

## Article

# Towards Robust Underwater Object Detection: UWOD Dataset and Transfer Learning Insights

Nouf A. Alrowais <sup>†</sup>, Anfal M. Alawajy <sup>†</sup> , Nada A. Almugrem <sup>†</sup> , Hadeel M. Aljami , Abdulaziz O. Alobaid , Masheal M. Alghamdi <sup>\*</sup> , Walaa A. Alsumari , Hassan R. Alqaeri , Aljwhara Almutairi , Remass Alsaeed, Royouf Alotaibi , Aghadir A. Jammah and Eman Bin Khunayn 

King Abdulaziz City for Science and Technology (KACST), Riyadh 11442, Saudi Arabia; aalawajy@kacst.gov.sa (A.M.A.); nmugren@kacst.gov.sa (N.A.A.); hassn\_120@outlook.com (H.R.A.); aghadirjammah@gmail.com (A.A.J.); ekhunayn@kacst.gov.sa (E.B.K.)

<sup>\*</sup> Correspondence: malghamdi@kacst.gov.sa

<sup>†</sup> These authors contributed equally to this work.

## Abstract

Underwater object detection faces significant challenges including uneven lighting, low contrast, and scattering-induced distortions. Existing underwater datasets are limited in scale, class diversity, and annotation consistency, which hinders robust model development. This work addresses these limitations by creating the UWOD dataset through the integration of seven publicly available underwater datasets, comprising over 107 K images with approximately 374 K annotations across 39 classes. We employ a semi-automatic annotation pipeline that combines manual labeling with iterative model-in-the-loop training to ensure high-quality ground truth. As a final verification step, all auto-generated labels were manually inspected and corrected as needed. We benchmark state-of-the-art object detectors—including YOLOv8, YOLOv7, YOLOv5, FCOS, EfficientDet, YOLOX, RT-DETR, SSD, and Faster R-CNN—establishing comprehensive performance baselines; YOLOv8 and YOLOv7 achieve the best accuracy–efficiency trade-off. Our transfer learning analysis shows that domain-specific pretraining substantially often outperforms pretraining on general-purpose datasets, yielding up to more than 50% improvement in low-data regimes versus training from scratch, with markedly lower seed-to-seed variance than scratch training. Sequential pretraining on COCO followed by UWOD achieves the strongest results on our most challenging dataset. Due to upstream licensing constraints, we release trained model weights and an automated annotation pipeline that encapsulate the learned underwater-domain knowledge, enabling immediate application to new imagery while respecting intellectual property.

**Keywords:** computer vision; transfer learning; deep learning; convolutional neural network; underwater; aquatic; marine; dataset; benchmark



Academic Editors: Daqian Shi,  
Wei Kong and Yuqi Ouyang

Received: 20 July 2026

Revised: 3 September 2026

Accepted: 5 September 2026

Published: 11 September 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction

Underwater Object Detection (UOD) has emerged as a significant area in computer vision with applications in pipeline leakage detection [1], underwater robotics [2], environmental monitoring [3–5], and maritime security [6,7]. UOD remains challenging due to water turbidity, wavelength-dependent absorption, and scattering [8–10]. Public underwater datasets also tend to be limited in size, resolution, class coverage, and resource availability, which constrains model performance and generalization [11]. Despite these

challenges, progress in deep learning and modern detectors continues to drive advances in underwater perception [12–14].

Deep learning methods [15,16] generally outperform traditional approaches [17–21], especially at scale [22–25]. Because performance is data-dependent, transfer learning [26–28] is widely used to leverage knowledge from large datasets; however, dataset shift can cause negative transfer [29,30]. In underwater settings, models pretrained on general-purpose imagery may underperform relative to domain-specific pretraining [31–33]. Therefore, a large, public, and diverse underwater object dataset is foundational for developing reliable detectors [13,34,35]. A common practice is to fine-tune detectors pretrained on MS COCO [36], which complicates apples-to-apples comparison of underwater detectors and obscures the role of domain-specific pretraining [37–39]. Moreover, prior work provides limited systematic analysis of transfer learning strategies on large-scale underwater datasets [40,41]. To address these gaps, we introduce the UWOD dataset and conduct a comprehensive evaluation of transfer learning strategies for underwater object detection.

Our main contributions are:

- Comprehensive underwater dataset compilation and curation: We compile the UWOD dataset, comprising over 107,000 images and approximately 374,000 annotations across 39 classes, by integrating seven public underwater datasets and employing a semi-automatic, model-in-the-loop annotation pipeline. This addresses critical gaps in scale and class diversity for underwater vision.
- Domain-specific transfer learning analysis: We provide a systematic evaluation of transfer learning strategies tailored to underwater detection, demonstrating that underwater-specific pretraining substantially outperforms general-purpose pretraining, with up to 50% gains in low-data regimes compared to training from scratch.
- Systematic performance benchmarking: We establish strong baselines for underwater object detection by evaluating YOLOv8 [42], YOLOv7 [43], YOLOv5 [44], FCOS [45], EfficientDet [46], YOLOX [47], RT-DETR [48], SSD [49], and Faster R-CNN [16] on UWOD, providing evidence-based guidance for model selection.
- Released model and pipeline: Subject to upstream licenses, we release trained model weights and an automated annotation pipeline that capture underwater-domain knowledge from our compilation, enabling immediate application and facilitating downstream transfer learning without redistributing restricted data. An interactive notebook implementing this pipeline is publicly available at [https://drive.google.com/drive/folders/1xWJd-q9BjPrOhGMyMknUGAw1PnT0-YoK?usp=drive\\_link](https://drive.google.com/drive/folders/1xWJd-q9BjPrOhGMyMknUGAw1PnT0-YoK?usp=drive_link) (accessed on 4 September 2026), allowing direct application of the released weights for automated annotation.

## 2. Related Work

This section reviews traditional methods, deep learning approaches, dataset resources, and the core challenges of underwater imagery.

### 2.1. Traditional Object Detection Methods

Early object detection relied on handcrafted features and classical classifiers. Viola and Jones [19] introduced a boosted cascade for rapid detection, demonstrating real-time face detection. Dalal and Triggs [18] proposed HOG features for robust human detection. Support Vector Machines (SVMs) [20] were widely used for classification. While effective in controlled settings, such pipelines struggle underwater due to wavelength-dependent absorption, scattering, and low contrast [8–10].



## 2.2. Deep Learning Approaches

Deep learning transformed object detection, with CNN-based models surpassing traditional methods. Two-stage detectors such as Fast R-CNN [50] and Faster R-CNN [16] have been applied to UOD; e.g., Li et al. [15] achieved accurate fish detection with Fast R-CNN, and Huang et al. [51] improved marine organism detection via data augmentation. However, two-stage models can be less suitable for real-time constraints and small/multi-scale targets typical underwater.

One-stage models offer speed advantages and have been adapted to underwater conditions. Mandal et al. [52] assessed fish abundance from video using deep detectors. Liu et al. [53] built an embedded YOLOv3 system for online fish detection/tracking. Jain [54] proposed DeepSeaNet (EfficientDet-based) with enhanced feature fusion. On URPC-style benchmarks [55,56], Faster R-CNN often yields higher AP while YOLOv3 attains higher FPS, reflecting an accuracy–speed trade-off [10]. Subsequent evaluations comparing YOLOv3/4/5/8 and EfficientDet variants report dataset- and condition-dependent effectiveness [54,57].

## 2.3. Challenges and Solutions in Underwater Object Detection

Underwater imagery is degraded by light attenuation, color casts, turbidity, and backscatter, leading to low contrast and reduced SNR [8,58]. Suspended particles further obscure targets and introduce noise [59]. Surveys detail these challenges and their impact on detection [9,10]. Pre-processing pipelines that perform color correction and enhancement prior to detection can improve accuracy [60,61]. Hybrid approaches that integrate traditional priors with learned features have also shown promise [62]. Xie et al. [63] proposed transmission-guided bilevel adaptive learning to better handle underwater degradations.

## 2.4. Transfer Learning and Domain Adaptation

Transfer learning is particularly valuable for UOD given the scarcity and heterogeneity of labeled data [63]. Pretraining on large datasets such as ImageNet [64] and MS COCO [36] improves downstream accuracy [27], while recent studies emphasize adapting to underwater-specific degradations (e.g., visibility and color shifts) [65,66]. Domain adaptation and generalization methods help models cope with distribution shifts across locations and sensors [55,67,68]. Careful source–target alignment is necessary to avoid negative transfer when domains are mismatched [69,70].

## 2.5. Datasets for Underwater Object Detection

General-purpose datasets (ImageNet [64], MS COCO [36], PASCAL VOC [71], Objects365 [72]) underwrite modern detection research by providing diversity and scale. In contrast, underwater datasets are more constrained. The Underwater Robot Picking Contest line of datasets (e.g., URPC [55] and UTDAC [56]) offers task-focused imagery but limited class and environment diversity [55,73]. DUO aggregates and re-annotates multiple contest sources to improve label quality [74]. UODD builds on RUIE [61] with enhanced images for species detection [75]. Brackish targets temperate waters [76,77], while Aquarium provides controlled-scene images [78]. VDD-C comprises ~105k annotated diver images across pool and field settings [79,80]. UOT100 focuses on tracking with 104 sequences and 74K+ annotated frames [81,82]. More recently, RUOD [11] aggregates large-scale, diverse underwater data, addressing lighting/turbidity variations across 14K images and 74,903 instances. Table 1 compares these datasets with UWOD, which spans a wider range of imaging environments and class coverage than any individual source. Despite these advances, gaps remain in class breadth, environmental variety, and standardized benchmarks suitable for robust cross-model comparison.

**Table 1.** Comparison of UWOD with existing underwater object detection datasets. Images and instances are the counts published by each source. Redis. refers to whether the compiled imagery may be re-hosted by third parties, not to whether the data may be used for research.

| Dataset                                         | Images  | Instances | Cls.           | Environment                                                                         | Redis.         |
|-------------------------------------------------|---------|-----------|----------------|-------------------------------------------------------------------------------------|----------------|
| Sources integrated into UWOD                    |         |           |                |                                                                                     |                |
| Aquarium [78]                                   | 638     | 4821      | 7              | Controlled aquarium tanks                                                           | O <sup>a</sup> |
| Brackish [77]                                   | 14,674  | 35,565    | 6              | Temperate brackish strait; fixed camera, varying visibility                         | O <sup>b</sup> |
| DUO [74]                                        | 7782    | 74,515    | 4              | Open-sea farm, Zhangzi Island; URPC and UDD imagery                                 | R <sup>c</sup> |
| UTDAC2020 [56]                                  | 6461    | 46,685    | 4              | Open-sea farm, Zhangzi Island; URPC2020 Dalian contest                              | R <sup>c</sup> |
| UODD [75]                                       | 3194    | 19,212    | 3              | Open-sea farm, Zhangzi Island; low-contrast benthic scenes                          | R <sup>c</sup> |
| VDD-C [79,80]                                   | 104,833 | 167,278   | 1              | Swimming pools and Caribbean field sites; video-sourced                             | O <sup>d</sup> |
| UOT100 [81,82]                                  | 74,042  | 74,042    | – <sup>e</sup> | 104 video sequences from third-party online video                                   | N <sup>f</sup> |
| External comparators (not integrated into UWOD) |         |           |                |                                                                                     |                |
| RUOD [11]                                       | 14,000  | 74,903    | 10             | Aggregated multi-site; haze, colour cast, light interference                        | R <sup>c</sup> |
| UWOD (ours)                                     | 107,216 | 373,935   | 39             | Aquarium tanks, brackish water, open-sea farms, pools, tropical sites, online video | R              |

**Redis.:** O = openly redistributable; R = research use, redistribution restricted or not explicitly granted; N = not redistributable, research only<sup>a</sup> CC BY 4.0 Attribution 4.0 International. <sup>b</sup> CC BY-SA 4.0 Attribution-ShareAlike 4.0 International. <sup>c</sup> Publicly available for research, but an explicit third-party redistribution grant or open license for the underlying imagery was not verified. <sup>d</sup> CC BY-SA 3.0 US license via the University of Minnesota Digital Conservancy. <sup>e</sup> UOT100 is an object tracking benchmark; the tracked category varies by sequence, so “–” indicates that a fixed class count does not apply. <sup>f</sup> UOT100’s terms restrict use to non-commercial research and prohibit reproduction or redistribution of the dataset.

## 2.6. Recent Advances and Future Directions

Recent work explores architectures and data strategies tailored to underwater variability. Xu et al. [83] propose a scale-aware feature pyramid for marine targets. Synthetic data has been explored to alleviate annotation scarcity [57]. Transfer learning and domain adaptation remain central to improving robustness across conditions [37,55]. Zhou et al. [58] survey real-time UOD techniques for complex environments, and Zheng et al. [59] incorporate degraded priors for fish detection. Our work builds on this body of research by compiling a large, harmonized underwater dataset, benchmarking state-of-the-art detectors, and systematically analyzing transfer learning strategies specific to the underwater domain.

## 3. Datasets

The effectiveness of underwater object detection models relies on comprehensive, high-quality data. We introduce the Underwater Object Detection (UWOD) dataset, a large-scale compilation designed to address limitations of existing underwater datasets in size, diversity, and annotation quality. This section describes the data sources, construction steps (including deduplication and class harmonization), the annotation pipeline, a summary of the final splits, and a comparison to existing datasets.

### 3.1. Dataset Sources

UWOD amalgamates seven publicly available underwater datasets. Each source contributes distinct species, environments, and imaging conditions, increasing both diversity and robustness. To mitigate overlaps across sources, we used perceptual hashing (Image-Hash with pHash) [84] to identify near-duplicates. Unlike cryptographic hashes, pHash groups visually similar images even if resized or recompressed. We computed pHash for all images and removed near-duplicates, retaining the instance with the higher-quality annotations; all unique images were preserved. A summary appears in Table 2.

**Table 2.** Summary of datasets integrated into UWOD.

| Dataset          | Images (Total) | Images (Used) | Classes | Class List (Subset)                                         |
|------------------|----------------|---------------|---------|-------------------------------------------------------------|
| Aquarium [78]    | 638            | 638           | 7       | fish, jellyfish, penguin, puffin, shark, starfish, stingray |
| Brackish [76,77] | 14,674         | 14,674        | 6       | large fish, small fish, crab, jellyfish, shrimp, starfish   |
| DUO [74]         | 7782           | 2840          | 4       | echinus, holothurian, scallop, starfish                     |
| UTDAC2020 [56]   | 6461           | 6461          | 4       | echinus, holothurian, scallop, starfish                     |
| UODD [75]        | 3194           | 2300          | 3       | echinus, holothurian, scallop                               |
| VDD-C [79,80]    | 104,833        | 26,206        | 1       | person/diver                                                |
| UOT100 [81,82]   | 74,042         | 54,097        | *       | various marine species                                      |

\* UOT100 is an object tracking benchmark; a fixed class count does not apply.

Notes on the integrated datasets.

- Aquarium [78]: Controlled-environment images of multiple species; a strong starting point for object detection.
- Brackish [76,77]: Temperate/brackish waters with varying visibility, increasing environmental diversity.
- UTDAC2020 [56] and DUO [74]: Focus on *echinus*, *holothurian*, *scallop*, *starfish*, contributing key benthic classes.
- UODD [75]: Emphasizes diverse viewpoints, complex scenes, and low-contrast conditions.
- VDD-C [79,80]: Diver/person imagery; to balance categories we sampled ~25% of images.
- UOT100 [81,82]: Tracking-oriented sequences; we re-annotated frames to add unlabeled co-occurring objects where needed.

