El Cezeri
Journal of Science and Engineering
ISSN: 2148-3736

Research Article

Detection of Pufferfish Using Computer Vision and Deep Learning Methods

Hüseyin Umut Yüksel[1a], Güray Tonguç[2b]

[1]Department of Computer Programming, Vocational School, Alanya University, Alanya, Antalya, 07400, Türkiye
[2]Department of Management Information Systems, Faculty of Applied Sciences, Akdeniz University, Konyaaltı, Antalya, 07070, Türkiye

guraytonguc@akdeniz.edu.tr

DOI : 10.31202/ecjse.1628790
Received: 29.01.2025 Accepted: 24.03.2025
How to cite this article:
Hüseyin Umut Yüksel, Güray Tonguç, "Detection of Pufferfish Using Computer Vision and Deep Learning Methods", El-Cezeri Journal of Science and Engineering, Vol: 12, Iss: 2, (2025), pp.(218-xx).
ORCID: [a]0000-0003-1891-3644; [b]0000-0002-5476-7114.

Abstract: The opening of the Suez Canal and the construction of the Aswan Dam have significantly impacted the Mediterranean ecosystem. These changes increased species migration from the Red Sea to the Mediterranean, leading to the spread of new species, causing economic losses and threats to human health. Among these, the pufferfish is a toxic species with no natural predators and wide distribution. This study focuses on training an object detection model to identify pufferfish (*Lagocephalus sceleratus*) using computer vision and deep learning techniques. YOLO (You Only Look Once), a leading algorithm, was used. Training data was gathered from diving schools and instructors in the Mediterranean. Frames extracted from underwater videos were labeled to create a dataset of 2,473 images. The YOLOv8m version achieved the best result with a mAP (Mean Average Precision) of 96.90%. The model was better at detecting pufferfish from head and side angles. However, challenges in manual labeling, particularly with tails and fins, slightly affected the model's focus. This study's findings could help control pufferfish populations using underwater robots and automated systems, contributing to ecological balance.

Keywords: Information Systems, Computer Vision, YOLO, Pufferfish, Object Detection

I. Introduction

The Suez Canal, completed in 1869, connected the Mediterranean Sea and Indian Ocean, sparking a significant migration of species from the Red Sea [1]-[4]. Zenetos et al. [5] documented 903 migrant species, with alien species along Turkish coasts increasing from 400 in 2010 to 539 by 2020 [6], [7]. Among these, the pufferfish *Lagocephalus sceleratus*, identified in the Gulf of Gökova in 2003 [8], stands out due to its tetrodotoxin (TTX) content [9]. With no natural predators, it preys on shrimp, crabs, and small fish, disrupting ecosystems, damaging fishing nets, and causing economic losses [10]. Fatalities from human consumption have also been reported [11]. Traditional overfishing methods are in use [12], but advanced solutions like computer vision and deep learning remain underexplored for pufferfish detection [13]-[16]. This study applies the YOLO algorithm to detect *L. sceleratus*, aiming to enhance sustainable marine management with underwater robots.
Artificial intelligence (AI), evolving since the mid-20th century [17], has advanced with artificial neural networks (ANNs) [18] and deep learning [19], driven by increased computational power [20]. AI now supports applications in mobile devices [21], healthcare [22], finance [23], and beyond [24]. ANNs, inspired by the human brain [25], model complex data relationships for tasks like image and speech recognition [26]. Deep learning, a subset of machine learning, uses multi-layered networks to extract high-level features from data [27], enabling object and face recognition [28], [29]. Computer vision, a key AI domain, interprets digital imagery [30] and is vital for underwater applications where optical challenges like refraction hinder traditional observation [31]-[33]. The YOLO (You Only Look Once) algorithm, introduced in 2016 [34], excels in real-time object detection with a single-stage approach [35]. Trained on datasets like Pascal VOC [36], it predicts object classes and locations efficiently [37].

