Detection, classification and tracking of fishes: an analysis of the YOLOv8 model

July 25, 2025

Fish detection problem

History of fish detection

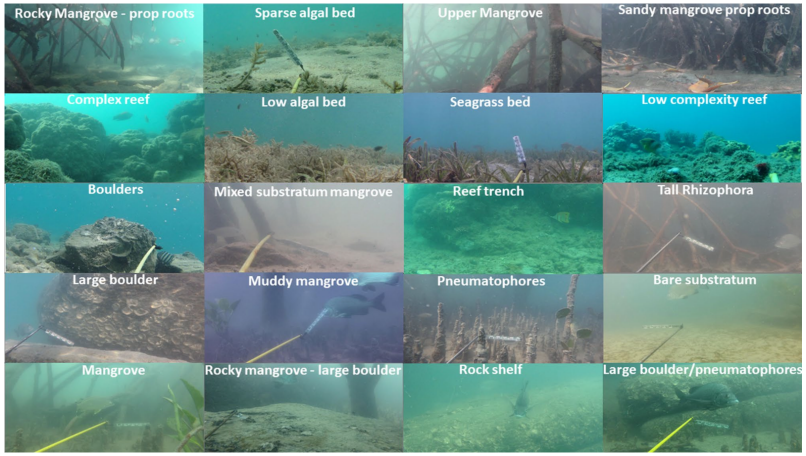
YOLO

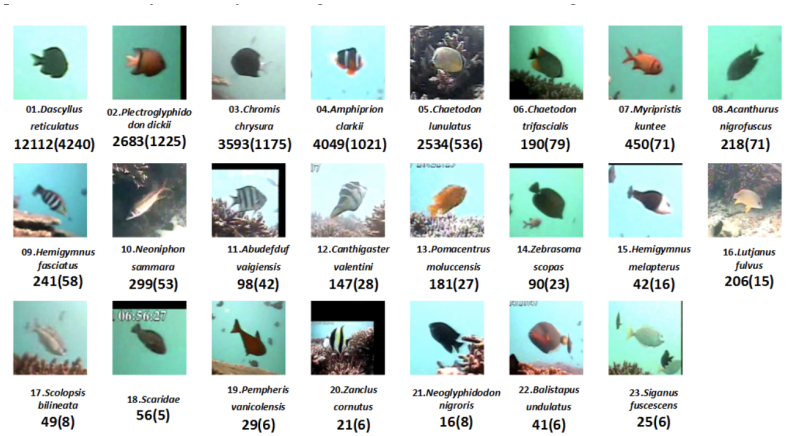
YOLOv8

Sub-things to explain

Datasets

DeepFish ->





<- Fish4Knowledge

Configuration files

```
train: path_to_train/images
val: path_to_val/images
test: path_to_test/images

nc: 1
names: ['Fish']
```

```
train: path_to_train/images
val: path_to_val/images
test: path_to_test/images

nc: 23
names: [
  'Caranx_sexfasciatus',
  'F1',
  'F2',
  'F3',
  'F4',
  'F5',
  'F6',
  'F7',
  'Acanthopagrus_palmaris',
  'Lutjanus_russellii',
  'acanthopagrus_and_caranx',
  'acanthopagrus_palmaris',
  'Gerres',
  'Caranx',
  'Amniataba_caudivittatus',
  'gerres_2',
  'gerres',
  'Epinephelus',
  'Fish',
  'juvenile',
  'palmaris',
  'EJP',
  'caudivittatus'
]
```