### 3.2. Dataset Construction

We followed a standardized integration protocol:

**Dataset review and preprocessing:** Each source underwent an annotation quality audit and license check. For annotation management and verification, we used Label Studio [85], an open-source data labeling platform that supports images and video, object detection (boxes, polygons, keypoints), ML-assisted labeling, and data exploration via a web UI.

**Class mapping and label harmonization:** We aligned source labels to a unified UWOD taxonomy. Overlapping classes were merged; novel classes were appended to the UWOD list and reindexed. This ensured consistent semantics across all sources.

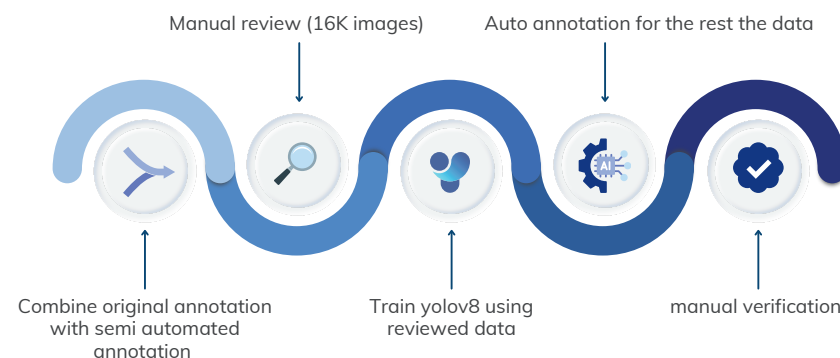
**Deduplication and selection:** We identified exact and near-duplicate images via pHash (256-bit, hash\_size 16), where near-duplicates are re-encoded or re-saved copies

of the same image that differ at file level but not visually. Images with identical hashes (Hamming distance 0) were treated as duplicates, the instance with the greater number of annotations was kept, with image resolution as a tie-break. Distance 0 was chosen as the most conservative setting, removing only copies that are visually indistinguishable while retaining every image that differs in content. When no duplicates were found, all unique images were retained.

**Integration:** After harmonization and deduplication, we merged images and labels into a single repository with standardized formats and metadata.

### 3.3. Annotation Pipeline

While most sources were fully annotated, tracking datasets (e.g., UOT100) often lacked labels for co-occurring objects. We applied the re-annotation pipeline in Figure 1.



**Figure 1.** UWOD re-annotation pipeline. Starting from seed labels, we add model-assisted proposals (YOLOv8), perform expert verification on a subset, retrain, auto-annotate the remaining data, and conduct a final human audit. This loop increases label completeness (especially for co-occurring objects) and standardizes annotation quality across sources.

Steps:

1. **Seed labels + semi-automated labels.** Train an initial YOLOv8 [42] model on verified data; generate predictions on under-labeled frames; merge with existing annotations.
2. **Manual review.** Expert annotators verified ~16,000 images, cross-checking original videos when applicable and performing occasional reverse image searches for class confirmation. Throughout the manual review stages, annotators and supervisors held short daily discussions in which ambiguous cases, such as heavily occluded, blurred, or turbidity-degraded instances and visually similar classes, were resolved by consensus, keeping labeling decisions consistent across annotators and batches. Each image passed through at least two review stages, with the second serving as a verification and correction pass over the existing labels.
3. **Retrain YOLOv8.** Use the verified set to retrain YOLOv8, improving proposal quality for the remaining data.
4. **Auto-annotate remaining data.** Apply the improved model to unlabeled/partially labeled images to accelerate coverage.
5. **Final verification.** Manually validate auto-labels; correct any errors to ensure consistency and quality across UWOD.

#### Quality of the Auto-Generated Labels

The labels produced in Step 1 are the raw output of the YOLOv8 model, so their quality prior to human verification is given by the model's agreement with the verified annotations. Evaluating the annotation-stage model on the held-out test split, and counting a proposal as correct when it matches a verified annotation at  $\text{IoU} \geq 0.5$  with the same class

label at the confidence threshold of 0.25 used during auto-annotation, the model attained a precision of 0.758 and a recall of 0.740 ( $F1 = 0.749$ ). Of the proposals presented to annotators, 57.0% were accepted unchanged, 18.9% required only geometric refinement, 1.4% required class correction, and 22.8% corresponded to no annotated object (Table 3).

These figures confirm that the model-in-the-loop stage substantially reduced annotation effort, since more than half of all proposals were accepted without modification. It also establishes that the human verification of Steps 2 and 5 were necessary rather than optional, since close to a quarter of proposals corresponded to no annotated object. The annotations used throughout this work are the corrected ones rather than the model output.

**Table 3.** Quality of the auto-generated labels prior to manual verification, measured against the verified annotations on the held-out test split. A proposal counts as correct when it matches a verified annotation at  $IoU \geq 0.5$  with the same class label at the confidence threshold of 0.25 used during auto-annotation.

| Metric                        | Value |
|-------------------------------|-------|
| Precision                     | 0.758 |
| Recall                        | 0.740 |
| F1                            | 0.749 |
| Accepted without modification | 57.0% |
| Required geometric refinement | 18.9% |
| Required class correction     | 1.4%  |
| No corresponding annotation   | 22.8% |

### 3.4. UWOD Dataset Overview

The final UWOD dataset comprises over 107,000 images across 39 classes, capturing a broad range of underwater scenes, species, and conditions. For video-sourced datasets UOT100 [81,82], we avoid random frame sampling for splits, since adjacent frames are near-identical and would otherwise leak across train/val/test and inflate metrics. Splits were instead assigned at the video level, with all frames from a given video kept within a single split. As video names identify the class, stratification is applied per class; classes with three or more videos have whole videos distributed across the splits, while classes with only one or two videos have the video divided contiguously in time rather than sampled at random. The same procedure was applied to the Brackish dataset [77], whose published split assigns frames at random [76], and therefore distributes each of its 89 source recordings across train, validation, and test sets; we reassigned its splits at the video level so that all frames of a recording remain within a single split. We use a stratified split with ~70% for training and the remainder for validation and testing (Table 4).

**Table 4.** UWOD splits and counts. We report images and instance annotations for train/val/test. Splits are stratified by source and class to preserve distributional diversity, with ~70% for training and the remainder for validation and testing.

| Subset     | Images  | Annotations |
|------------|---------|-------------|
| Train      | 75,209  | 262,475     |
| Validation | 15,582  | 56,434      |
| Test       | 16,425  | 55,026      |
| Total      | 107,216 | 373,935     |

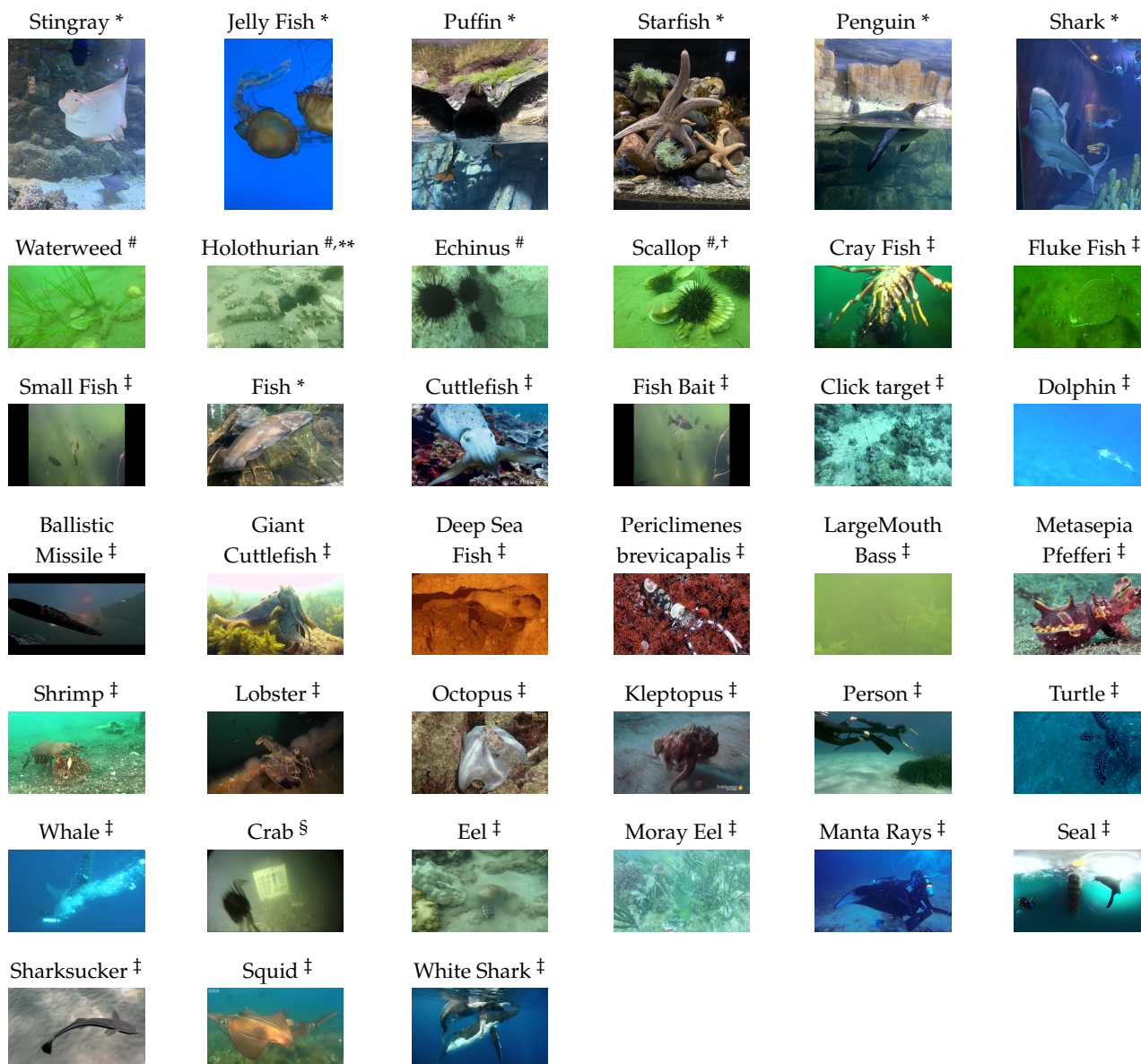
### 3.5. Dataset Diversity, Quality, and Comparison with Existing Datasets

UWOD spans 39 classes, including *fish*, *jellyfish*, *crab*, *shrimp*, *echinus*, *holothurian*, *scallop*, *starfish*, and others. Inclusion of the *person* class (divers) supports applications such as



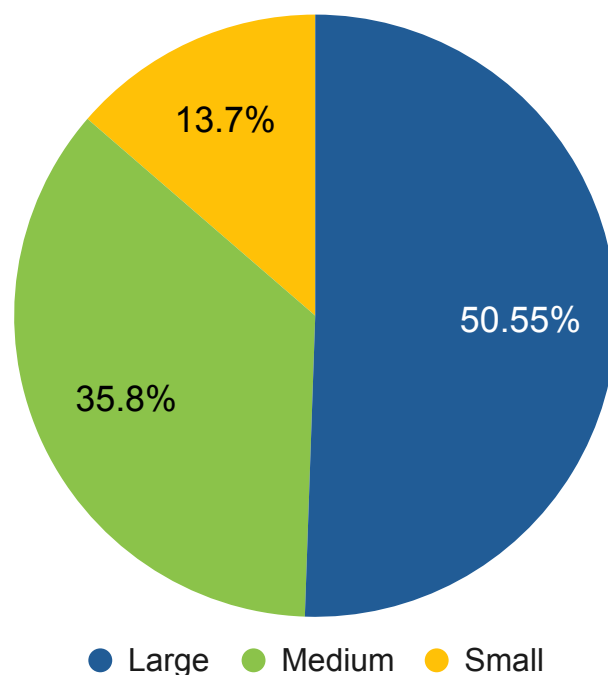
AUV navigation and diver assistance, for a more detailed breakdown of the classes, along with the respective number of images and instances for each, please refer to Table 5 while Figure 2 provides representative visual examples of each class.

Figure 3 shows the distribution of object sizes in the dataset: *large* instances ( $\geq 96 \times 96$  pixels, per the MS COCO definition) account for 50.55% of all annotations, *medium* instances ( $32 \times 32$  to  $96 \times 96$  pixels) for 35.79%, and *small* instances (under  $32 \times 32$  pixels) for the remaining 13.66%.



**Figure 2.** Representative UWOD class examples spanning pelagic, benthic, and human/diver categories. Samples are drawn from the integrated sources (see superscripts/notes). Visual variability in species, scale, turbidity, lighting, and viewpoints reflects real underwater conditions targeted by our benchmarks. Note: Sources—\* Aquarium [78]; \*\* UODD [75]; # UTDAC2020 [56]; † DUO [74]; ‡ UOT100 [81,82]; § Brackish [76,77].

In terms of image quality, we utilized the Underwater Color Image Quality Evaluation (UCIQE) metric [86] to classify the dataset into three groups.



**Figure 3.** Object size distribution in UWOD dataset.

**Table 5.** UWODdataset overview: class-wise image and annotation distribution.

| Class Name            | Instances | Images                   | Class Name                | Instances | Images |
|-----------------------|-----------|--------------------------|---------------------------|-----------|--------|
| Ballistic Missile     | 969       | 969                      | Moray Eel                 | 505       | 502    |
| Click Target          | 1151      | 1122                     | Octopus                   | 4020      | 3933   |
| Crab                  | 17,566    | 6984                     | Penguin                   | 387       | 58     |
| Cray Fish             | 590       | 515                      | Periclimenes brevicapalis | 361       | 361    |
| Cuttlefish            | 278       | 278                      | Person                    | 101,135   | 42,098 |
| Deep Sea Fish         | 1002      | 1002                     | Puffin                    | 284       | 66     |
| Dolphin               | 934       | 933                      | Scallop                   | 9211      | 2018   |
| Echinus               | 52,899    | 9504                     | Seal                      | 543       | 453    |
| Eel                   | 5092      | 4005                     | Shark                     | 5373      | 2762   |
| Fish                  | 80,134    | 17,081                   | Sharksucker               | 1608      | 1229   |
| Fish Bait             | 1975      | 1887                     | Shrimp                    | 1050      | 1050   |
| Fluke Fish            | 1000      | 1000                     | Small Fish                | 10,815    | 2513   |
| Giant Cuttlefish      | 1878      | 1006                     | Squid                     | 866       | 330    |
| Holothurian           | 12,531    | 5461                     | Starfish                  | 29,065    | 13,160 |
| Jelly Fish            | 2804      | 1421                     | Stingray                  | 903       | 789    |
| Kleptopus             | 2682      | 2653                     | Turtle                    | 5190      | 5159   |
| LargeMouth Bass       | 964       | 687                      | Waterweed                 | 3925      | 2781   |
| Lobster               | 743       | 742                      | Whale                     | 3075      | 3008   |
| Manta Rays            | 4056      | 3777                     | White Shark               | 1074      | 563    |
| Metasepia Pfefferi    | 375       | 375                      |                           |           |        |
| Total Images: 107,216 |           | Total Instances: 373,935 |                           |           |        |

Note: The source data included overlapping fish-related classes inherited from different datasets; we merged Large Fish and Fish into a single Fish class, defining Fish as medium-to-large-bodied fish, Small Fish as small-bodied fish (e.g., schooling or juvenile individuals), Deep Sea Fish as fish annotated in deep/benthic footage exhibiting deep-sea morphology, and Fluke Fish as flatfish species (e.g., flounder, sole).

