



**UNIVERSITÀ
DEGLI STUDI
DI TRIESTE**

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

July 25, 2025

① Theory

② Detection

③ Tracking

④ Results

⑤ Conclusions

1 Theory

2 Detection

3 Tracking

4 Results

5 Conclusions

Theory
○●○○○

Detection
○○○○○○○○○○○○○○○○

Tracking
○○○○

Results
○

Conclusions
○

Fish detection problem

Theory
oo●ooo

Detection
oooooooooooooo

Tracking
oooo

Results
o

Conclusions
o

History of fish detection

Theory
ooo•oo

Detection
oooooooooooooo

Tracking
oooo

Results
o

Conclusions
o

YOLO

Theory
oooo●o

Detection
oooooooooooooooooooo

Tracking
oooo

Results
○

Conclusions
○

YOLOv8

Theory
ooooo●

Detection
oooooooooooooo

Tracking
oooo

Results
○

Conclusions
○

Sub-things to explain

① Theory

② Detection

③ Tracking

④ Results

⑤ Conclusions

Datasets

DeepFish ->



	41. <i>Dascyllus reticulatus</i> 12112(4240)		62. <i>Plectroglyphidodon olivaceus</i> 2683(1225)		63. <i>Chromis oxyura</i> 3593(1175)		64. <i>Acanthurus leucosternon</i> 4049(1021)		65. <i>Chromis lunulata</i> 2534(536)		66. <i>Chromis trifasciata</i> 190(79)		67. <i>Mugilogobius asterias</i> 450(71)		68. <i>Anthurus spilopterus</i> 218(71)
	69. <i>Amphiprion frenatus</i> 241(58)		70. <i>Hemigymnus melapterus</i> 299(53)		71. <i>Acanthurus leucosternon</i> 98(42)		72. <i>Ctenochaetus striatus</i> 147(28)		73. <i>Pomacentrus moluccenis</i> 181(27)		74. <i>Dascyllus reticulatus</i> 90(23)		75. <i>Hemitaurichthys mitchilli</i> 42(16)		76. <i>Hemitaurichthys zoster</i> 206(15)
	77. <i>Apogonichthys filamentosus</i> 49(8)		78. <i>Acanthurus leucosternon</i> 56(5)		79. <i>Pomacentrus semicirculatus</i> 29(6)		80. <i>Zanclorhynchus cornutus</i> 21(6)		81. <i>Acanthurus nigrofasciatus</i> 16(8)		82. <i>Zanclorhynchus cornutus</i> 41(6)		83. <i>Siganus laqueus</i> 25(6)		

<- Fish4Knowledge

Datasets structure

DeepFish

```
1 .
2 7117
3     train
4     valid
5 7268
6 7393
7 7398
8 7426
9 7434
10 7463
11 7482
12 7490
13 7585
14 7623
15 9852
16 9862
17 9866
18 9870
19 9892
20 9894
21 9898
22 9907
23 9908
24 classes.txt
25 Nagative_samples
26 test
```

Fish4Knowledge

```
1 f4k
2     fish_image
3         fish_01
4             fish_x.png
5             ...
6         gt_bounding_boxes
7             gt_106.xml
8             ...
9         mask_image
10            mask_01
11                mask_x.png
12                ...
13         videos
14             gt_106.mp4
15             ...
```

YOLO format

```
1 .
2 data.yaml
3 test
4     images
5     labels
6     labels.cache
7 train
8     images
9     labels
10    labels.cache
11 val
12     images
13     labels
14     labels.cache
```

Configuration files (data.yaml)

```
1 train: path_to_train/images
2 val: path_to_val/images
3 test: path_to_test/images
4
5 nc: 1
6 names: ['Fish']
```

```
1 train: path_to_train/images
2 val: path_to_val/images
3 test: path_to_test/images
4
5 nc: 23
6 names: [
7   'Caranx_sexfasciatus',
8   'F1',
9   'F2',
10  'F3',
11  'F4',
12  'F5',
13  'F6',
14  'F7',
15  'Acanthopagrus_palmaris',
16  'Lutjanus_russellii',
17  'acanthopagrus_and_caranx',
18  'acanthopagrus_palmaris',
19  'Gerres',
20  'Caranx',
21  'Amniataba_caudivittatus',
22  'gerres_2',
23  'gerres',
24  'Epinephelus',
25  'Fish',
26  'juvenile',
27  'palmaris',
28  'EJP',
29  'caudivittatus'
30 ]
```