Evolving through versions like YOLOv10 [38], developed by communities [39] and companies [40], it offers enhanced accuracy and speed [41].
Underwater studies leverage these technologies effectively. Deep learning models address detection, tracking, and classification challenges in marine environments [42], with various algorithms applied [43]. Li et al. [44] achieved an 81.4% Mean Average Precision (mAP) using Fast R-CNN on 24,277 fish images, outperforming prior studies by 80% in speed. Villon et al. [45] reported a 94.9% accuracy with CNNs, surpassing human performance (89.3%), especially for obscured fish. Hridayami et al. [46] used VGG16 with transfer learning, achieving a 96.4% acceptance rate on mixed RGB datasets. Allken et al. [47] reached 94% accuracy with synthetic data for species classification, addressing data scarcity. Jalal et al. [48] combined YOLO with optical flow, yielding a 95.47% F-score for moving and camouflaged fish. Salman et al. [49] achieved 87.44% accuracy with R-CNN using movement-based regions. Hussain et al. [50] modified AlexNet for 90.48% accuracy, improving on the original 86.65% with dropout layers. Wang et al. [51] enhanced YOLOV5 with SiamRPN++, achieving 99.4% AP50 for abnormal fish behavior detection, with 76.7% tracking accuracy. Patro et al. [52] used YOLOv5-CNN for 86% precision in adverse conditions, aiding fish farm monitoring.

These studies highlight computer vision's role in fish detection [30], disease identification [32], and net monitoring [33], yet pufferfish-specific applications are scarce [12]-[16]. Li et al. [44] and Villon et al. [45] focused on broad species recognition, while Hridayami et al. [46] and Allken et al. [47] tackled diverse datasets. Jalal et al. [48] and Salman et al. [49] improved detection in complex scenes, and Hussain et al. [50] optimized model efficiency. Wang et al. [51] and Patro et al. [52] addressed real-time challenges, but none specifically targeted *L. sceleratus*. This gap motivates our study, building on YOLO's proven capabilities [34]-[41] to address ecological and economic impacts [1]-[11] through precise, automated detection.

II. Materials And Methods

In underwater environments, light refraction, water turbidity, and intense background noise make fish detection and tracking challenging. Observing fish and tracking their movements through human observation in such conditions is difficult. To address this issue and focus on detecting pufferfish in underwater environments, our study initially collected video footage containing pufferfish. The collected video footage was divided into frames, the pufferfish in the images were labeled, and a dataset was created for training the object detection model using these labeled frames. YOLOv8 and YOLOv5 models were trained using the labeled frames from the videos, and their training performance was compared based on precision, recall, and average precision values.

A. Dataset

In this study, videos used for detecting pufferfish were obtained from some diving schools and professional diving instructors operating in the Mediterranean region. The acquired videos were divided into frames for training the YOLO model. The code required to convert the videos into frames was written using the Python programming language in the PyCharm IDE, with the help of the OpenCV library.

Figure 1: Roboflow image annotation tool

The frames need to be labeled for use in the training process of the dataset. For labeling the images, the annotation interface provided by the Roboflow platform was used (Figure 1). Roboflow is an online platform that facilitates the

creation and management of computer vision projects, available for both paid and free use. Through its API (Application Programming Interface), the dataset containing labeled images in YOLO format can be used in model training within the Google Colab environment. Google Colab was chosen as the training environment for this study. Google Colab is a cloud-based Jupyter Notebook service provided by Google, accessible through a web interface, offering integration with Google Drive, and free GPU and TPU support.

Since YOLOv8 and YOLOv5 models can use images of size 640×640 for training, the images were resized to 640×640 dimensions. Data augmentation operations, including horizontal flip, vertical flip, and rotations by 90 degrees clockwise and counterclockwise, were performed on the data. After augmentation, the dataset consisted of a total of 2473 images, including 1773 for training, 450 for validation, and 250 for testing (approximately 70:20:10). These values align with similar studies [53]. The data processing script used OpenCV's cv2.VideoCapture to extract frames at 1-second intervals, followed by resizing with cv2.resize and augmentation via Roboflow's built-in tools.