UCIQE quantifies underwater image degradation as a weighted linear combination of chroma dispersion, luminance contrast, and average saturation, computed in the CIELab color space as stated in the following equation:

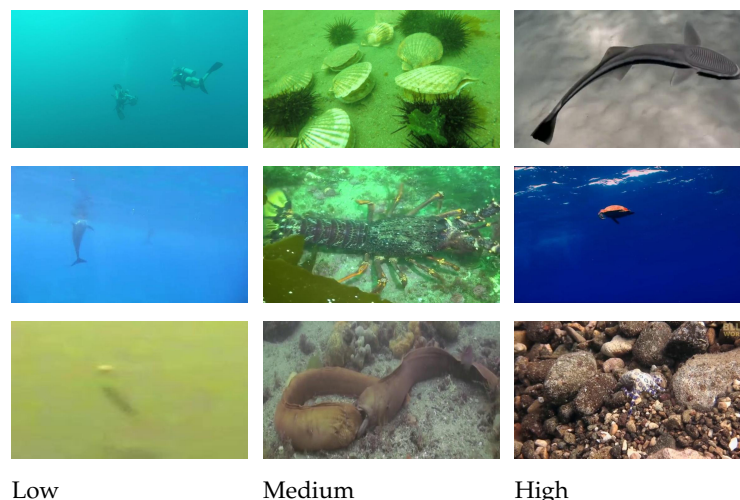
$$\text{UCIQE} = c_1 \cdot \sigma_c + c_2 \cdot \text{con}_l + c_3 \cdot \mu_s, \quad (1)$$

where  $\sigma_c$  is the standard deviation of the chroma channel,  $\text{con}_l$  is the contrast of the luminance channel, and  $\mu_s$  is the average saturation. Given an underwater image in the RGB color space, we convert it to CIELab and normalize the  $L$ ,  $a$ , and  $b$  channels. The chroma at each pixel is defined as  $\text{chroma} = \sqrt{a^2 + b^2}$ , and  $\sigma_c$  is its standard deviation over the image. The luminance contrast  $\text{con}_l$  is computed as the difference between the 99th and 1st percentile of the luminance channel  $L$ , which is more robust to outlier pixels than the full dynamic range. The saturation at each pixel is  $s = \text{chroma} / L$  (set to 0 where  $L = 0$ ), and  $\mu_s$  is its mean over the image. Following the original formulation [86], we use the trained coefficients  $c_1 = 0.4680$ ,  $c_2 = 0.2745$ , and  $c_3 = 0.2576$ . UCIQE was computed on the original, full-resolution images, without any resizing prior to computation.

We classify images into three quality groups based on the following UCIQE value thresholds:

- High quality:  $\text{UCIQE} \geq 0.8$  (less than 17% of images).
- Medium quality:  $0.5 \leq \text{UCIQE} < 0.8$  ( $\approx 30\%$  of images).
- Low quality:  $\text{UCIQE} < 0.5$  (more than 50% of images).

Figure 4 shows representative sample images from each of the three quality groups (High, Medium, and Low).

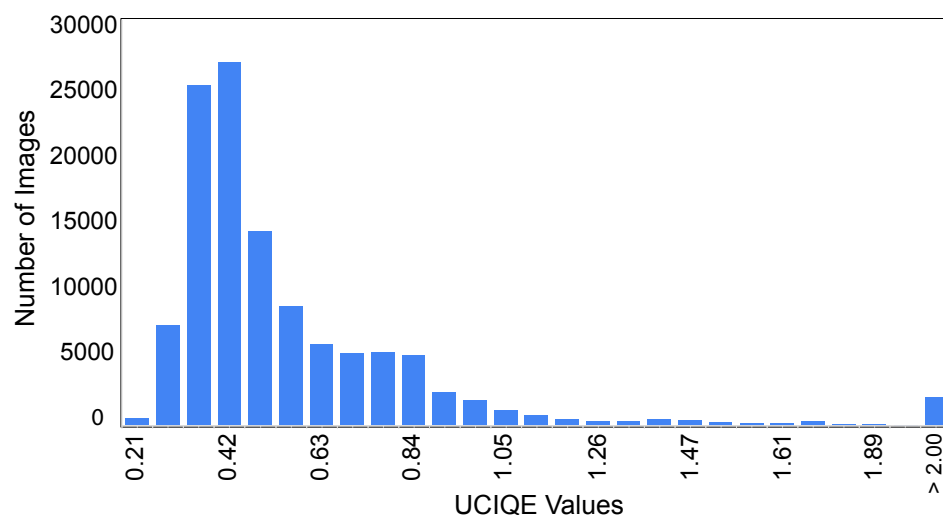


**Figure 4.** Sample images from the dataset categorized by UCIQE values: **(High)** ( $\text{UCIQE} \geq 0.8$ ), **(Medium)** ( $0.5 \leq \text{UCIQE} < 0.8$ ), and **(Low)** ( $\text{UCIQE} < 0.5$ ).

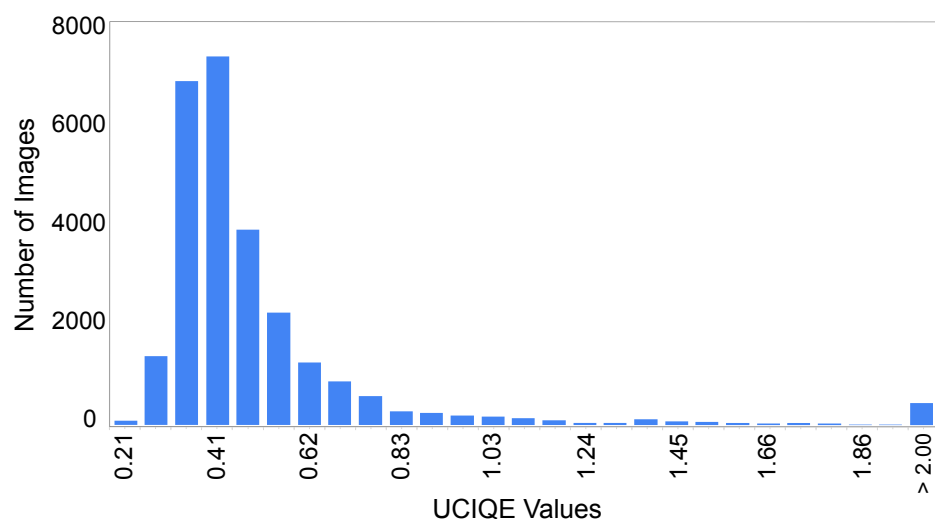
We used two different methodologies to identify robust cutoff boundaries for the UCIQE quality thresholds. First, we applied multi-level Otsu thresholding [87], to the UCIQE scores of the UWOD dataset. We adopted multi-Otsu as the primary method because it computes a globally optimal partition of the score histogram by maximizing between-class variance, and is deterministic and reproducible, unlike clustering methods that depend on initialization. Applied to our data, multi-Otsu identified boundaries at 0.548 and 0.846. To verify that this result reflects genuine structure in the data rather than an artifact of the method, we independently applied  $k$ -means clustering [88] ( $k = 3$ ), a standard unsupervised discretization technique for converting a continuous score into categorical bands [89], to the same distribution.  $K$ -means converged to boundaries at 0.552 and 0.849, within 0.004 of the multi-Otsu result, confirming that the two independently motivated

methodologies agree closely and that both closely approximate 0.5 and 0.8. As a result, the classification is not sensitive to the specific clustering method used to derive it. We adopted these round values as the practical low/medium and medium/high cutoffs used throughout this work, for ease of interpretation. This clustering-based derivation is further consistent with the overall shape of UWOD's UCIQE distribution Figures 5 and 6. UCIQE is a relative colorfulness–contrast measure and is used here only as a coarse stratification of imaging conditions, not as an absolute quality scale.

As shown by these classifications, low-quality images make up more than half of the training dataset. This is due to the challenges posed by underwater photography, where the water medium causes light scattering, thereby impacting the overall image color quality.

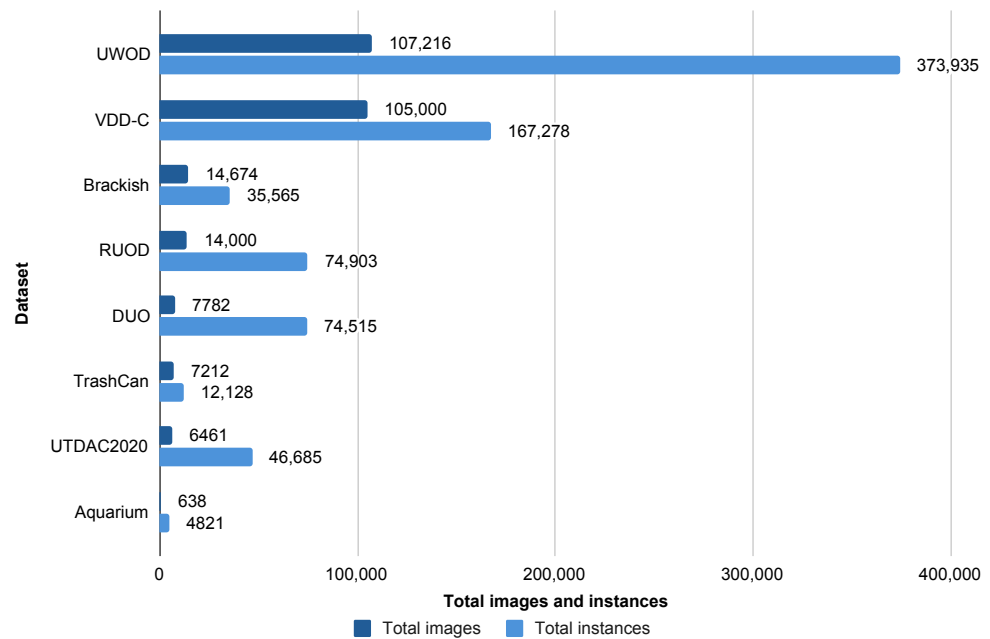


**Figure 5.** This histogram shows the number of train images corresponding to UCIQE values, highlighting the image color quality distribution.



**Figure 6.** A histogram mapping the number of test images to their corresponding UCIQE values.

This variability in turbidity, lighting, and depth reflects realistic underwater conditions, aiding robustness and generalization in trained models. Additionally, relative to UTDAC2020 [56], DUO [74], Brackish [76,77], Aquarium [78], and VDD-C [79,80], UWOD provides substantially more images and annotations (Figure 7), offering a stronger basis for training and benchmarking.



**Figure 7.** Scale comparison of UWOD versus existing underwater datasets (images and instances). UWOD increases both dataset size and instance density, enabling stronger baselines and more reliable evaluation across varied turbidity, lighting, and scene complexity.

### 3.6. Dataset Availability and Ethical Considerations

Due to upstream licensing, particularly for UOT100 [81,82], we cannot redistribute the compiled UWOD dataset in full. We comply with all original licenses and use UWOD strictly for research purposes consistent with those terms. Instead of releasing the compiled data, we ensure reproducibility and continuity as follows: (1) we provide a detailed reconstruction protocol (sources, filtering, deduplication thresholds, class mapping); (2) we release trained model weights and the automated annotation pipeline.

This approach enables researchers to (a) apply our models to new underwater imagery for immediate use, (b) rebuild a comparable dataset by acquiring the original sources and following our protocol, and (c) benefit from domain-specific knowledge transfer without license conflicts.

## 4. Materials and Methods

This section describes the models, experimental design, datasets, implementation details, and evaluation metrics used in our study of underwater object detection.

### 4.1. Object Detection Models

We evaluate nine representative detectors spanning one-stage, two-stage, and transformer-based paradigms, capturing characteristics of underwater imagery (scale variation, turbidity, and low contrast): YOLOv8 [42] with a C2f-based backbone including an SPPF module, YOLOv7 [43] with an ELAN-based backbone, YOLOv5 [44]; a widely adopted one-stage detector emphasizing speed–accuracy trade-offs, EfficientDet [46]; a scalable detector leveraging EfficientNet backbones and bi-directional feature pyramids, FCOS [45]; an anchor-free fully convolutional one-stage detector, YOLOX [47]; an anchor-free one-stage detector with a decoupled head, RT-DETR [48]; a real-time transformer-based detector, Faster R-CNN [16]; a two-stage detector that combines region proposals with classification and regression and SSD [49]; a one-stage detector with multi-scale feature



maps. Collectively, these models [16,42–49] represent widely adopted, state-of-the-art approaches and enable a comprehensive comparison in underwater settings.

#### 4.2. Experimental Design

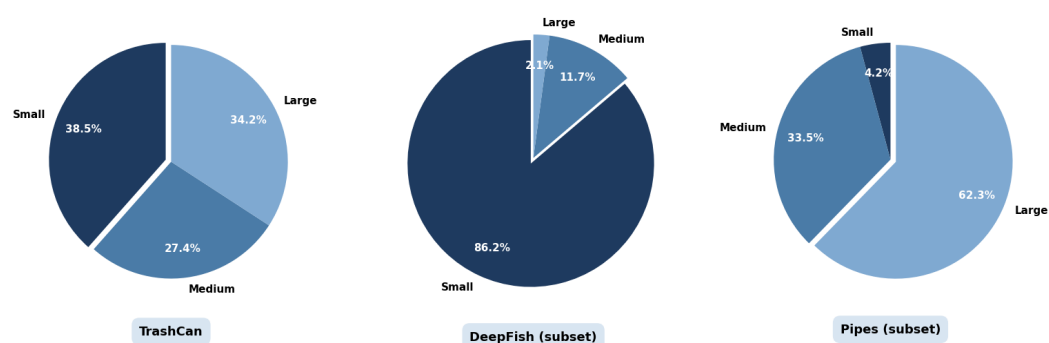
We design three complementary experiments to assess the value of UWOD for underwater object detection.

- (1) **Baselines on UWOD.** We train and evaluate YOLOv8 [42], YOLOv7 [43], FCOS [45], EfficientDet [46], YOLOX [47], RT-DETR [48], Faster R-CNN [16], and SSD [49] on UWOD to establish robust, comparable baselines.
- (2) **Robustness analysis.** We analyze the performance of YOLOv8 [42], YOLOv7 [43], YOLOv5 [44], FCOS [45], EfficientDet [46], and YOLOX [47] trained on UWOD across different object sizes (small, medium, and large) and image quality UCIQE-based splits (low/medium/high) to assess their robustness under varying underwater conditions.
- (3) **Transfer learning to small-scale underwater datasets.** We compare four pre-training trajectories to quantify domain benefits under limited labels:
  - (S1) COCO [36] → small-scale underwater dataset,
  - (S2) UWOD → small-scale underwater dataset,
  - (S3) COCO [36] → UWOD → small-scale underwater dataset,
  - (S4) small-scale underwater dataset only (no pre-training; baseline).

Unless noted otherwise, results use single-model, single-scale inference following each method's reference evaluation protocol [16,42,43,47–49].

#### 4.3. Small-Scale Underwater Datasets

To study transfer learning under realistic low-resource conditions, we include three small-scale datasets summarized in Figures 8 and 9: TrashCan [90], Pipes [91], and DeepFish [92]. We deliberately evaluate curated subsets of Pipes [91] and DeepFish [92] to simulate scenarios where acquiring and annotating large volumes of underwater imagery is expensive and logistically difficult (e.g., specialized platforms, limited dive/ROV time, adverse visibility), which is common for this challenging environment. This subset protocol also prevents ceiling effects on simpler tasks (e.g., Pipes) while controlling computational budget.