YOLOv8 models

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

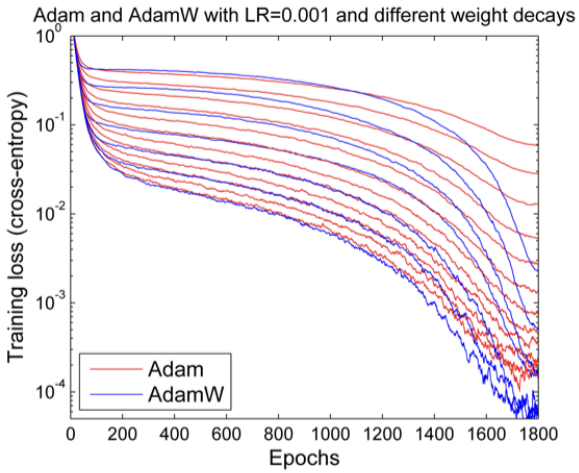
AdamW

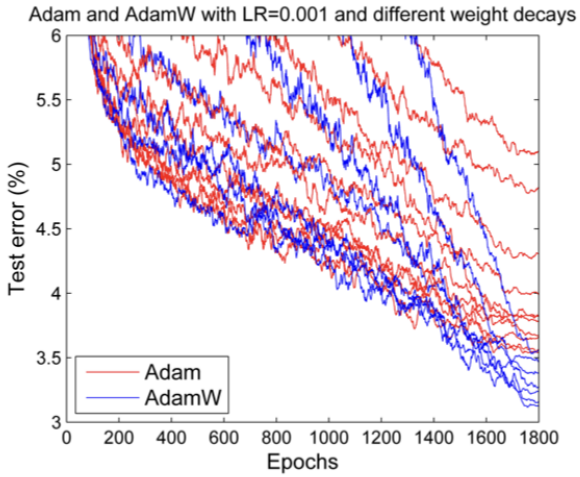
Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

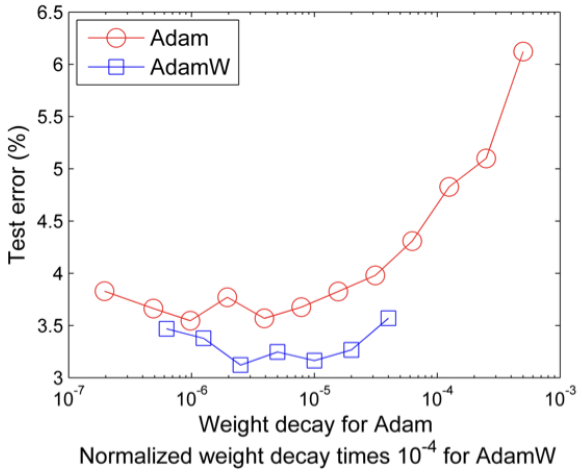
given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

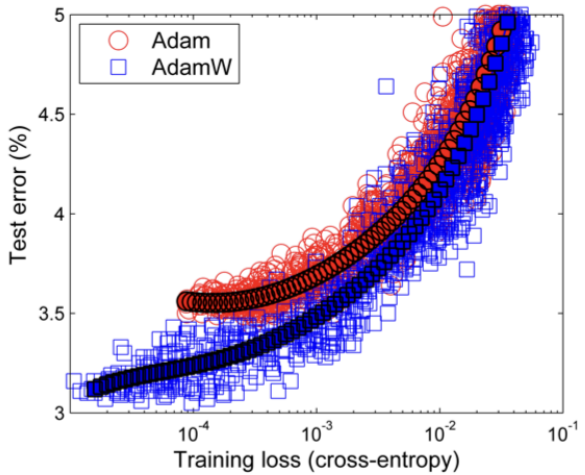
1

[1]


Adam and AdamW with LR=0.001 and different weight decays
Training loss (cross-entropy)
Epochs
Adam
AdamW


Adam and AdamW with LR=0.001 and different weight decays
Test error (%)
Epochs
Adam
AdamW


Test error (%)
Adam
AdamW
Weight decay for Adam
Normalized weight decay times 10^-4 for AdamW


Test error (%)
Adam
AdamW
Training loss (cross-entropy)

[2]

[2] [1]

Loss Function

$$loss = L_{cls} + L_{box} + L_{DFL}$$

- **Classification loss:** crucial for predicting the correct class of objects.
- **Box loss:** ensures that the bounding boxes around objects are accurate.
- **DFL:** Distributed Focal Loss, boosts performance by distributing the loss function across different object scales and classes.

