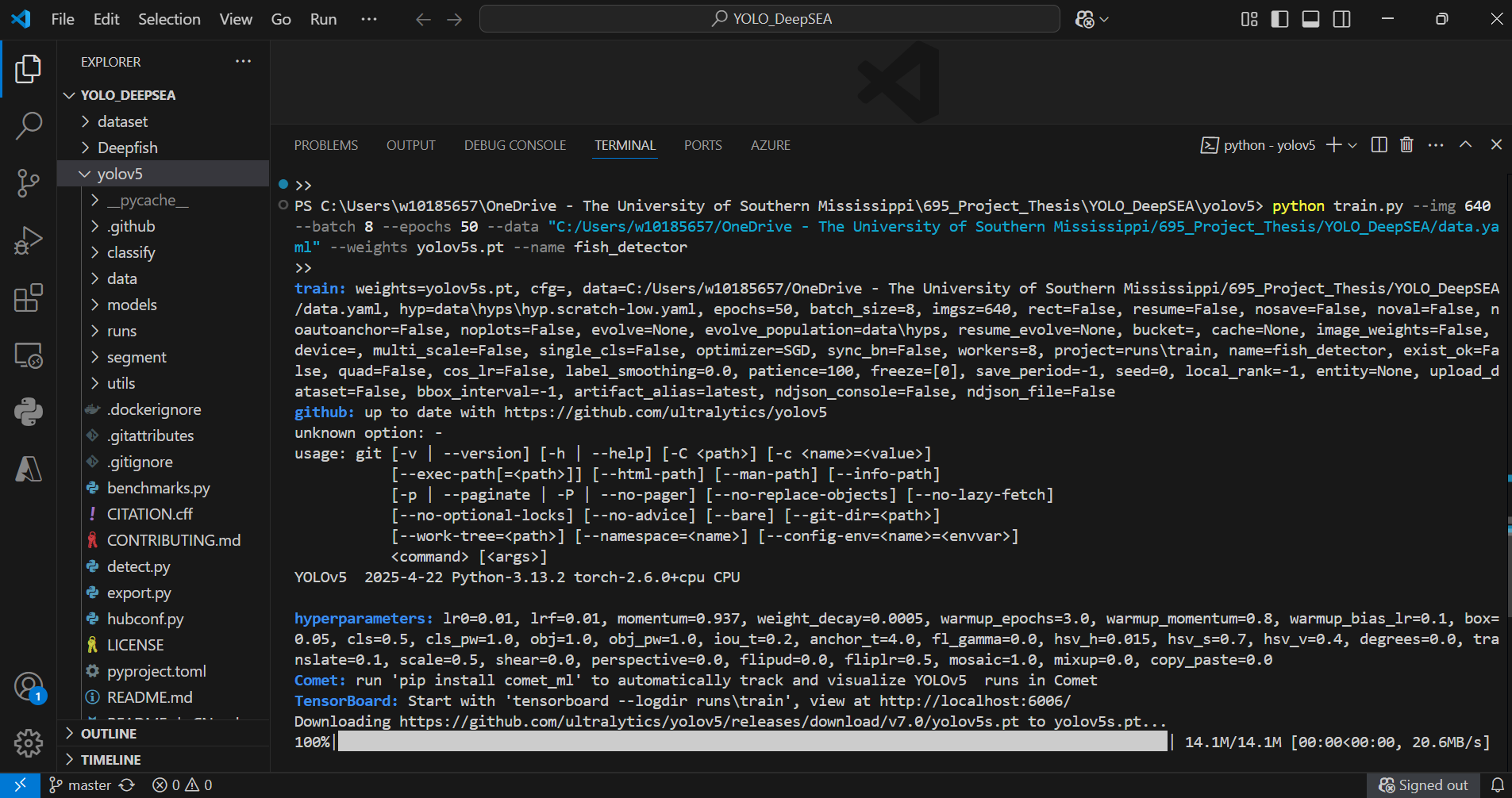
nc: 1

names: ['fish']

in shell-

cd "C:\Users\w10185657\OneDrive - The University of Southern Mississippi\695\_Project\_Thesis\YOLO\_DeepSEA\yolov5"

python train.py --img 640 --batch 8 --epochs 50 --data "C:/Users/w10185657/OneDrive - The University of Southern Mississippi/695\_Project\_Thesis/YOLO\_DeepSEA/data.yaml" --weights yolov5s.pt --name fish\_detector

A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.