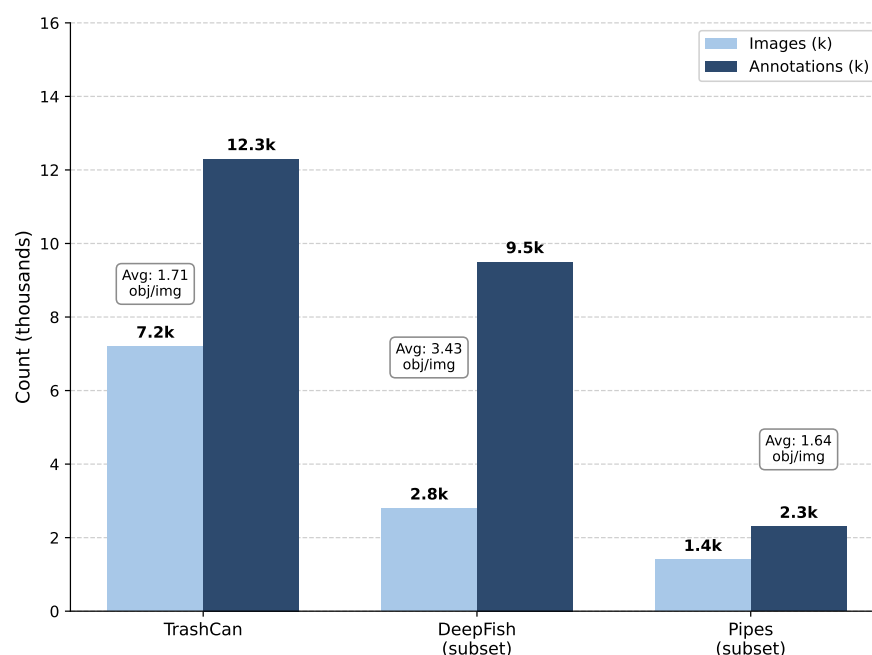


**Figure 8.** Size distribution of the small-scale datasets used to evaluate transfer learning under data-scarce conditions, computed by image-area thresholds (small < 2%, medium 2–10%, large > 10%). Pipes [91] and DeepFish [92] are evaluated on curated subsets to emulate limited-availability settings and control compute.

We characterize dataset difficulty by object density (average instances per image) and normalized object size. Size categories follow percentage of image area: small < 2%, medium 2–10%, large > 10%. Under this scheme, TrashCan [90] exhibits a balanced size distribution (38.5% small, 27.4% medium, 34.2% large; 1.71 objects/image). DeepFish



(subset) [92] is the most challenging, with predominantly small instances (86.2% < 2% area) and the highest density (3.43 objects/image). Pipes (subset) [91] is the simplest, dominated by large instances (62.3% > 10% area) and fewer objects per image (1.64).

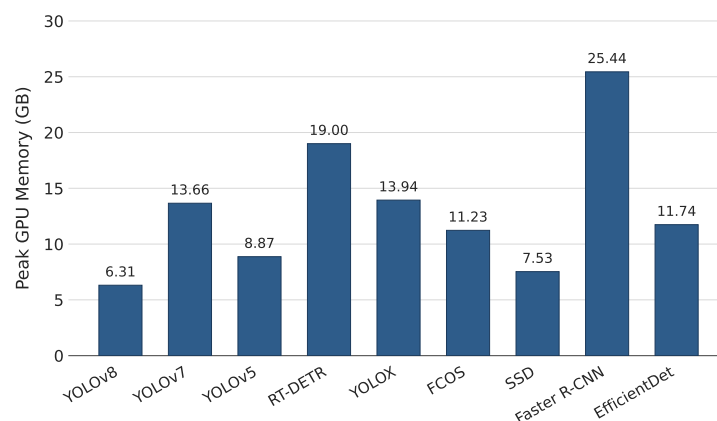


**Figure 9.** Total images, total annotations, and average objects per image for the small-scale datasets used to evaluate transfer learning under data-scarce conditions. Pipes [91] and DeepFish [92] are evaluated on curated subsets to emulate limited-availability settings and control compute.

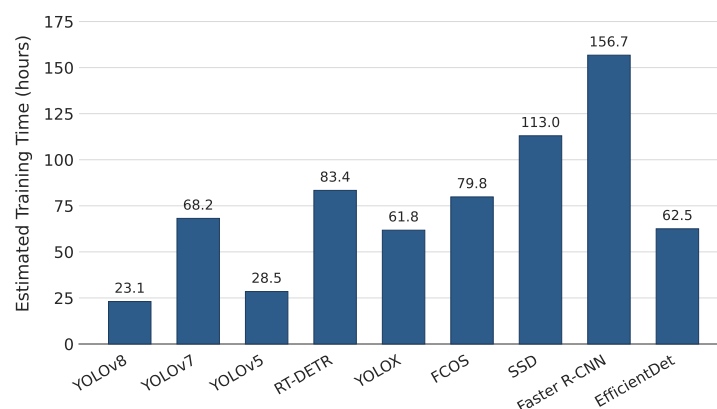
#### 4.4. Implementation Details and Training Environment

To maintain comparability, we standardize core hyperparameters across methods [16,42–49] where feasible: batch size of 16 and initial learning rate of 0.01. For YOLOv8 [42], we use the YOLOv8m configuration with its default C2f-based backbone (including SPPF;  $640 \times 640$  input resolution) and the default SGD optimizer. For YOLOv7 [43], we use the configuration with an ELAN backbone ( $640 \times 640$  input resolution) and the default SGD optimizer. For YOLOv5 [44], we use the YOLOv5m configuration with its default C3-based backbone (including SPPF;  $640 \times 640$  input resolution) and the default SGD optimizer. For YOLOX [47], we use the YOLOX-m configuration with its default CSPDarknet backbone and PAFPN neck ( $640 \times 640$  input resolution) and the default SGD optimizer. For RT-DETR [48], we use a ResNet-101 backbone ( $640 \times 640$  input resolution) with the default AdamW optimizer. For FCOS [45] and Faster R-CNN [16], we use a ResNet-50 backbone with a Feature Pyramid Network ( $800 \times 1333$  input resolution) and the default SGD optimizer. For SSD [49], we use the SSD512 configuration with VGG-16 backbone ( $512 \times 512$  input resolution), and we lowered the learning rate to  $1 \times 10^{-4}$  for training stability; a value consistent with SSD’s own intermediate training rate [49], since the default  $1 \times 10^{-3}$  produced NaN losses. For EfficientDet [46], we use an EfficientNet-B1 backbone with a BiFPN ( $640 \times 640$  input resolution) and the default SGD optimizer. For large-scale pre-training on COCO [36] and UWOD, we train for approximately 739K iterations per model. Each small-scale underwater dataset (TrashCan [90], DeepFish [92], Pipes [91]) is fine-tuned for 100 epochs. For the 10%, 50%, and 100% training data conditions in Section 5.4, we report mean  $\pm$  SD across three random seeds (0, 42, 70), varying weight initialization and training time stochasticity with all other settings fixed. All experiments run on a single NVIDIA A100 GPU (NVIDIA Corporation, Santa Clara, CA, USA; 80 GB). To support reproducibility, we additionally report each detector’s computational require-

ments: peak GPU memory was measured using PyTorch’s CUDA (PyTorch 2.11.0, built with CUDA 12.8 support) memory-profiling utilities, and training time. Figures 10 and 11 report these values for each of the nine evaluated detectors. Peak memory ranged from 6.31 GB (YOLOv8) to 25.44 GB (Faster R-CNN); estimated full training time ranged from approximately 23 h (YOLOv8) to approximately 157 h (Faster R-CNN). Faster R-CNN required both the most memory and the longest training time among the nine methods, whereas the YOLO-family detectors (YOLOv8, YOLOv7, YOLOv5) were the least resource-intensive to train.



**Figure 10.** Peak GPU memory required to train each of the nine evaluated detectors on UWOD, measured on a single NVIDIA A100 GPU (80 GB).

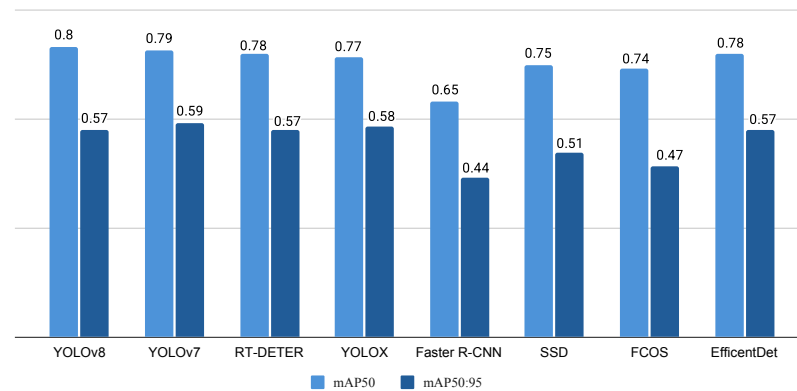


**Figure 11.** Estimated total training time for each detector.

## 5. Evaluation and Analysis

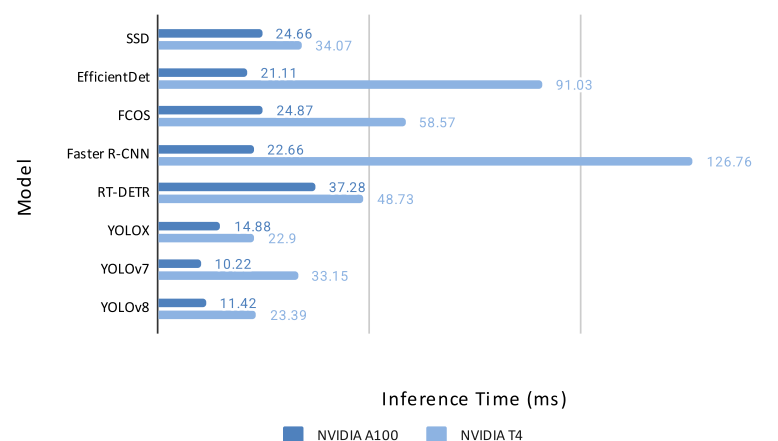
### 5.1. Performance of State-of-the-Art Object Detection Models

Figure 12 illustrates the performance of various state-of-the-art object detection algorithms on the UWOD dataset, evaluated using the metrics  $mAP_{50}$  and  $mAP_{50:95}$ . YOLOv8 [42] and YOLOv7 [43] stand out with strong accuracy and efficient processing, making them top contenders for high-performance tasks. RT-DETR [48], FCOS [45], EfficientDet [46] and YOLOX [47] provide a balanced approach. Meanwhile, Faster R-CNN [16] shows the lowest accuracy among the models. While our results indicate that Faster R-CNN [16] has the lowest accuracy among the models tested on the UWOD dataset, it is important to consider that its performance may improve with extensive hyperparameter tuning. However, exploring such optimizations is beyond the scope of our current study.



**Figure 12.** The performance of a state-of-the-art object detection algorithm on the UWOD dataset.

Figure 13 presents a comprehensive comparison of model inference times across different hardware configurations (NVIDIA A100 and NVIDIA T4). The analysis reveals significant variations in computational efficiency among the evaluated models. On the NVIDIA A100, YOLOv7 achieves the fastest inference time (10.22 ms), followed closely by YOLOv8 (11.42 ms) and YOLOX (14.88 ms), confirming the YOLO-based detectors as the most computationally efficient on this hardware. EfficientDet (21.11 ms), Faster R-CNN (22.66 ms), SSD (24.66 ms), and FCOS (24.87 ms) exhibit broadly comparable, intermediate inference times, whereas RT-DETR records the slowest inference on the A100 at 37.28 ms, consistent with the added computational cost of its transformer-based, ResNet-101 backbone. On the NVIDIA T4, however, the relative ranking shifts substantially: YOLOX becomes the fastest model (22.9 ms), narrowly ahead of YOLOv8 (23.39 ms), while YOLOv7's inference time increases considerably to 33.15 ms, placing it only marginally ahead of SSD (34.07 ms). RT-DETR (48.73 ms) and FCOS (58.57 ms) increase further, while EfficientDet (91.03 ms) and Faster R-CNN (126.76 ms) display the steepest slowdowns among the evaluated models, with Faster R-CNN's inference time increasing more than fivefold between the two platforms. Considered jointly with the accuracy results reported in Figure 12, these findings confirm that YOLOv8 and YOLOv7 offer the strongest overall accuracy–efficiency trade-off among the evaluated detectors, combining the highest mAP<sub>50</sub> and mAP<sub>50:95</sub> scores with the lowest inference times on both platforms, while RT-DETR and EfficientDet attain comparable accuracy at substantially higher inference cost, particularly on the T4, and Faster R-CNN records both the lowest accuracy overall and, on the T4, the slowest inference time as well. These timing comparisons highlight the trade-offs between model accuracy, computational efficiency, and hardware sensitivity, providing crucial insights for selecting appropriate models for real-time, resource-constrained underwater deployment scenarios such as AUVs and ROVs.



**Figure 13.** Model inference time comparison across different hardware platforms.

### 5.2. Class-Wise Performance Analysis

While Section 5.1 reports aggregate  $mAP_{50}$  and  $mAP_{50:95}$  across the full UWOD test set, dataset-level averages can mask substantial variation in per-class performance, particularly given UWOD's long-tailed class distribution Table 5: instance counts span more than two orders of magnitude, from 101,135 for *Person* and 80,134 for *Fish* down to 278 for *Cuttlefish* and 284 for *Puffin*. To directly assess whether this imbalance is reflected in detection performance, we report class-wise  $mAP_{50}$ ,  $mAP_{50:95}$ , and R for YOLOv7 [43] and YOLOv8 [42], the two top-performing detectors identified in Section 5.1, across all 39 UWOD classes. Table 6 orders classes by descending instance count to highlight the relationship between class frequency and detection performance.