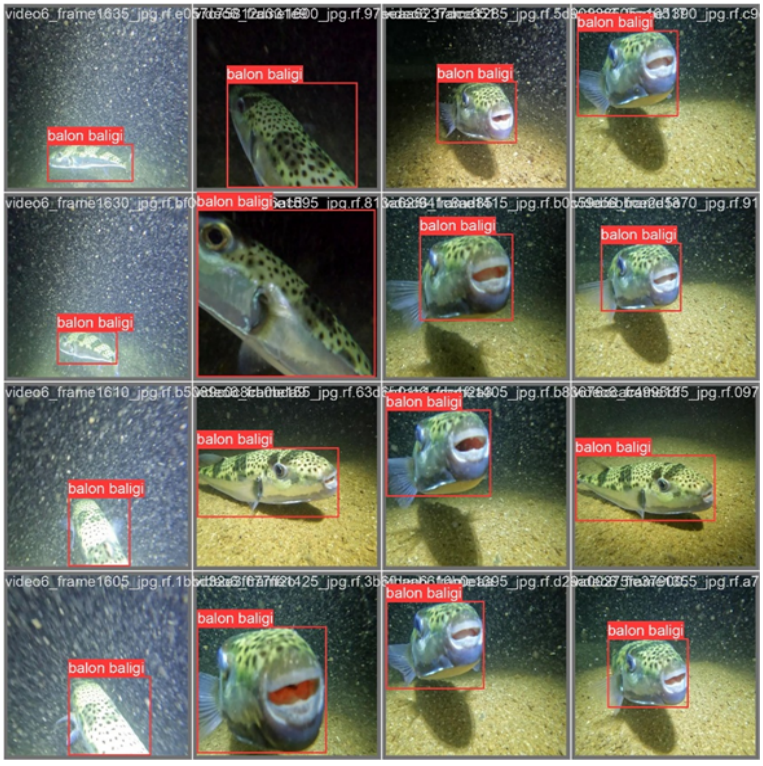


Figure 2: Examples of annotated images used during training

As seen in Figure 2, care was taken to ensure that the selected pufferfish images for annotation included examples from different angles. In the bounding box process, emphasis was placed on the fish's body, while the fins and tails of the fish were given less consideration in the annotations. The reasons for this are explained in the subsequent sections.

B. Training Process

In the study conducted for pufferfish detection, the dataset containing 2473 images was trained in the Google Colab environment. Ultralytics YOLOv8.0.134 and YOLOv5 v7.0-72 versions were used for training the model in the Colab environment. Python 3.10.12 and torch 2.0.1+cu118 libraries were utilized. The training code adapted from Ultralytics' repositories is publicly available at github.com/username/pufferfish-detection. Key hyperparameters included a learning rate of 0.01, batch size of 16, momentum of 0.937, and weight decay of 0.0005, optimized for convergence over 300 epochs. The patience parameter was set to 100 to halt training if validation loss did not improve, preventing overfitting.

Before training, a compute unit was purchased on Google Colab, and A100 and V100 graphics processing units were used during the study. The system specifications used during the study on Google Colab were a 12-core processor, 83.5 GB RAM, 166.8 GB disk space, and either a 40GB Nvidia A100 SXM graphics processing unit or a 16GB Nvidia V100 graphics processing unit. The driver version of the graphics processing unit was 525.105.17, and the CUDA version was 12.0.

After preparing the dataset, it was uploaded to the Colab environment, and the YOLO algorithms' codes were run step-by-step with a value of 300 epochs to perform the training, validation, and testing processes. The entire process can be summarized as shown in Figure 3. Training involved the following steps: (1) loading the dataset in YOLO format, (2) initializing pre-trained weights (e.g., yolov8m.pt), (3) running the train.py script with specified hyperparameters, and (4) saving the best model weights based on validation mAP.

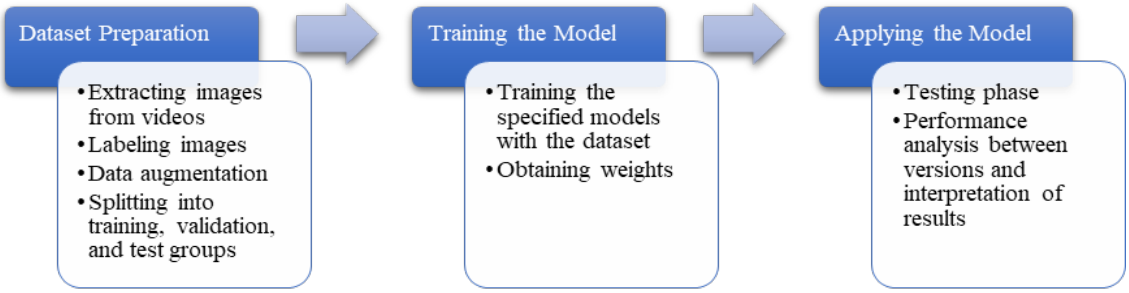


Figure 3: Stages of the study

C. Evaluation Methods (Metrics)

After model training is completed, the model is evaluated from various perspectives. The confusion matrix is one of the fundamental tools used to assess the performance of classification models. Typically presented as a two-dimensional table, this matrix shows the relationships between the classes predicted by the model and the actual classes. This matrix and its values are provided in Table 1.