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

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \left(\frac{m_t}{v_t + \epsilon} + \lambda \phi_t \right)$$

Where:

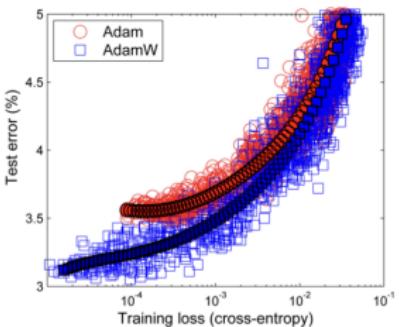
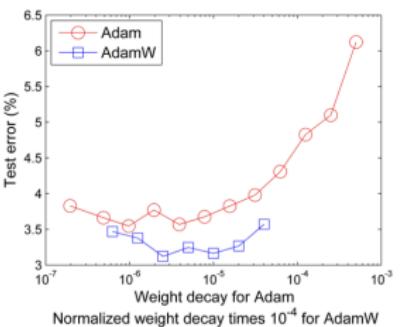
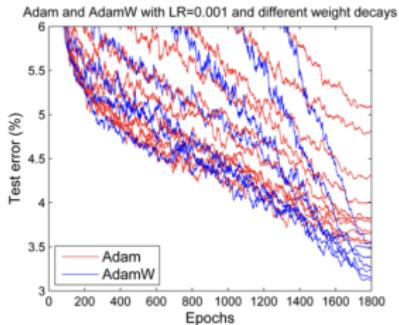
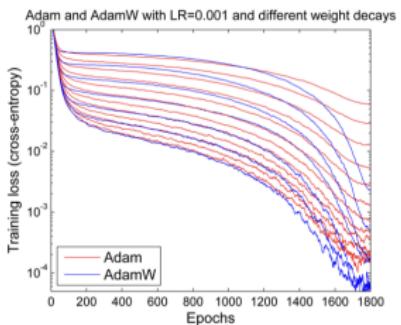
- $m_t \leftarrow \frac{\beta_1 \cdot m_{t-1} + (1 - \beta_1) g_t}{1 - \beta_1^t}$
- $v_t \leftarrow \frac{\beta_2 \cdot v_{t-1} + (1 - \beta_2) g_t^2}{1 - \beta_2^t}$
- $g_t \leftarrow \nabla L_t(\phi_{t-1}) \quad (\text{no } \lambda \phi_{t-1} \text{ here!})$

Algorithm 2 Adam with L₂ regularization and Adam with decoupled weight decay (AdamW)

```

1: given  $\alpha = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ ,  $\lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $m_{t=0} \leftarrow \theta$ , second moment
   vector  $v_{t=0} \leftarrow \theta$ , schedule multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$                                  $\triangleright$  select batch and return the corresponding gradient
6:    $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$ 
7:    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$                                  $\triangleright$  here and below all operations are element-wise
8:    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
9:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$                                           $\triangleright \beta_1$  is taken to the power of  $t$ 
10:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$                                           $\triangleright \beta_2$  is taken to the power of  $t$ 
11:   $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$                                  $\triangleright$  can be fixed, decay, or also be used for warm restarts
12:   $\theta_t \leftarrow \theta_{t-1} - \eta_t \left( \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1} \right)$ 
13: until stopping criterion is met
14: return optimized parameters  $\theta_t$ 