**Table 6.** Class-wise  $mAP_{50}$ ,  $mAP_{50:95}$ , and R for YOLOv7 and YOLOv8 on the UWOD test set, ordered by descending instance count, see Table 5 for class-wise image and annotation distribution.

| Class                     | YOLOv7     |               |        | YOLOv8     |               |         |
|---------------------------|------------|---------------|--------|------------|---------------|---------|
|                           | $mAP_{50}$ | $mAP_{50:95}$ | R      | $mAP_{50}$ | $mAP_{50:95}$ | R       |
| Person                    | 0.841      | 0.494         | 0.766  | 0.962      | 0.449         | 0.69    |
| Fish                      | 0.775      | 0.508         | 0.718  | 0.705      | 0.355         | 0.478   |
| Echinus                   | 0.138      | 0.0749        | 0.45   | 0.653      | 0.313         | 0.511   |
| Starfish                  | 0.379      | 0.332         | 0.396  | 0.686      | 0.458         | 0.531   |
| Crab                      | 0.977      | 0.739         | 0.93   | 0.976      | 0.759         | 0.934   |
| Holothurian               | 0.0419     | 0.0244        | 0.15   | 0.606      | 0.292         | 0.463   |
| Small Fish                | 0.955      | 0.66          | 0.934  | 0.965      | 0.699         | 0.959   |
| Scallop                   | 0.0759     | 0.0478        | 0.397  | 0.816      | 0.453         | 0.629   |
| Shark                     | 0.789      | 0.657         | 0.763  | 0.662      | 0.391         | 0.502   |
| Turtle                    | 0.943      | 0.708         | 0.84   | 0.859      | 0.684         | 0.794   |
| Eel                       | 0.445      | 0.194         | 0.41   | 0.397      | 0.177         | 0.371   |
| Manta Rays                | 0.5        | 0.158         | 0.426  | 0.36       | 0.126         | 0.109   |
| Octopus                   | 0.904      | 0.572         | 0.828  | 0.878      | 0.597         | 0.807   |
| Waterweed                 | 0.0597     | 0.0299        | 0.0366 | 0.145      | 0.076         | 0.0905  |
| Whale                     | 0.89       | 0.602         | 0.773  | 0.836      | 0.586         | 0.8     |
| Jelly Fish                | 0.935      | 0.581         | 0.908  | 0.909      | 0.569         | 0.883   |
| Kleptopus                 | 0.998      | 0.885         | 0.988  | 0.995      | 0.895         | 1       |
| Fish Bait                 | 0.967      | 0.65          | 0.917  | 0.947      | 0.671         | 0.906   |
| Giant Cuttlefish          | 0.986      | 0.853         | 0.989  | 0.982      | 0.845         | 0.989   |
| Sharksucker               | 0.163      | 0.112         | 0.0295 | 0.289      | 0.0997        | 0.00261 |
| Click target              | 0.925      | 0.713         | 0.923  | 0.925      | 0.743         | 0.923   |
| White Shark               | 0.88       | 0.748         | 0.886  | 0.945      | 0.8           | 0.953   |
| Shrimp                    | 0.99       | 0.772         | 0.985  | 0.991      | 0.813         | 0.985   |
| Deep Sea Fish             | 0.997      | 0.944         | 1      | 0.995      | 0.956         | 1       |
| Fluke Fish                | 0.991      | 0.818         | 0.979  | 0.995      | 0.866         | 0.993   |
| Ballistic Missile         | 0.995      | 0.965         | 0.993  | 0.995      | 0.972         | 0.993   |
| LargeMouth Bass           | 0.864      | 0.671         | 0.775  | 0.848      | 0.695         | 0.745   |
| Dolphin                   | 0.974      | 0.658         | 0.931  | 0.961      | 0.689         | 0.924   |
| Stingray                  | 0.858      | 0.486         | 0.837  | 0.822      | 0.352         | 0.718   |
| Squid                     | 0.727      | 0.306         | 0.694  | 0.804      | 0.296         | 0.697   |
| Lobster                   | 0.475      | 0.382         | 0.349  | 0.292      | 0.256         | 0.195   |
| Cray Fish                 | 0.935      | 0.425         | 0.832  | 0.932      | 0.484         | 0.909   |
| Seal                      | 0.778      | 0.407         | 0.72   | 0.876      | 0.505         | 0.722   |
| Moray Eel                 | 0.998      | 0.859         | 1      | 0.995      | 0.896         | 1       |
| Penguin                   | 0.51       | 0.237         | 0.75   | 0.462      | 0.211         | 0.694   |
| Metasepia Pfefferi        | 0.997      | 0.967         | 1      | 0.995      | 0.985         | 1       |
| Periclimenes brevicapalis | 0.996      | 0.982         | 1      | 0.995      | 0.993         | 1       |
| Puffin                    | 0.536      | 0.266         | 0.571  | 0.752      | 0.352         | 0.622   |
| Cuttlefish                | 0.998      | 0.994         | 1      | 0.995      | 0.995         | 1       |
| All                       | 0.79       | 0.59          | 0.74   | 0.8        | 0.57          | 0.73    |

Table 6 reveals that class frequency is not uniformly predictive of detection difficulty. The ten rarest classes achieve substantially higher average  $mAP_{50:95}$  (0.597) compared to

the ten most frequent classes (0.485), suggesting that visual distinctiveness rather than sample size determines detection performance. Classes with distinctive features—such as *Metasepia pfefferi* ( $mAP_{50:95} = 0.985$ ), Cuttlefish ( $mAP_{50:95} = 0.995$ ), and *Periclimenes brevicapalis* ( $mAP_{50:95} = 0.993$ )—achieve near-perfect performance despite low instance counts. Conversely, some frequent classes like Holothurian ( $mAP_{50:95} = 0.292$ ) and Waterweed ( $mAP_{50:95} = 0.137$ ) show lower performance due to visual ambiguity. Notably, rare classes with challenging visual characteristics (Penguin:  $mAP_{50:95} = 0.211$ ; Lobster:  $mAP_{50:95} = 0.256$ ) perform poorly, confirming that visual properties matter more than frequency in long-tailed distributions.

### 5.3. Performance Analysis Based on Object Size and Image Quality

We analyze YOLOv8 [42], YOLOv7 [43], YOLOv5 [44], YOLOX [47], FCOS [45], and EfficientDet [46] trained on UWOD, comparing performance by object size and image quality.

**Object size.** Using COCO-style size ranges  $mAP_{50:95}^{S/M/L}$ , YOLOv7 achieves the highest performance on small objects ( $mAP_{50:95}^S = 0.297$ ,  $AR_{50:95}^S = 0.503$ ), while YOLOv8 leads on medium ( $mAP_{50:95}^M = 0.580$ ,  $AR_{50:95}^M = 0.708$ ) and large objects ( $mAP_{50:95}^L = 0.784$ ,  $AR_{50:95}^L = 0.854$ ), as shown in Table 7. YOLOv5 demonstrates strong and consistent performance across all object sizes, ranking third in mAP for small, medium, and large objects. YOLOX shows moderate performance, outperforming FCOS and EfficientDet on medium objects, and outperforming FCOS but trailing EfficientDet on large objects. FCOS generally ranks lowest across sizes, while EfficientDet is low on small and medium objects but exceeds YOLOX on large objects.

**Image quality.** Using the UCIQE-based splits (low/medium/high), YOLOv8 attains the highest  $mAP_{50:95}$  across all quality levels (0.729, 0.733, and 0.771 for low, medium, and high quality, respectively). For  $mAP_{50}$ , YOLOv7 is best on medium-quality images (0.899), while YOLOv8 leads on low (0.905) and high quality (0.924) images, as summarized in Table 8. Notably, YOLOX attains competitive  $mAP_{50}$  scores (0.857, 0.851, and 0.809 for low, medium, and high quality, respectively), ranking fourth on medium- and high-quality images, ahead of FCOS and EfficientDet, though fifth on low-quality images, behind FCOS. Its  $mAP_{50:95}$ , however, is the lowest of all six models on low-quality images (0.556) and second-lowest on medium- and high-quality images (0.587 and 0.584), ahead of only EfficientDet. This yields the largest  $mAP_{50}$ – $mAP_{50:95}$  gap among the evaluated models at every quality level (0.301, 0.264, and 0.225 for low, medium, and high quality, respectively), indicating that YOLOX localizes objects reasonably well at the lenient 50% IoU threshold but its precision degrades sharply under stricter IoU criteria. FCOS and EfficientDet generally underperform compared to the top methods across quality levels.

**Table 7.** Performance comparison of YOLOv8, YOLOv7, YOLOv5, YOLOX, FCOS, and EfficientDet on UWOD based on object size. YOLOv7 achieves the best performance for small objects, while YOLOv8 excels for medium and large objects.

| Model        | $mAP_{50:95}^S$ | $AR_{50:95}^S$ | $mAP_{50:95}^M$ | $AR_{50:95}^M$ | $mAP_{50:95}^L$ | $AR_{50:95}^L$ |
|--------------|-----------------|----------------|-----------------|----------------|-----------------|----------------|
| YOLOv7       | <b>0.297</b>    | <b>0.503</b>   | 0.570           | 0.698          | 0.770           | 0.842          |
| YOLOv8       | 0.279           | 0.501          | <b>0.580</b>    | <b>0.708</b>   | <b>0.784</b>    | <b>0.854</b>   |
| YOLOv5       | 0.257           | 0.464          | 0.512           | 0.652          | 0.706           | 0.786          |
| EfficientDet | 0.164           | 0.231          | 0.377           | 0.502          | 0.612           | 0.699          |
| FCOS         | 0.198           | 0.296          | 0.357           | 0.467          | 0.510           | 0.626          |
| YOLOX        | 0.180           | 0.339          | 0.414           | 0.564          | 0.604           | 0.692          |

Bold values denote the best performance in each column.

**Table 8.** Performance comparison of YOLOv8, YOLOv7, YOLOv5, YOLOX, FCOS, and EfficientDet on UWOD based on image quality. YOLOv8 achieves the best overall performance across all image quality categories, with particularly strong results across low-, medium-, and high-quality images.

| Model        | Low Quality       |                      | Medium Quality    |                      | High Quality      |                      |
|--------------|-------------------|----------------------|-------------------|----------------------|-------------------|----------------------|
|              | mAP <sub>50</sub> | mAP <sub>50:95</sub> | mAP <sub>50</sub> | mAP <sub>50:95</sub> | mAP <sub>50</sub> | mAP <sub>50:95</sub> |
| YOLOv7       | 0.897             | 0.690                | <b>0.899</b>      | 0.711                | 0.919             | 0.749                |
| YOLOv8       | <b>0.905</b>      | <b>0.729</b>         | 0.897             | <b>0.733</b>         | <b>0.924</b>      | <b>0.771</b>         |
| YOLOv5       | 0.880             | 0.639                | 0.886             | 0.665                | 0.896             | 0.684                |
| EfficientDet | 0.832             | 0.584                | 0.772             | 0.572                | 0.716             | 0.547                |
| FCOS         | 0.862             | 0.603                | 0.835             | 0.602                | 0.807             | 0.604                |
| YOLOX        | 0.857             | 0.556                | 0.851             | 0.587                | 0.809             | 0.584                |

Bold values denote the best performance in each column.

**Overall.** when trained on UWOD, YOLOv7 is most effective for small objects, whereas YOLOv8 delivers the strongest accuracy on medium/large objects and across varying image qualities. YOLOv5 provides consistent, competitive performance across different conditions, ranking third by both object size and image quality. By object size, FCOS and EfficientDet show the lowest overall performance, with the caveat that EfficientDet exceeds YOLOX on large objects. By image quality, however, YOLOX and EfficientDet are instead the two lowest-performing models on mAP<sub>50:95</sub> at every quality level, with FCOS ranking fourth throughout rather than lowest. YOLOX demonstrates a notable trade-off between mAP<sub>50</sub> and mAP<sub>50:95</sub>: Its mAP<sub>50</sub> remains competitive with FCOS and ahead of EfficientDet, while its mAP<sub>50:95</sub> trails both YOLOv5 and FCOS across all quality levels (see Tables 7 and 8).

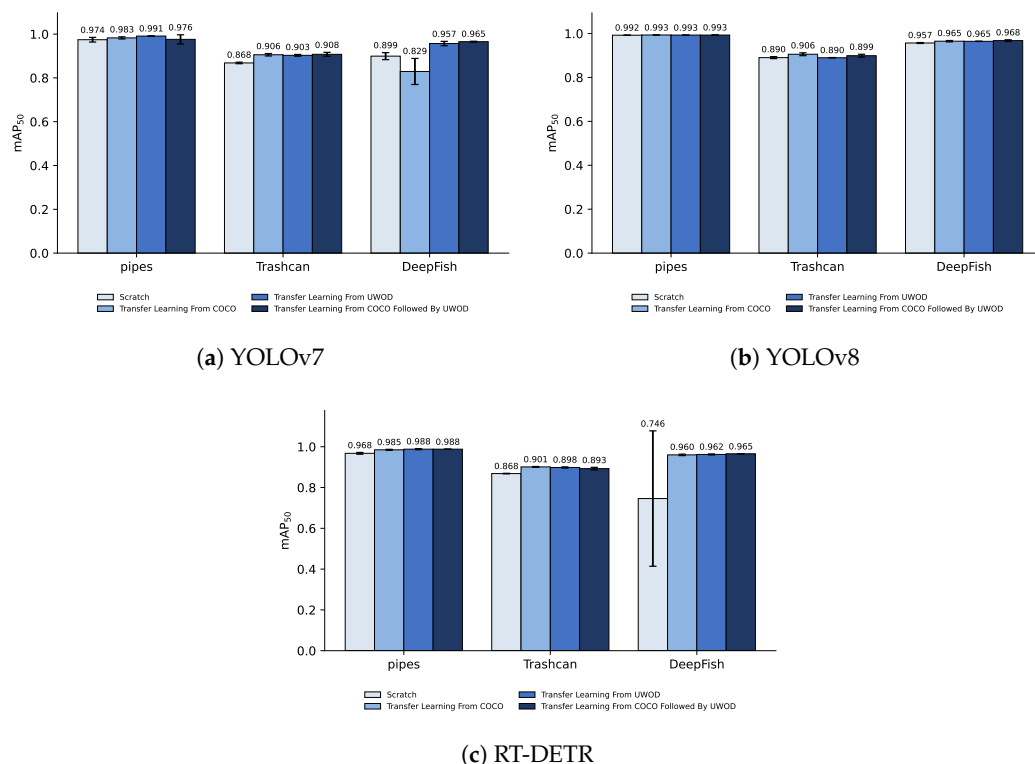
#### 5.4. Transfer Learning Experiments

##### 5.4.1. Models Performance on Small Scale Underwater Datasets

The performance of YOLOv7 [43], YOLOv8 [42], and RT-DETR [48] was evaluated on the Pipes, TrashCan, and DeepFish datasets using four training strategies—training from scratch, transfer learning from COCO, transfer learning from UWOD, and sequential transfer learning (COCO → UWOD)—each averaged over three random seeds (Section 4.4). As shown in Figure 14, sequential transfer learning achieved the best mean mAP<sub>50</sub> on the more complex DeepFish dataset across all three models, with improvements over training from scratch reaching 29.4% for RT-DETR; on TrashCan, COCO pre-training alone was strongest for YOLOv8 and RT-DETR, while sequential transfer was marginally ahead for YOLOv7. On simpler datasets such as Pipes, all pre-training strategies yielded similarly high mAP<sub>50</sub> scores (0.97–0.99 across all models), indicating limited benefit from transfer learning once sufficient in-domain data is available. RT-DETR [48] demonstrated its best performance with sequential or UWOD-only pre-training on Pipes and DeepFish, with COCO pre-training alone competitive only on TrashCan. Training from scratch consistently underperformed across all models, particularly on DeepFish: RT-DETR’s scratch-trained mAP<sub>50</sub> reached 0.936 and 0.939 on two seeds but collapsed to 0.363 on the third (mean  $0.746 \pm 0.332$ ), underscoring the initialization sensitivity this multi-seed evaluation was designed to detect. These findings, visualized in Figure 14, highlight the value of sequential and COCO-based pre-training for complex underwater datasets, while showing diminishing returns for simpler datasets like Pipes. This advantage is most plausibly explained by the complementary nature of the two pretraining stages: COCO provides broad, general-purpose visual features learned from a large and diverse image corpus, while subsequent UWOD pretraining adapts these features toward underwater-specific visual characteristics—color attenuation, scattering-induced haze, and the appearance of marine



organisms and debris—that are largely absent from COCO but shared across our target datasets. This two-stage adaptation narrows the domain gap the final fine-tuning stage must bridge, which particularly benefits DeepFish, the most challenging target dataset (Section 4.3). On simpler datasets such as Pipes, where even single-source pretraining already yields near-ceiling performance, the additional benefit of sequential pretraining is correspondingly smaller.



**Figure 14.** mAP<sub>50</sub> performance comparison on full datasets under different transfer learning strategies. Bars show mean  $\pm$  SD across three random seeds (0, 42, 70).