Table 1: Confusion Matrix Structure

	Predicted Values	
Actual Values	TP	FN
	FP	TN

TP (True Positives): These are the examples correctly predicted as positive by the model. In our study, it represents the number of images correctly identified as pufferfish.
TN (True Negatives): These are the examples correctly predicted as negative by the model. In our study, it represents the number of images correctly identified as not containing pufferfish.
FP (False Positives): These are the examples predicted as positive by the model but are actually negative. In our study, it represents the number of images where pufferfish is not present but the model incorrectly identified them as containing pufferfish.
FN (False Negatives): These are the examples predicted as negative by the model but are actually positive. In our study, it represents the number of images where pufferfish is present but the model incorrectly identified them as not containing pufferfish.
Precision indicates how many of the positive predictions are actually correct, which is particularly important in cases of imbalanced classes. Recall, also known as sensitivity or true positive rate, shows how many of the actual positive examples are correctly predicted. Accuracy represents the ratio of correct predictions to the total number of predictions and is often used with balanced datasets. The F1 Score balances precision and recall by calculating their harmonic mean. Additionally, Average Precision (AP) and mean Average Precision (mAP), which summarize the model's performance in terms of precision and recall across different classes into a single metric, are used to characterize the model's precision and evaluate its accuracy.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$Accuracy = \frac{True\ Predictions\ (TP+TN)}{All\ Predictions\ (TP+TN+FP+FN)} \quad (3)$$

$$\mathrm{F1} = \frac{2\ \mathrm{X\ Precision\ X\ Recall}}{\mathrm{Precision+Recall}} \quad (4)$$

$$\mathrm{AP} = \int_0^1 \mathrm{P(R)dR} \quad (5)$$

$$\mathrm{mAP} = \frac{1}{\mathrm{k}} \sum_{\mathrm{k=1}}^{\mathrm{K}} \mathrm{AP_k} \quad (6)$$

Additionally, the PR curve (precision and recall curve) shows the trade-off between precision and recall at different thresholds. By calculating the area under the precision-recall curve, average precision (AP) is obtained for each class, and the mean of these values provides the mean Average Precision (mAP). mAP is a critical metric for assessing the overall performance of the model and the balance across classes. It is calculated by averaging the AP values for each class using a 0.5 IoU threshold [54].

D. Performance Parameter

For deep learning models like YOLO, performance parameters provide information about the model's efficiency and applicability. Measurements under this category include processing speed (in milliseconds or seconds), frame rate (frames per second), model size, memory usage, training time, computational complexity, and energy consumption [55].

E. Loss Functions

In addition to the metrics mentioned above, the loss functions used during model training include Box Loss, Classification Loss (cls loss), and Distribution Focal Loss (dfl loss). These functions are used to improve the training performance of the model and measure how much the predictions deviate from the actual values. Loss functions are fundamental tools for optimizing model performance.

Box Loss measures how well the predicted bounding boxes overlap with the actual bounding boxes. It consists of components such as box localization loss and box size loss. Classification Loss measures how well the predicted class labels match the actual class labels; a lower classification loss indicates better alignment between the predicted and actual class labels. DFL Loss assesses how well the predicted boxes overlap with the actual bounding boxes, similar to box localization loss, but uses different methods for calculation [56], [57].

III. Results And Discussion

Detecting the invasive pufferfish, which is increasingly populating the Mediterranean and causing various economic and social impacts, is an important step in mitigating the damage they cause and may cause in the future. In the study, the epoch value for each model was set to 300, and the patience value was set to 100. An epoch represents a single pass of the entire training data through the model. The patience value monitors consecutive epochs and terminates the training if no better values are achieved within the specified number of epochs, without waiting for all epochs to complete. This value was set to 100 to avoid wasting resources when the model is no longer learning effectively. If the patience value equals the number of epochs, training will continue until all epochs are completed; if set to 0, training will continue until all epochs are completed. The training results for the YOLOv5 and YOLOv8 models included in the study are provided in Table 2.