General Parameters

Parameter	**Value**	**Description**
Epochs	50-100	Number of training iterations
Batch Size	16	Images processed simultaneously
Image Size	640	Input image resolution (pixels)
Optimizer	AdamW	Adaptive moment estimation with weight decay
Learning Rate	0.001	Initial learning rate with cosine scheduling
Weight Decay	0.0005	L2 regularization strength
patience	10	Epochs to wait before early stopping
Warmup epochs	3	Gradual learning rate increase at start
cos_lr	True	Cosine learning rate scheduler
close_mosaic	10	Disable mosaic augmentation in last N epochs

Specific Parameters (single-class vs multi-class)

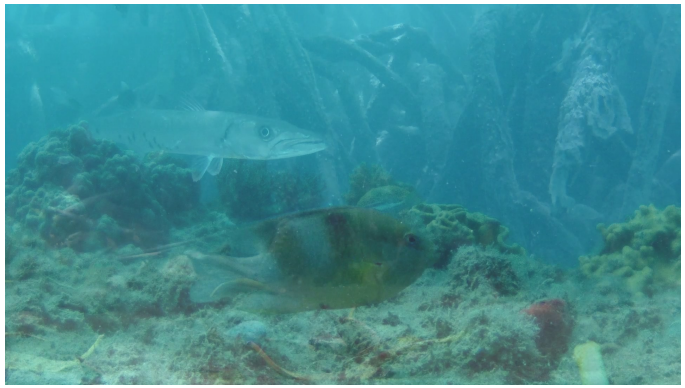
Parameter	**Single-class**	**Multi-class**	**Description**
cls	0.5	1.0	Weight affecting importance of correct class prediction
box	7.5	7.5	Weight influencing emphasis on accurately predicting bounding box coordinates
dfl	1.5	1.5	Weight used for fine-grained classification
mixup	0.0	0.15	Blends two images and their labels, creating composite image
copy_paste	0.0	0.3	Copies and pastes objects across images to increase object instances

Training

```
(dl-env) ✗ ~/UniTs/First_Year/Deep_Learning/Deep-Learning/src/object_detector  master ± python3 train_yolo.py --data ../../Datasets/Deepfish_YOLO/data.yaml --device mps --model s
🔧 CUDA available: False
🔧 MPS available: True
🔧 Detected device: mps
📦 Loading COCO pretrained YOLOv8s
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8s.pt to 'models/yolov8s.pt'...
100%|██████████| 21.5M/21.5M [00:07<00:00, 2.84MB/s]
🏋️  Training Configuration:
   Mode: multi-class
   Model: YOLOv8s
   Classes: 1
   Epochs: 50
   Batch size: 16
   Image size: 640
   Device: mps
   Config: ../../Datasets/Deepfish_YOLO/data.yaml

🚀 Starting training...
New https://pypi.org/project/ultralytics/8.3.169 available 😃 Update with 'pip install -U ultralytics'
Ultralytics 8.3.163 🚀 Python-3.13.5 torch-2.7.1 MPS (Apple M3)
```

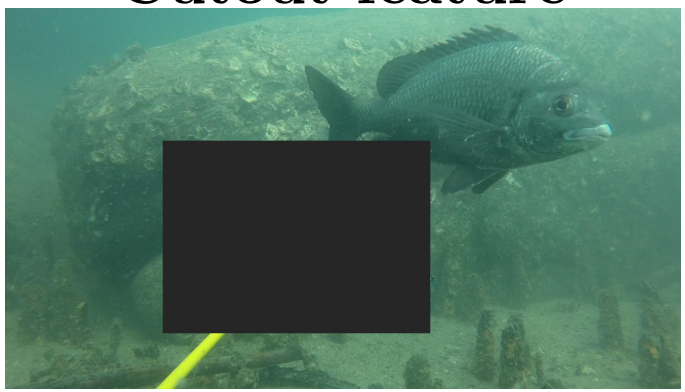
```
Model summary: 129 layers, 11,135,987 parameters, 11,135,971 gradients, 28.6 GFLOPs

Transferred 349/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
train: Fast image access ✅ (ping: 0.2±0.1 ms, read: 680.8±166.8 MB/s, size: 155.2 KB)
train: Scanning /Users/christianfaccio/UniTs/First_Year/Deep_Learning/Deep-Learning/Datasets/Deepfish_YOLO/train/labels
val: Fast image access ✅ (ping: 0.1±0.0 ms, read: 786.3±105.5 MB/s, size: 164.2 KB)
val: Scanning /Users/christianfaccio/UniTs/First_Year/Deep_Learning/Deep-Learning/Datasets/Deepfish_YOLO/val/labels.cac
Plotting labels to src/object_detector/runs/train/fish_multi_class_s/labels.jpg...
optimizer: AdamW(lr=0.001, momentum=0.937) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to src/object_detector/runs/train/fish_multi_class_s
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50      7.46G        nan        nan        nan         94        640: 100%|██████████| 209/209 [07:41<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 33/33 [01:01
                   all       1042       3657          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50      7.42G        nan        nan        nan        182        640: 100%|██████████| 209/209 [10:14<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 33/33 [00:32
                   all       1042       3657          0          0          0          0
```