#### 5.4.2. Impact of Dataset Size on Models Performance

We evaluated the impact of our UWOD dataset on model performance when training data is highly limited. Using 10%, 20%, 30%, 40%, and 50% of the training data from the Pipes, TrashCan, and DeepFish datasets, we assessed how transfer learning from UWOD influences object detection models under scarce data conditions. We tested three models—YOLOv8 [42], YOLOv7 [43], and RT-DETR [48]—using four training strategies: training from scratch, transfer learning from COCO, transfer learning from UWOD, and sequential transfer learning (COCO followed by UWOD). The results (single-run mAP<sub>50</sub>) are presented in Figures 15–17.

Our experiments demonstrate substantial performance gains with transfer learning from UWOD, especially in low-data conditions across all models and datasets. The improvements are particularly pronounced when comparing UWOD-based transfer learning to training from scratch, with performance gains reaching up to more than 50% in extreme low-data scenarios.

**Performance on Pipes Dataset:** In the Pipes dataset with only 10% of the data, YOLOv8 [42] achieves an mAP<sub>50</sub> of 0.544 when trained from scratch, improving to 0.942 with transfer learning from UWOD and 0.555 with transfer learning from COCO. Sequential transfer learning (COCO followed by UWOD) reaches 0.944. YOLOv7 [43] shows similarly large improvements, from 0.016 when training from scratch to 0.952 with UWOD transfer learning and 0.819 with COCO transfer learning. RT-DETR [48] demonstrates the strongest

gains with COCO pre-training, increasing from 0.133 when training from scratch to 0.953 with COCO transfer learning, narrowly surpassing UWOD transfer learning at 0.927.

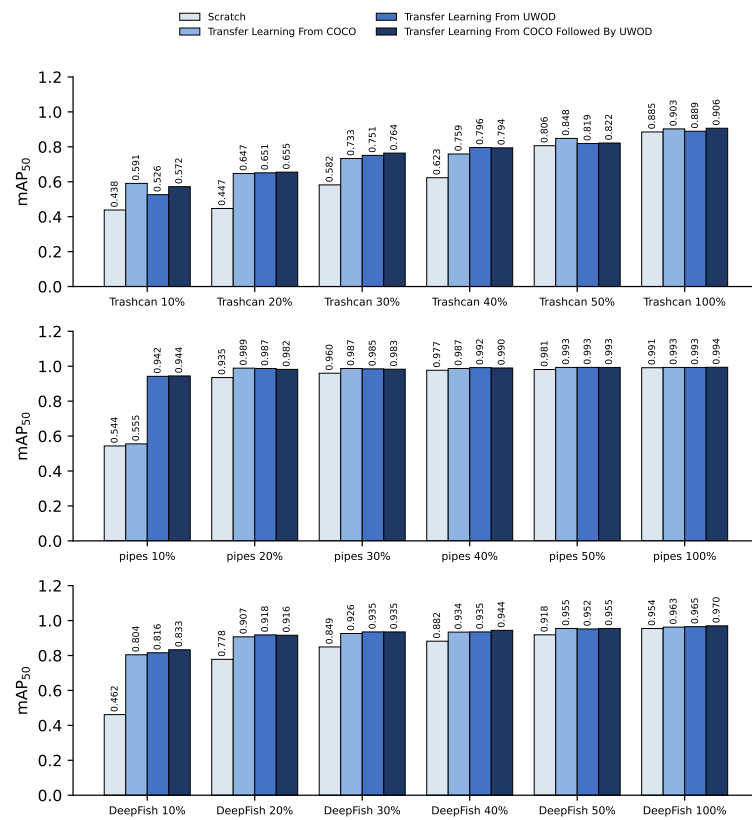


Figure 15. YOLOv8 [42] performance on small-scale underwater datasets with varying training data sizes.

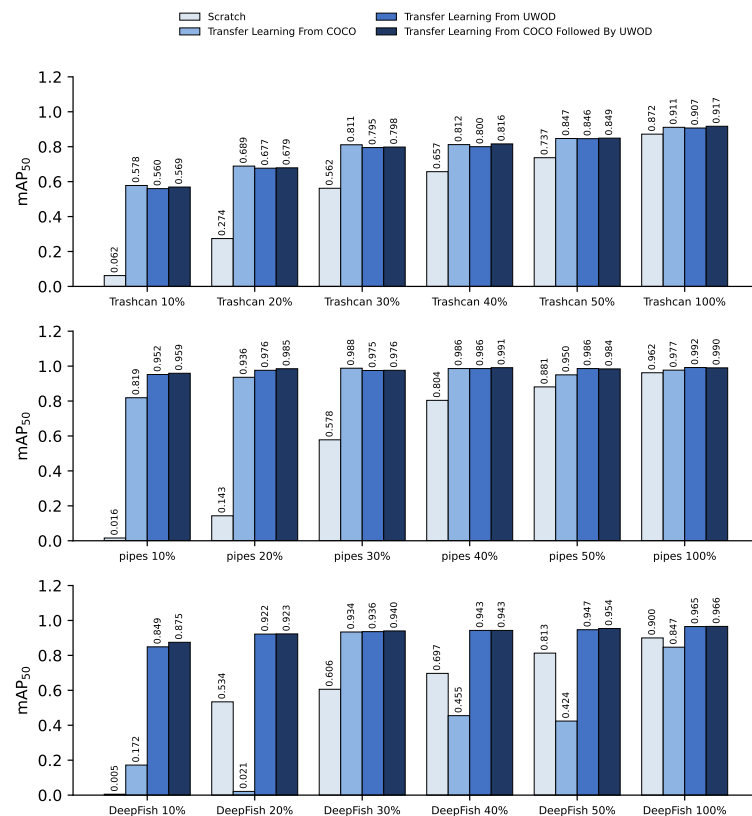


Figure 16. YOLOv7 [43] performance on small-scale underwater datasets with varying training data sizes.

**Performance on TrashCan Dataset:** The TrashCan dataset, representing a more complex multi-class scenario with 16 object categories, shows consistent benefits from domain-specific pre-training. With 10% of training data, YOLOv8 [42] improves from 0.438 when training from scratch to 0.526 with UWOD transfer learning, trailing COCO transfer learning at 0.591. Sequential transfer learning achieves 0.572, falling between the two. YOLOv7 [43] exhibits substantial improvements from 0.062 when training from scratch to 0.560 with UWOD transfer learning, slightly below COCO transfer learning at 0.578.

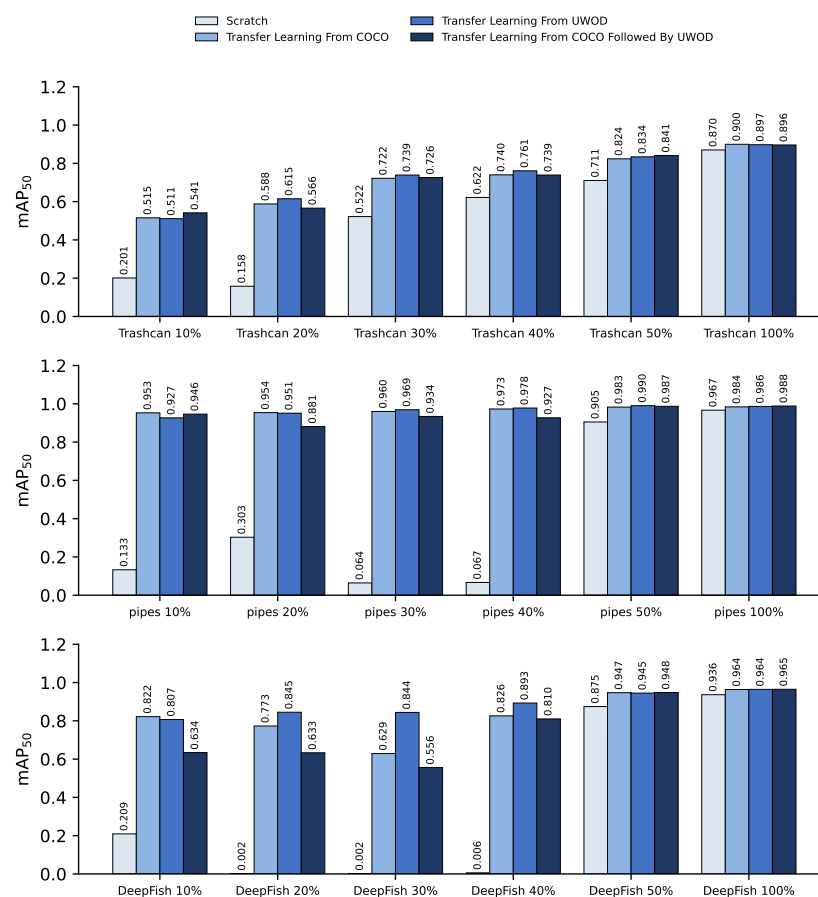
**Performance on DeepFish Dataset:** The DeepFish dataset results highlight the particular effectiveness of domain-specific transfer learning for underwater applications. With 10% of training data, YOLOv8 [42] enhances performance from 0.462 when training from scratch to 0.816 with UWOD transfer learning, trailing COCO transfer learning performance at 0.804 only slightly. Notably, on the 10% DeepFish slice, YOLOv7 [43] trained from scratch collapses to 0.005 mAP<sub>50</sub>, while COCO pre-training reaches only 0.172—both far below UWOD pre-training 0.849 under the identical training configuration (same schedule, learning rate, and class set). With so little in-domain data available, both random initialization and COCO's general-purpose features struggle to adapt; the same architecture recovers strongly once pretrained on UWOD (0.849) or sequentially via COCO → UWOD (0.875). This contrast still directly motivates domain-specific pretraining for low-data underwater regimes.

**Trends Across Data Sizes:** As training data increases, the benefits of transfer learning from UWOD remain consistent across all datasets. Even with 50% of training data, UWOD-based transfer learning continues to outperform training from scratch and often matches or exceeds COCO-based transfer learning. Sequential transfer learning frequently achieves the best overall performance, particularly for YOLOv7 [43] and YOLOv8 [42] on complex datasets like TrashCan and DeepFish.

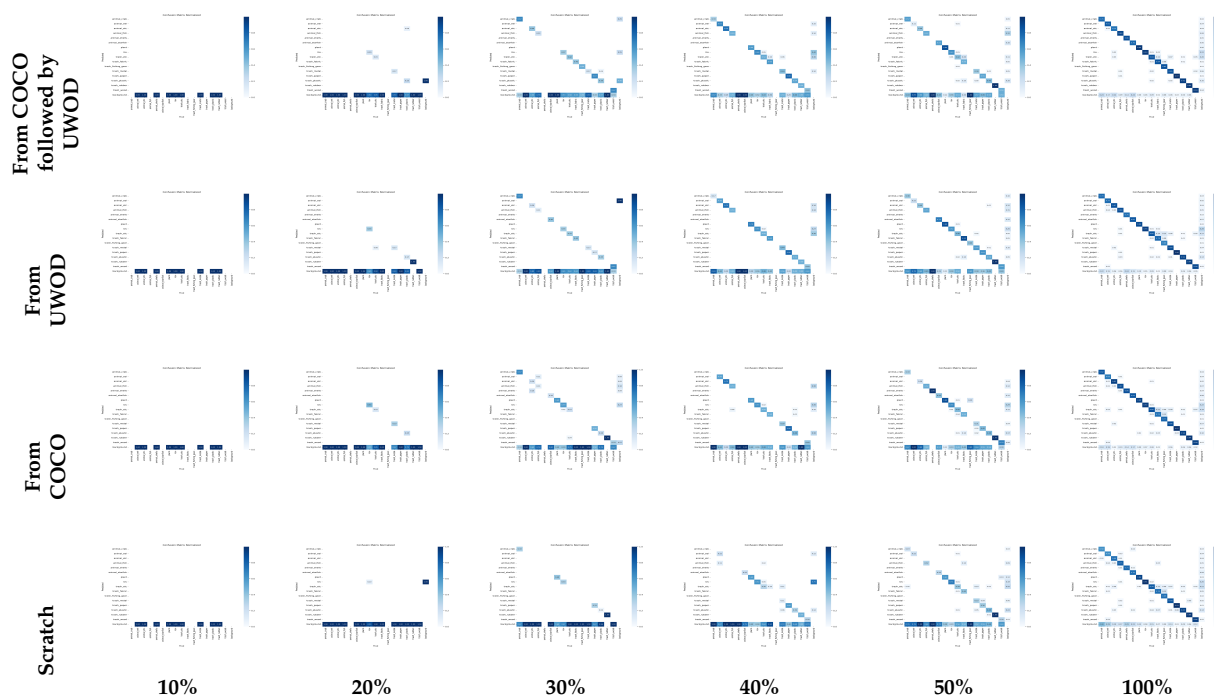
On the other hand, we evaluated the detection performance for the classification/detection models using confusion matrix including training from Scratch and Transfer Learning from COCO, COCO followed by UWOD, and UWOD on small-scale underwater across varying training data sizes 10%, 20%, 30%, 40% and 50%. Figure 18 shows a sample of detection performance for YOLOv8 [42] on Trashcan dataset.

The confusion matrices presented in Figure 18 provide additional insights into model performance across different transfer learning strategies. The visualizations demonstrate that models trained with UWOD-based transfer learning exhibit superior classification accuracy and reduced confusion between similar object classes, particularly evident in the 30% and 40% data scenarios where domain-specific features provide critical advantages over general pre-training approaches.

These findings emphasize the substantial value of domain-specific datasets like UWOD for underwater computer vision applications, particularly when data collection and annotation resources are constrained. The consistent performance improvements across different models, datasets, and data sizes validate the effectiveness of our dataset compilation and transfer learning approach.



**Figure 17.** RT-DETR [48] performance on small-scale underwater datasets with varying training data sizes.



**Figure 18.** Evaluation of YOLOv8 [42] training from scratch and various transfer learning strategies on Trashcan underwater dataset across varying training data sizes: 10%, 20%, 30%, 40%, 50%, and 100%.

### 5.4.3. Multi-Seed Robustness Validation

To directly address concerns regarding the robustness of single-run results to random initialization, Table 9 reports mean  $\pm$  standard deviation across three random seeds (0, 42, 70) for all three models, three small-scale underwater datasets, and four transfer learning strategies. We restrict this seed-level analysis to 10%, 50%, and 100% of the training data—spanning the low-data, moderate-data, and full-data regimes—since repeating all six training data sizes from Section 5.4.2 across three seeds each was computationally prohibitive; the intermediate 20%, 30%, and 40% training data sizes therefore report the single-run values shown in Figures 15–17. The results confirm that our transfer learning conclusions are robust to random seed: UWOD and sequential COCO  $\rightarrow$  UWOD pre-training consistently outperform scratch training across all three multi-seed training data sizes, with one notable exception: RT-DETR trained from scratch on full DeepFish data reached mAP<sub>50</sub> of 0.936 and 0.939 on two seeds but collapsed to 0.363 on the third (mean  $0.746 \pm 0.332$ ), illustrating the initialization sensitivity this analysis was designed to detect.