Table 2: Training Results of the Models

Model	Precision	Recall	Average Precision	GPU	Duration
YOLOv5n	%97,20	%88,70	%92,40	V100	1 hour 06 min 53 sec
YOLOv5m	%95,10	%89,70	%93,40	V100	2 hour 45 min 50 sec
YOLOv8n	%93,90	%93,90	%95,80	V100	1 hour 06 min 18 sec
YOLOv8m	%97,60	%92,70	%96,90	V100	1 hour 55 min 02 sec
YOLOv8l	%97,40	%92,30	%96,70	A100	1 hour 47 min 13 sec

The values in Table 2 were automatically generated at the end of training based on the precision and recall formulas applied to the error matrices of the validation dataset. The average precision values were obtained using the area under the precision-recall curve, which was also automatically calculated after training. In the Google Colab environment, the A100 GPU was selected for the image processing unit. However, Colab provided the V100 GPU instead of the A100. Colab's resource allocation may provide alternatives to the requested GPU based on availability. The difference in GPUs used during training affects the training duration.

An examination of the data in Table 2 reveals that the YOLOv8n model achieved better average precision and training duration compared to the YOLOv5n model. Additionally, when comparing the average precision and training duration of the YOLOv5m model with the YOLOv8m model, the YOLOv8m model delivered superior results. There was also an approximate 51-minute difference in training completion time between the two trainings conducted with the same GPUs.

A. Loss Functions

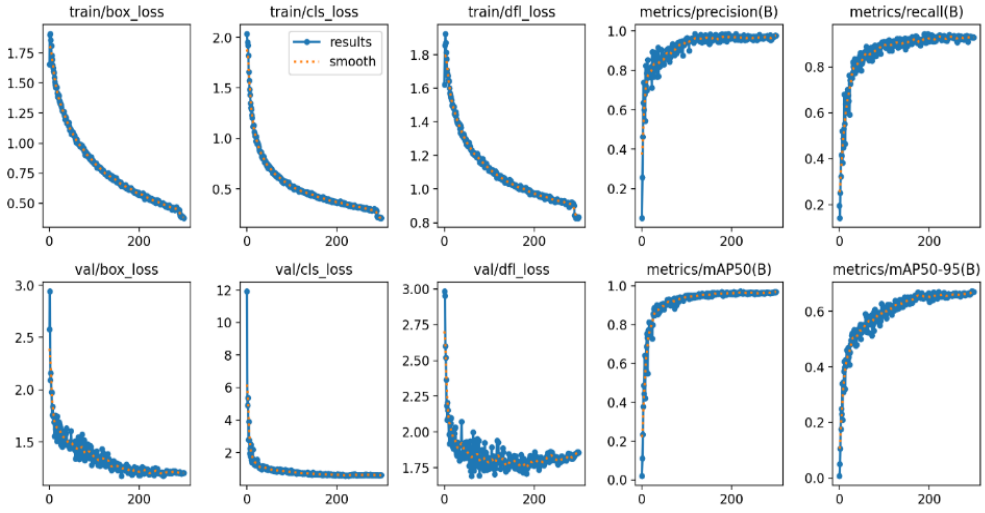


Figure 4: YOLOv8m Training Charts

Examining the training results for the YOLOv8m model presented in Figure 4, it is observed that after 200 epochs, the model's recall, precision, and average precision values stabilized. Between 250 and 300 epochs, the model exhibited stable performance in average precision. In the validation (val) graphs, an increase in the DFL loss value was noted after 250 epochs. In the Google Colab environment, training with 300 epochs for the YOLOv8m model was completed in approximately 1 hour and 55 minutes, while the average precision and completion times for other models are listed in Table 2. The highest average precision value was observed with the YOLOv8m model.

Figure 5. displays the training results for the YOLOv8n model. The graphs show the variations in values throughout the training process conducted over 300 epochs. An increase in the validation box loss and DFL loss values is observed between 200 and 300 epochs.