Mixup feature



Mosaic feature

Cutout feature



NMS

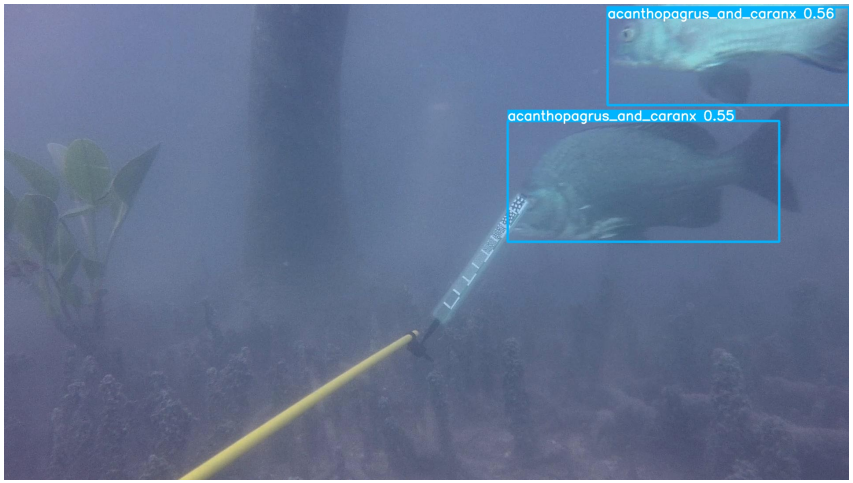
The **Non-Maximum Suppression** is a technique that cleans up the output of a detection model by filtering out redundant and overlapping bounding boxes to ensure that each object is identified only once. It operates by iterating through the predicted bounding boxes and making decisions based on two key metrics: **confidence scores** and the **IoU** threshold:

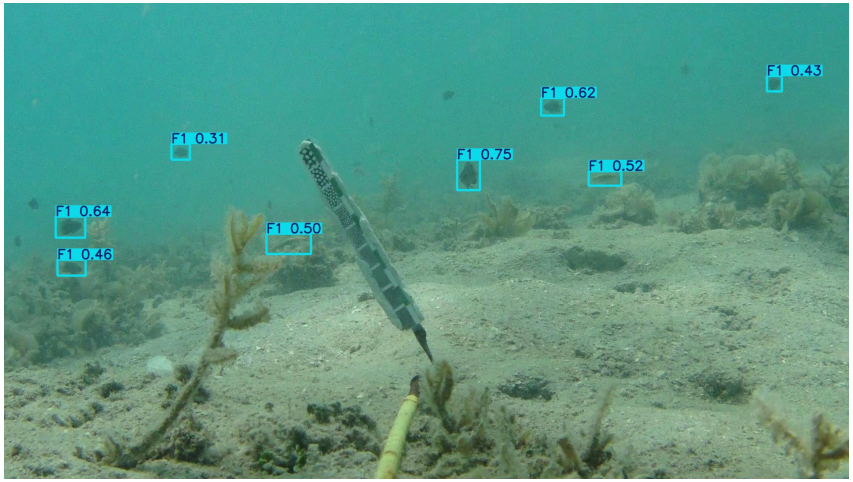
1. Sort by confidence
2. Select the best box
3. Repeat