**Table 9.** Multi-seed robustness validation: mAP<sub>50</sub> (mean  $\pm$  SD,  $n = 3$  seeds) across three models, three datasets, four transfer-learning strategies, and three training data sizes (10%, 50%, 100%). For each model, dataset and training-data size combination, underlined value(s) denote the transfer-learning strategy achieving the highest mean mAP<sub>50</sub>.

| Model   | Training Data Size | Dataset  | Scratch           | COCO                                | UWOD                                | COCO $\rightarrow$ UWOD             |
|---------|--------------------|----------|-------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| YOLOv7  | 10%                | Pipes    | 0.018 $\pm$ 0.003 | 0.882 $\pm$ 0.058                   | 0.949 $\pm$ 0.008                   | <u>0.955 <math>\pm</math> 0.011</u> |
|         |                    | TrashCan | 0.087 $\pm$ 0.023 | <u>0.575 <math>\pm</math> 0.005</u> | 0.549 $\pm$ 0.013                   | <u>0.558 <math>\pm</math> 0.009</u> |
|         |                    | DeepFish | 0.135 $\pm$ 0.113 | <u>0.165 <math>\pm</math> 0.157</u> | 0.847 $\pm$ 0.016                   | <u>0.870 <math>\pm</math> 0.005</u> |
|         | 50%                | Pipes    | 0.854 $\pm$ 0.037 | 0.962 $\pm$ 0.017                   | <u>0.989 <math>\pm</math> 0.003</u> | 0.963 $\pm$ 0.028                   |
|         |                    | TrashCan | 0.736 $\pm$ 0.014 | 0.845 $\pm$ 0.002                   | 0.847 $\pm$ 0.004                   | <u>0.849 <math>\pm</math> 0.003</u> |
|         |                    | DeepFish | 0.797 $\pm$ 0.030 | 0.581 $\pm$ 0.140                   | 0.947 $\pm$ 0.004                   | <u>0.950 <math>\pm</math> 0.006</u> |
|         | 100%               | Pipes    | 0.974 $\pm$ 0.011 | 0.983 $\pm$ 0.005                   | <u>0.991 <math>\pm</math> 0.001</u> | 0.976 $\pm$ 0.021                   |
|         |                    | TrashCan | 0.868 $\pm$ 0.004 | 0.906 $\pm$ 0.006                   | 0.903 $\pm$ 0.004                   | <u>0.908 <math>\pm</math> 0.009</u> |
|         |                    | DeepFish | 0.899 $\pm$ 0.016 | 0.829 $\pm$ 0.060                   | 0.957 $\pm$ 0.010                   | <u>0.965 <math>\pm</math> 0.002</u> |
| YOLOv8  | 10%                | Pipes    | 0.467 $\pm$ 0.117 | 0.576 $\pm$ 0.086                   | 0.955 $\pm$ 0.012                   | 0.953 $\pm$ 0.011                   |
|         |                    | TrashCan | 0.416 $\pm$ 0.019 | <u>0.592 <math>\pm</math> 0.001</u> | <u>0.522 <math>\pm</math> 0.004</u> | 0.560 $\pm$ 0.010                   |
|         |                    | DeepFish | 0.487 $\pm$ 0.025 | <u>0.799 <math>\pm</math> 0.005</u> | 0.821 $\pm$ 0.004                   | <u>0.830 <math>\pm</math> 0.005</u> |
|         | 50%                | Pipes    | 0.982 $\pm$ 0.001 | 0.991 $\pm$ 0.002                   | <u>0.993 <math>\pm</math> 0.001</u> | <u>0.993 <math>\pm</math> 0.001</u> |
|         |                    | TrashCan | 0.807 $\pm$ 0.005 | <u>0.847 <math>\pm</math> 0.006</u> | 0.810 $\pm$ 0.007                   | 0.825 $\pm$ 0.005                   |
|         |                    | DeepFish | 0.920 $\pm$ 0.004 | <u>0.952 <math>\pm</math> 0.004</u> | <u>0.952 <math>\pm</math> 0.001</u> | <u>0.952 <math>\pm</math> 0.004</u> |
|         | 100%               | Pipes    | 0.992 $\pm$ 0.001 | <u>0.993 <math>\pm</math> 0.000</u> | <u>0.993 <math>\pm</math> 0.000</u> | <u>0.993 <math>\pm</math> 0.001</u> |
|         |                    | TrashCan | 0.890 $\pm$ 0.005 | <u>0.906 <math>\pm</math> 0.007</u> | 0.890 $\pm$ 0.001                   | 0.899 $\pm$ 0.007                   |
|         |                    | DeepFish | 0.957 $\pm$ 0.002 | 0.965 $\pm$ 0.003                   | 0.965 $\pm$ 0.001                   | <u>0.968 <math>\pm</math> 0.004</u> |
| RT-DETR | 10%                | Pipes    | 0.304 $\pm$ 0.151 | <u>0.957 <math>\pm</math> 0.006</u> | 0.935 $\pm$ 0.009                   | 0.950 $\pm$ 0.008                   |
|         |                    | TrashCan | 0.189 $\pm$ 0.024 | <u>0.506 <math>\pm</math> 0.009</u> | 0.511 $\pm$ 0.008                   | <u>0.528 <math>\pm</math> 0.012</u> |
|         |                    | DeepFish | 0.225 $\pm$ 0.022 | <u>0.821 <math>\pm</math> 0.001</u> | 0.794 $\pm$ 0.020                   | 0.759 $\pm$ 0.109                   |
|         | 50%                | Pipes    | 0.914 $\pm$ 0.017 | <u>0.982 <math>\pm</math> 0.001</u> | 0.980 $\pm$ 0.009                   | 0.977 $\pm$ 0.009                   |
|         |                    | TrashCan | 0.731 $\pm$ 0.019 | <u>0.830 <math>\pm</math> 0.005</u> | <u>0.834 <math>\pm</math> 0.002</u> | 0.833 $\pm$ 0.007                   |
|         |                    | DeepFish | 0.880 $\pm$ 0.005 | 0.947 $\pm$ 0.001                   | <u>0.944 <math>\pm</math> 0.001</u> | <u>0.948 <math>\pm</math> 0.001</u> |
|         | 100%               | Pipes    | 0.968 $\pm$ 0.004 | 0.985 $\pm$ 0.003                   | <u>0.988 <math>\pm</math> 0.003</u> | <u>0.988 <math>\pm</math> 0.001</u> |
|         |                    | TrashCan | 0.868 $\pm$ 0.001 | <u>0.901 <math>\pm</math> 0.002</u> | 0.898 $\pm$ 0.003                   | 0.893 $\pm$ 0.006                   |
|         |                    | DeepFish | 0.746 $\pm$ 0.332 | 0.960 $\pm$ 0.004                   | 0.962 $\pm$ 0.002                   | <u>0.965 <math>\pm</math> 0.001</u> |

## 6. Conclusions

This work addressed the critical challenges in underwater object detection by developing a comprehensive dataset and evaluating transfer learning strategies for domain-specific applications. The limitations of existing underwater datasets, including restricted size,

class diversity, and annotation quality, have significantly hindered the development of robust underwater object detection models [93,94]. Our research contributes to addressing these gaps through the creation of the UWOD dataset and systematic evaluation of transfer learning approaches.

The UWOD dataset represents a substantial advancement in underwater computer vision resources, comprising over 107,000 images with approximately 374,000 annotations across 39 distinct classes. Through the integration of seven publicly available underwater datasets and implementation of a semi-automatic annotation pipeline, we created a comprehensive benchmark that captures diverse underwater environments, species, and imaging conditions. The rigorous annotation process began with manual labeling of an initial subset of images, which was used to train a model that generated labels for the remaining data; all auto-generated labels were then manually reviewed and corrected where needed. This process ensures high-quality ground truth data essential for training robust detection models.

Our comprehensive evaluation of state-of-the-art object detection models on the UWOD dataset established important performance benchmarks for the underwater domain. YOLOv8 [42] and YOLOv7 [43] demonstrated superior performance with strong accuracy and efficient processing capabilities, while RT-DETR [48] and YOLOX [47] provided balanced approaches suitable for various underwater scenarios. These findings provide valuable guidance for researchers selecting appropriate models for underwater object detection tasks.

The transfer learning experiments revealed significant insights into the effectiveness of domain-specific pre-training for underwater applications. Our results demonstrate that transfer learning from the UWOD dataset substantially often outperforms transfer learning from general datasets like COCO, particularly in low-data scenarios. In extreme cases with only 10% of training data, UWOD-based transfer learning improved performance by up to more than 50% compared to training from scratch. Sequential transfer learning (COCO followed by UWOD) achieved the strongest performance on our most challenging target dataset, DeepFish, though its advantage over single-source pretraining was smaller on simpler datasets such as Pipes and TrashCan, indicating the synergistic benefits of combining general object detection knowledge with domain-specific features under substantial domain shift. We further validated these findings across three random seeds, confirming that the reported gains are not artifacts of a single training run, though training from scratch showed markedly higher seed-to-seed variability than pretrained strategies in low-data conditions. These findings are particularly valuable for practical applications where data collection and annotation are challenging and expensive.

The study's impact extends beyond performance metrics to practical implications for underwater computer vision applications. By demonstrating that effective models can be trained with limited domain-specific data through appropriate transfer learning strategies, our work facilitates the development of automated annotation systems and reduces the barrier to entry for underwater object detection research. To maximize research impact while respecting licensing constraints, we provide trained model weights and automated annotation pipelines that capture the domain-specific knowledge gained from our comprehensive dataset compilation. This approach enables researchers to apply our model directly to new underwater imagery and benefit from our transfer learning insights without requiring access to the complete compiled dataset.

However, several limitations must be acknowledged. Due to licensing restrictions of some integrated datasets, particularly UOT100, the complete UWOD dataset cannot be directly shared, which limits exact reproducibility of our specific dataset splits. For video-derived source UOT100, the partition is assigned at the video level, which facilitates faithful



reconstruction of our splits once the original sources are obtained. Additionally, our evaluation focused on specific underwater environments and object types, and generalization to other marine conditions requires further investigation. The computational requirements for training on large-scale datasets may also limit accessibility for some researchers.

Future research directions should focus on expanding domain-specific datasets to encompass more diverse underwater conditions, including different water bodies, depths, and lighting scenarios. Investigation of additional domain adaptation techniques performance beyond transfer learning, such as adversarial training and meta-learning approaches, could further improve model robustness. The development of specialized architectures optimized for underwater imaging conditions represents another promising direction. Furthermore, exploring automated methods for creating large-scale underwater datasets while addressing licensing and ethical considerations could significantly advance the field.

In conclusion, this work makes significant contributions to underwater object detection through the development of a comprehensive dataset compilation methodology and systematic evaluation of transfer learning strategies. While complete dataset sharing is constrained by licensing restrictions, our approach of providing trained models and automated annotation tools offers practical value to the research community. The demonstrated effectiveness of domain-specific transfer learning and our insights into underwater object detection performance provide a strong foundation for future research and practical applications in underwater computer vision.

**Author Contributions:** Conceptualization, M.M.A., N.A.A. (Nouf A. Alrowais) and A.M.A.; methodology, N.A.A. (Nouf A. Alrowais), A.M.A. and N.A.A. (Nada A. Almugrem); software, A.A.J.; validation, N.A.A. (Nouf A. Alrowais), A.M.A. and N.A.A. (Nada A. Almugrem); formal analysis, N.A.A. (Nouf A. Alrowais), A.M.A., N.A.A. (Nada A. Almugrem) and W.A.A.; investigation, N.A.A. (Nouf A. Alrowais), A.M.A. and N.A.A. (Nada A. Almugrem); resources, M.M.A.; data curation, A.M.A., N.A.A. (Nada A. Almugrem), H.R.A., R.A. (Remass AlSaeed), A.A., R.A. (Royouf Alotaibi), A.O.A. and E.B.K.; writing—original draft preparation, N.A.A. (Nouf A. Alrowais), A.M.A., N.A.A. (Nada A. Almugrem), A.O.A. and H.M.A.; writing—review and editing, M.M.A.; visualization, N.A.A. (Nouf A. Alrowais), A.M.A. and N.A.A. (Nada A. Almugrem); supervision, M.M.A.; project administration, M.M.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable; all person/diver imagery originates from a publicly released, consented dataset (VDD-C); no new human-subject data were collected.

**Informed Consent Statement:** Not applicable; all person/diver imagery originates from a publicly released, consented dataset (VDD-C); no new human-subject data were collected.

**Data Availability Statement:** Data sources are contained within the article.

**Acknowledgments:** During the preparation of this manuscript, the authors used ChatGPT (GPT-5.5, OpenAI) and Claude (Claude Sonnet 5, Anthropic) to improve the clarity and grammar of the writing. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Zhao, X.; Wang, X.; Du, Z. Research on detection method for the leakage of underwater pipeline by YOLOv3. In *Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA)*; IEEE: New York, NY, USA, 2020; pp. 637–642.
2. Lee, D.; Kim, G.; Kim, D.; Myung, H.; Choi, H.T. Vision-based object detection and tracking for autonomous navigation of underwater robots. *Ocean Eng.* **2012**, *48*, 59–68. [[CrossRef](#)]
3. Watanabe, J.I.; Shao, Y.; Miura, N. Underwater and airborne monitoring of marine ecosystems and debris. *J. Appl. Remote Sens.* **2019**, *13*, 044509. [[CrossRef](#)]

4. Salhaoui, M.; Molina-Molina, J.C.; Guerrero-González, A.; Arioua, M.; Ortiz, F.J. Autonomous underwater monitoring system for detecting life on the seabed by means of computer vision cloud services. *Remote Sens.* **2020**, *12*, 1981. [\[CrossRef\]](#)
5. Ahmed, S.; Khan, M.F.R.; Labib, M.F.A.; Chowdhury, A.E. An observation of vision based underwater object detection and tracking. In *Proceedings of the 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*; IEEE: New York, NY, USA, 2020; pp. 117–122.
6. Liu, R.W.; Yuan, W.; Chen, X.; Lu, Y. An enhanced CNN-enabled learning method for promoting ship detection in maritime surveillance system. *Ocean Eng.* **2021**, *235*, 109435. [\[CrossRef\]](#)
7. Moghimi, M.K.; Mohanna, F. Reliable object recognition using deep transfer learning for marine transportation systems with underwater surveillance. *IEEE Trans. Intell. Transp. Syst.* **2022**, *24*, 2515–2524. [\[CrossRef\]](#)
8. Li, Y.; Lu, H.; Li, J.; Li, X.; Li, Y.; Serikawa, S. Underwater image de-scattering and classification by deep neural network. *Comput. Electr. Eng.* **2016**, *54*, 68–77. [\[CrossRef\]](#)
9. Yuan, X.; Guo, L.; Luo, C.; Zhou, X.; Yu, C. A survey of target detection and recognition methods in underwater turbid areas. *Appl. Sci.* **2022**, *12*, 4898. [\[CrossRef\]](#)
10. Fayaz, S.; Parah, S.; Qureshi, G. Underwater object detection: Architectures and algorithms—A comprehensive review. *Multimed. Tools Appl.* **2022**, *81*, 20871–20916. [\[CrossRef\]](#)
11. Fu, C.; Liu, R.; Fan, X.; Chen, P.; Fu, H.; Yuan, W.; Zhu, M.; Luo, Z. Rethinking General Underwater Object Detection: Datasets, Challenges, and Solutions. *Neurocomputing* **2023**, *517*, 243–256. [\[CrossRef\]](#)
12. Li, C.; Anwar, S.; Porikli, F. Underwater scene prior inspired deep underwater image and video enhancement. *Pattern Recognit.* **2020**, *98*, 107038. [\[CrossRef\]](#)
13. Li, C.; Guo, C.; Ren, W.; Cong, R.; Hou, J.; Kwong, S.; Tao, D. An underwater image enhancement benchmark dataset and beyond. *IEEE Trans. Image Process.* **2019**, *29*, 4376–4389. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Islam, M.I.; Xia, Y.; Sattar, J. Fast underwater image enhancement for improved visual perception. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3227–3234. [\[CrossRef\]](#)
15. Li, X.; Shang, M.; Qin, H.; Chen, L. Fast Accurate Fish Detection and Recognition of Underwater Images with Fast R-CNN. In *Proceedings of the OCEANS 2015-MTS/IEEE Washington*; IEEE: New York, NY, USA, 2015; pp. 1–5.
16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99.
17. Zhao, Z.Q.; Zheng, P.; Xu, S.t.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [\[CrossRef\]](#) [\[PubMed\]](#)
18. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*; IEEE: New York, NY, USA, 2005; Volume 1, pp. 886–893.
19. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR 2001; IEEE: New York, NY, USA, 2001; Volume 1, pp. I-511–I-518.
20. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [\[CrossRef\]](#)
21. Solunke, B.R.; Gengaje, S.R. A Review on Traditional and Deep Learning based Object Detection Methods. In *Proceedings of the 2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*; IEEE: New York, NY, USA, 2023; pp. 1–7.
22. Khairandish, M.O.; Gurta, R.; Sharma, M. A hybrid model of faster R-CNN and SVM for tumor detection and classification of MRI brain images. *Int. J. Mech. Prod. Eng. Res. Dev.* **2020**, *10*, 6863–6876. [\[CrossRef\]](#)
23. Tang, C.; Feng, Y.; Yang, X.; Zheng, C.; Zhou, Y. The Object Detection Based on Deep Learning. In *Proceedings of the 2017 4th International Conference on Information Science and Control Engineering (ICISCE)*; IEEE: New York, NY, USA, 2017; pp. 723–728. [\[CrossRef\]](#)
24. O'Mahony, N.; Campbell, S.; Carvalho, A.; Harapanahalli, S.; Hernandez, G.V.; Krpalkova, L.; Riordan, D.; Walsh, J. Deep learning vs. traditional computer vision. In *Proceedings of the Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC)*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 11, pp. 128–144.
25. Ward, I.R.; Laga, H.; Bennamoun, M. RGB-D image-based object detection: From traditional methods to deep learning techniques. *RGB-D Image Analysis and Processing*; Springer: Cham, Switzerland, 2019; pp. 169–201.
26. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [\[CrossRef\]](#)
27. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [\[CrossRef\]](#)
28. Panigrahi, S.; Nanda, A.; Swarnkar, T. A survey on transfer learning. In *Intelligent and Cloud Computing: Proceedings of ICICC 2019*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 1, pp. 781–789.
29. Torralba, A.; Efros, A.A. Unbiased look at dataset bias. In *Proceedings of the CVPR 2011*; IEEE: New York, NY, USA, 2011; pp. 1521–1528.

30. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2017; pp. 7167–7176.
31. Li, W.; Li, F.; Luo, Y.; Wang, P. Deep domain adaptive object detection: A survey. In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI)*; IEEE: New York, NY, USA, 2020; pp. 1808–1813.
32. Wang, D.; Shelhamer, E.; Liu, S.; Olshausen, B.; Darrell, T. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual, 3–7 May 2021.
33. Singh, B.; Davis, L.S. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2018; pp. 3578–3587.
34. Fabbri, C.; Islam, M.J.; Sattar, J. Enhancing underwater imagery using generative adversarial networks. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2018; pp. 7159–7165.
35. Ravanbakhsh, M.; Shortis, M.R.; Shafait, F.; Mian, A.; Harvey, E.S.; Seager, J.W. Automated Fish Detection in Underwater Images Using Shape-Based Level Sets. *Photogramm. Rec.* **2015**, *30*, 46–62. [[CrossRef](#)]
36. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; *Proceedings, Part V 13*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
37. Xu, S.; Zhang, M.; Song, W.; Mei, H.; He, Q.; Liotta, A. A systematic review and analysis of deep learning-based underwater object detection. *Neurocomputing* **2023**, *527*, 204–232. [[CrossRef](#)]
38. Anwar, S.M.; Majid, M.; Qayyum, A.; Awais, M.; Alnowami, M.; Khan, M.K. Medical image analysis using convolutional neural networks: A review. *J. Med Syst.* **2018**, *42*, 226. [[CrossRef](#)] [[PubMed](#)]
39. Yang, X.; Sun, H.; Fu, K.; Yang, J.; Sun, X.; Yan, M.; Guo, Z. Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks. *Remote Sens.* **2018**, *10*, 132. [[CrossRef](#)]
40. Moniruzzaman, M.; Islam, S.M.S.; Bennamoun, M.; Lavery, P. Deep learning on underwater marine object detection: A survey. In *Proceedings of the Advanced Concepts for Intelligent Vision Systems: 18th International Conference, ACIVS 2017, Antwerp, Belgium, 18–21 September 2017*; *Proceedings 18*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 150–160.
41. Jian, M.; Qi, Q.; Dong, J.; Yin, Y.; Lam, K.M. Integrating QDWD with pattern distinctness and local contrast for underwater saliency detection. *J. Vis. Commun. Image Represent.* **2018**, *53*, 31–41. [[CrossRef](#)]
42. Ultralytics. Ultralytics YOLOv8: Real-Time Object Detector. 2023. Available online: <https://github.com/ultralytics/ultralytics> (accessed on 26 June 2025).
43. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2023; pp. 7464–7475.
44. Ultralytics. YOLOv5. 2023. Available online: <https://github.com/ultralytics/yolov5> (accessed on 4 September 2026).
45. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*; IEEE: New York, NY, USA, 2019; pp. 9627–9636.
46. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2020; pp. 10781–10790.
47. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv* **2021**, arXiv:2107.08430.
48. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-time Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2024; pp. 16965–16974.
49. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In *Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; *Proceedings, Part I 14*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
50. Girshick, R. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*; IEEE: New York, NY, USA, 2015; pp. 1440–1448.
51. Huang, H.; Zhou, H.; Yang, X.; Zhang, L.; Qi, L.; Zang, A.Y. Faster R-CNN for marine organisms detection and recognition using data augmentation. *Neurocomputing* **2019**, *337*, 372–384. [[CrossRef](#)]
52. Mandal, R.; Connolly, R.M.; Schlacher, T.A.; Stantic, B. Assessing fish abundance from underwater video using deep neural networks. In *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*; IEEE: New York, NY, USA, 2018; pp. 1–6.
53. Liu, S.; Li, X.; Gao, M.; Cai, Y.; Nian, R.; Li, P.; Yan, T.; Lendasse, A. Embedded Online Fish Detection and Tracking System via YOLOv3 and Parallel Correlation Filter. In *Proceedings of the OCEANS 2018 MTS/IEEE Charleston*; IEEE: New York, NY, USA, 2018; pp. 1–6. [[CrossRef](#)]
54. Jain, S. DeepSeaNet: Improving Underwater Object Detection using EfficientDet. In *Proceedings of the 2024 4th International Conference on Applied Artificial Intelligence (ICAPAI)*; IEEE: New York, NY, USA, 2024; pp. 1–11. [[CrossRef](#)]

55. Liu, H.; Song, P.; Ding, R. Towards Domain Generalization in Underwater Object Detection. In *Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP)*; IEEE: New York, NY, USA, 2020; pp. 1971–1975.
56. Song, P.; Li, P.; Dai, L.; Wang, T.; Chen, Z. Boosting R-CNN: Reweighting R-CNN samples by RPN's error for underwater object detection. *Neurocomputing* **2023**, *530*, 150–164. [\[CrossRef\]](#)
57. Reddy Nandyala, N.; Kumar Sanodiya, R. Underwater Object Detection Using Synthetic Data. In *Proceedings of the 2023 11th International Symposium on Electronic Systems Devices and Computing (ESDC)*; IEEE: New York, NY, USA, 2023; Volume 1, pp. 1–6. [\[CrossRef\]](#)
58. Zhou, H.; Kong, M.; Yuan, H.; Pan, Y.; Wang, X.; Chen, R.; Lu, W.; Wang, R.; Yang, Q. Real-time underwater object detection technology for complex underwater environments based on deep learning. *Ecol. Inform.* **2024**, *82*, 102680. [\[CrossRef\]](#)
59. Zheng, S.; Wang, R.; Wang, L. Underwater Fish Object Detection with Degraded Prior Knowledge. *Electronics* **2024**, *13*, 2346. [\[CrossRef\]](#)
60. Guntha, P.; Beaulah, P.M.R. A Comprehensive Review on Underwater Object Detection Techniques. In *Proceedings of the 2024 International Conference on Computing and Data Science (ICCDs)*; IEEE: New York, NY, USA, 2024; pp. 1–6.
61. Liu, R.; Fan, X.; Zhu, M.; Hou, M.; Luo, Z. Real-world underwater enhancement: Challenges, benchmarks, and solutions under natural light. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 4861–4875. [\[CrossRef\]](#)
62. Zu, Y.; Zhang, L.; Li, S.; Fan, Y.; Liu, Q. EF-UODA: Underwater Object Detection Based on Enhanced Feature. *J. Mar. Sci. Eng.* **2024**, *12*, 729. [\[CrossRef\]](#)
63. Xie, S.; Li, P.; Gao, J.; Yue, Z.; Fan, X.; Liu, R. Breaking the water dilemma: Transmission-guided bilevel adaptive learning for underwater imagery. *Neurocomputing* **2024**, *596*, 127909. [\[CrossRef\]](#)
64. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*; IEEE: New York, NY, USA, 2009; pp. 248–255.
65. Huang, H.; Jiang, T.; Zhang, Z.; Sun, Y.; Qin, H.; Li, X.; Yang, X. Learning strategies for underwater robot autonomous manipulation control. *J. Frankl. Inst.* **2024**, *361*, 106773. [\[CrossRef\]](#)
66. Li, Z.; Li, J.; Zhang, P.; Zheng, L.; Shen, Y.; Li, Q.; Li, X.; Li, T. A transfer-based framework for underwater target detection from hyperspectral imagery. *Remote Sens.* **2023**, *15*, 1023. [\[CrossRef\]](#)
67. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135–153. [\[CrossRef\]](#)
68. Zhou, K.; Liu, Z.; Qiao, Y.; Xiang, T.; Loy, C.C. Domain generalization: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 4396–4415. [\[CrossRef\]](#) [\[PubMed\]](#)
69. Wang, J.; Li, H.; Dong, C.; Wang, J.; Zheng, B.; Xing, T. An Underwater Side-Scan Sonar Transfer Recognition Method Based on Crossed Point-to-Point Second-Order Self-Attention Mechanism. *Remote Sens.* **2023**, *15*, 4517. [\[CrossRef\]](#)
70. Dobbs, C.E.; Neilsen, T.B. Testing transfer learning for source ranging in a laboratory tank. *J. Acoust. Soc. Am.* **2024**, *155*, A277–A278. [\[CrossRef\]](#)
71. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [\[CrossRef\]](#)
72. Shao, S.; Li, Z.; Zhang, T.; Peng, C.; Yu, G.; Zhang, X.; Li, J.; Sun, J. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*; IEEE: New York, NY, USA, 2019; pp. 8430–8439.
73. Wang, Z.; Liu, C.; Wang, S.; Tang, T.; Tao, Y.; Yang, C.; Li, H.; Liu, X.; Fan, X. UDD: An Underwater Open-sea Farm Object Detection Dataset for Underwater Robot Picking. *arXiv* **2020**, arXiv:2003.01446.
74. Liu, C.; Li, H.; Wang, S.; Zhu, M.; Wang, D.; Fan, X.; Wang, Z. A Dataset and Benchmark of Underwater Object Detection for Robot Picking. In *Proceedings of the 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*; IEEE: New York, NY, USA, 2021; pp. 1–6.
75. Jiang, L.; Wang, Y.; Jia, Q.; Xu, S.; Liu, Y.; Fan, X.; Li, H.; Liu, R.; Xue, X.; Wang, R. Underwater species detection using channel sharpening attention. In *Proceedings of the 29th ACM International Conference on Multimedia*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 4259–4267.
76. Pedersen, M.; Haurum, J.B.; Gade, R.; Moeslund, T.B.; Madsen, N. Detection of Marine Animals in a New Underwater Dataset with Varying Visibility. In *Proceedings of the the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*; IEEE: New York, NY, USA, 2019.
77. Haurum, J.B.; Gudiksen, M.; Hansen, L.N.; Aalborg University. Kaggle: The Brackish Dataset. 2020. Available online: <https://www.kaggle.com/datasets/aalborguniversity/brackish-dataset/data> (accessed on 28 October 2024).
78. Roboflow. Aquarium Dataset. 2020. Available online: <https://public.roboflow.com/object-detection/aquarium> (accessed on 4 September 2024).
79. de Langis, K.; Fulton, M.; Sattar, J. Towards Robust Visual Diver Detection Onboard Autonomous Underwater Robots: Assessing the Effects of Models and Data. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: New York, NY, USA, 2021; pp. 5372–5378. [\[CrossRef\]](#)



80. de Langis, K.; Fulton, M.; Sattar, J. *Video Diver Dataset (VDD-C): 100,000 Annotated Images of Divers Underwater*; Data Repository for the University of Minnesota (DRUM): Minneapolis, MN, USA, 2021. [[CrossRef](#)]
81. Kezebou, L.; Oludare, V.; Panetta, K.; Agaian, S.S. Underwater Object Tracking Benchmark and Dataset. In *Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST)*; IEEE: New York, NY, USA, 2019; pp. 1–6. [[CrossRef](#)]
82. Panetta, K.; Kezebou, L.; Oludare, V.; Agaian, S. Comprehensive Underwater Object Tracking Benchmark Dataset and Underwater Image Enhancement With GAN. *IEEE J. Ocean. Eng.* **2022**, *47*, 59–75. [[CrossRef](#)]
83. Xu, F.; Wang, H.; Peng, J.; Fu, X. Scale-aware feature pyramid architecture for marine object detection. *Neural Comput. Appl.* **2021**, *33*, 3637–3653. [[CrossRef](#)]
84. Buchner, J. ImageHash: A Python Perceptual Image Hashing Module. 2024. Version 4.3.1. Available online: <https://pypi.org/project/ImageHash/> (accessed on 4 September 2026).
85. Label Studio. Label Studio: Open Source Data Labeling Platform. 2020. Available online: <https://labelstud.io/> (accessed on 20 January 2025).
86. Yang, M.; Sowmya, A. An Underwater Color Image Quality Evaluation Metric. *IEEE Trans. Image Process.* **2015**, *24*, 6062–6071. [[CrossRef](#)] [[PubMed](#)]
87. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [[CrossRef](#)]
88. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Berkeley, CA, USA, 1967; Volume 1, pp. 281–297.
89. Liu, H.; Hussain, F.; Tan, C.L.; Dash, M. Discretization: An Enabling Technique. *Data Min. Knowl. Discov.* **2002**, *6*, 393–423. [[CrossRef](#)]
90. Hong, J.; Fulton, M.; Sattar, J. Trashcan: A semantically-segmented dataset towards visual detection of marine debris. *arXiv* **2020**, arXiv:2007.08097.
91. Roboflow. Underwater Pipes Dataset. 2021. Available online: <https://universe.roboflow.com/roboflow-100/underwater-pipes-4ng4t> (accessed on 25 October 2024).
92. Saleh, A.; Laradji, I.H.; Konovalov, D.A.; Bradley, M.; Vazquez, D.; Sheaves, M. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Sci. Rep.* **2020**, *10*, 14671. [[CrossRef](#)] [[PubMed](#)]
93. Jian, M.; Yang, N.; Tao, C.; Zhi, H.; Luo, H. Underwater object detection and datasets: A survey. *Intell. Mar. Technol. Syst.* **2024**, *2*, 9. [[CrossRef](#)]
94. Chen, L.; Huang, Y.; Dong, J.; Xu, Q.; Kwong, S.; Lu, H.; Lu, H.; Li, C. Underwater Optical Object Detection in the Era of Artificial Intelligence: Current, Challenge, and Future. *ACM Comput. Surv.* **2025**, *58*, 62. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.