```



2

Loss Function

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

- **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) x ~/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]  
X 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 8.3.169 🚀 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/training/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
```

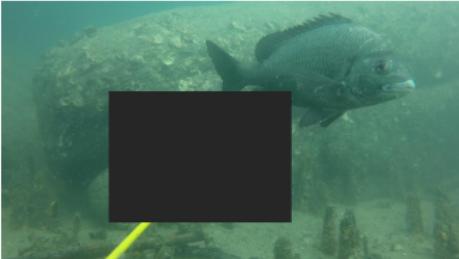
Mixup feature



Mosaic feature



Cutout feature



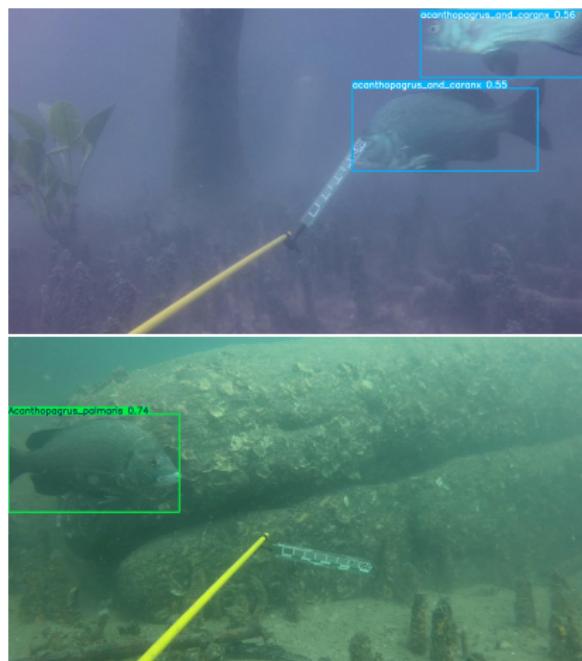
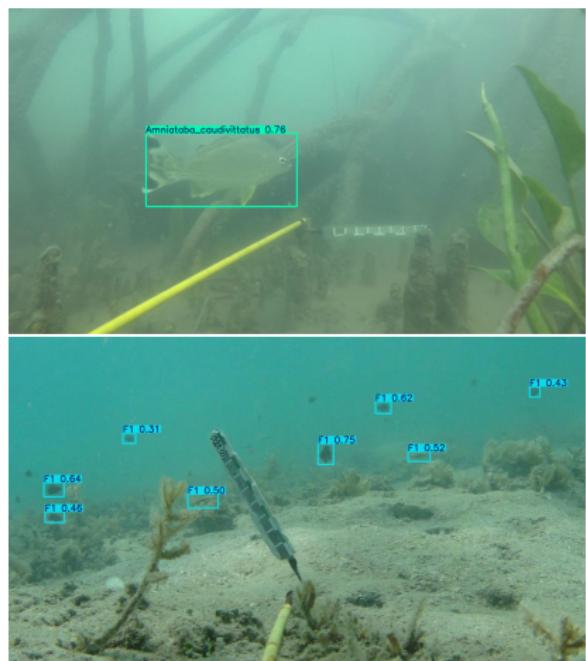
NMS

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



① Theory

② Detection

③ Tracking

④ Results

⑤ Conclusions

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

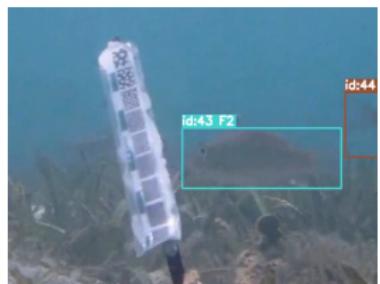
Stage 1: High-Confidence Match

- 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



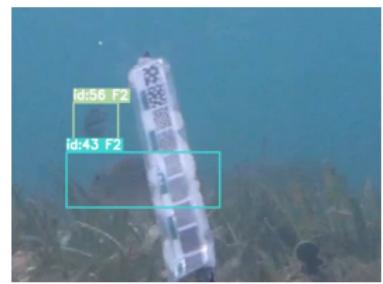
1. Before Occlusion

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 Theory

② Detection

③ Tracking

④ Results

⑤ Conclusions

① Theory

② Detection

③ Tracking

④ Results

⑤ Conclusions

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