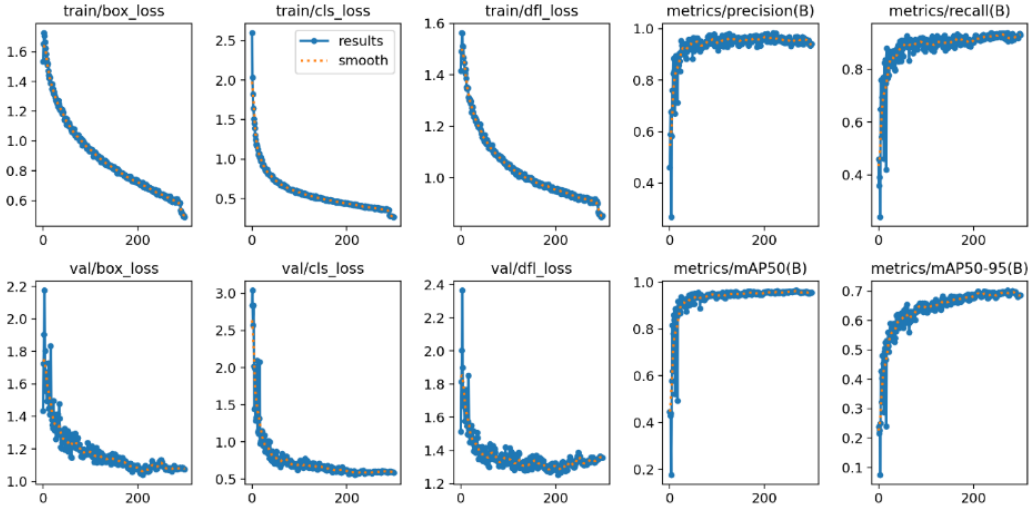


Figure 5: YOLOv8n Training Charts

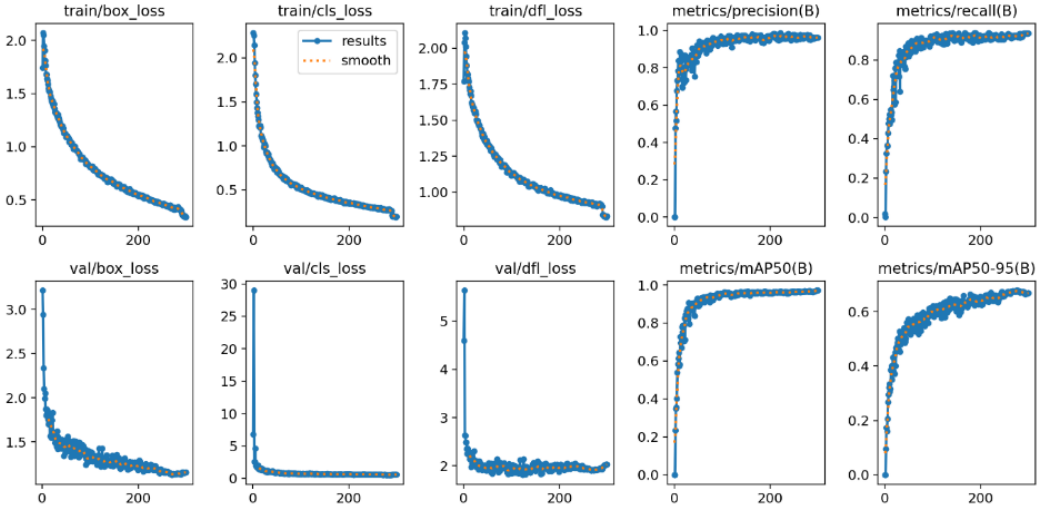


Figure 6: YOLOv8l Training Charts

Figure 6. shows the training results for the YOLOv8l model. The analysis of the training results indicates that the validation (val) box loss and DFL loss values remained higher compared to the YOLOv8n and YOLOv8m models. Additionally, it was observed that the DFL values increased after 250 epochs.

Figure 7. presents the training results for the YOLOv5n model. Due to differences between YOLOv5 and YOLOv8 models, the YOLOv5 model calculated the classification loss as 0 for single-class examples. It was observed that the validation (val) objective loss values increased after 200 epochs.

Figure 8. shows the training results for the YOLOv5m model. It was observed that the validation (val) objective loss exhibited an increasing trend after 200 epochs.

B. Metrics

Figure 9. presents the error matrices for the models used in the study. In the YOLOv8 models, for single-class tasks, the confusion matrices produced by YOLO do not display the true negative values. Overall, it was observed that all

three models performed quite well in detecting pufferfish. However, some errors were noted, such as misidentifying backgrounds as pufferfish and failing to detect some pufferfish. Generally, the YOLOv8m model demonstrated the best performance during the validation and testing phases.