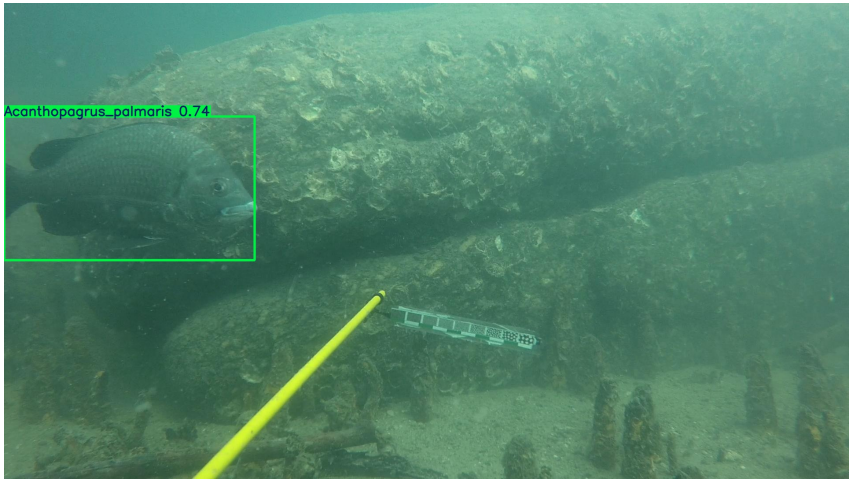
(IoU means Intersection over Union and the higher it is for a box, the more it overlaps the others)

Inference


Amniataba_caudivittatus 0.76


acanthopagrus_and_caranx 0.56
acanthopagrus_and_caranx 0.55


F1 0.43
F1 0.62
F1 0.31
F1 0.75
F1 0.52
F1 0.64
F1 0.50
F1 0.46


Acanthopagrus_palmaris 0.74

The Tracking Challenge: Occlusion & ID Switches

Problem: In complex underwater scenes, fish are frequently occluded by objects or other fish.

A naive tracker will often:

- **Lose the track** entirely when the fish disappears.
- **Create a new ID** when the fish reappears (ID Switch).

Goal: Maintain a persistent, correct ID for each fish despite occlusions.

The ByteTrack Algorithm

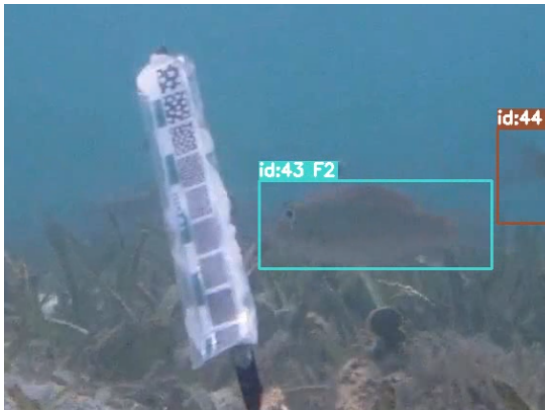
Stage 1: High-Confidence Match

- Uses motion prediction (Kalman Filter) to estimate the next position of existing tracks.
- Matches these predictions with high-confidence detections from the YOLO model.

Stage 2: Low-Confidence Match

- Instead of discarding low-confidence detections, it keeps them.
- It attempts to match these "doubtful" detections to the tracks that were lost in Stage 1.

Occlusion Example

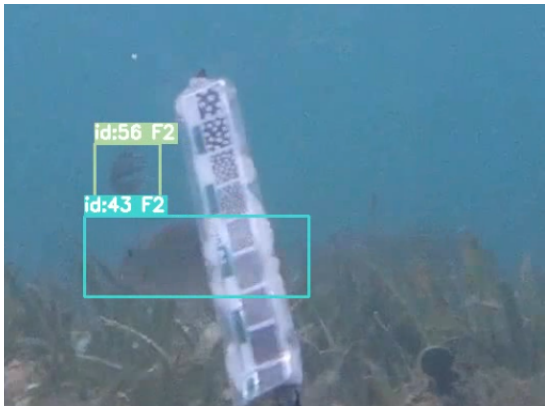


1. Before Occlusion

The fish ID 43 is clearly visible and being tracked.

2. During Occlusion

Fish ID 43 is now hidden behind an object.



3. After Occlusion

The fish reappears. ByteTrack re-identifies it as ID 43.

[1] Ilya Loshchilov and Frank Hutter.
Decoupled weight decay regularization, 2019.