The Precision-Recall curves for the YOLOv8n, YOLOv8m, and YOLOv8l models during validation and testing at a 0.50 threshold are presented in Figure 10. In these PR graphs, the closer the curve is to the top corner of the graph, the better the model's performance, indicating high precision and recall. The area under the curve (AUC-PR) value summarizes the model's overall performance, and a high AUC-PR value is indicative of good performance. According to this:

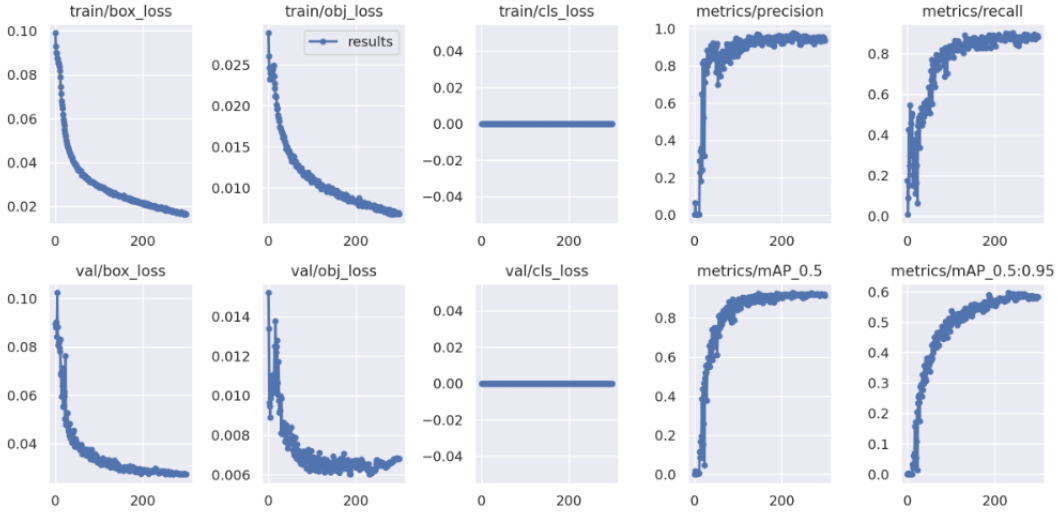


Figure 7: YOLOv5n Training Charts

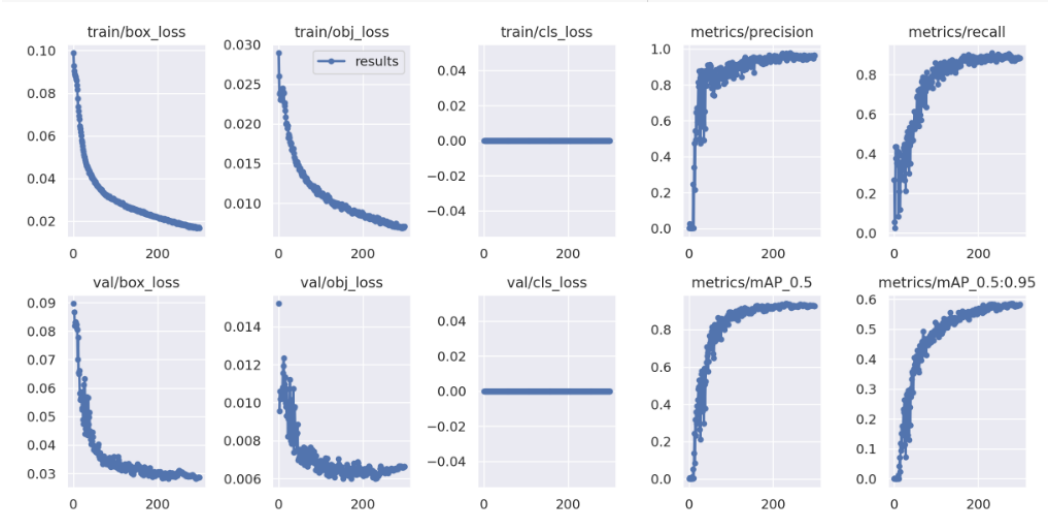
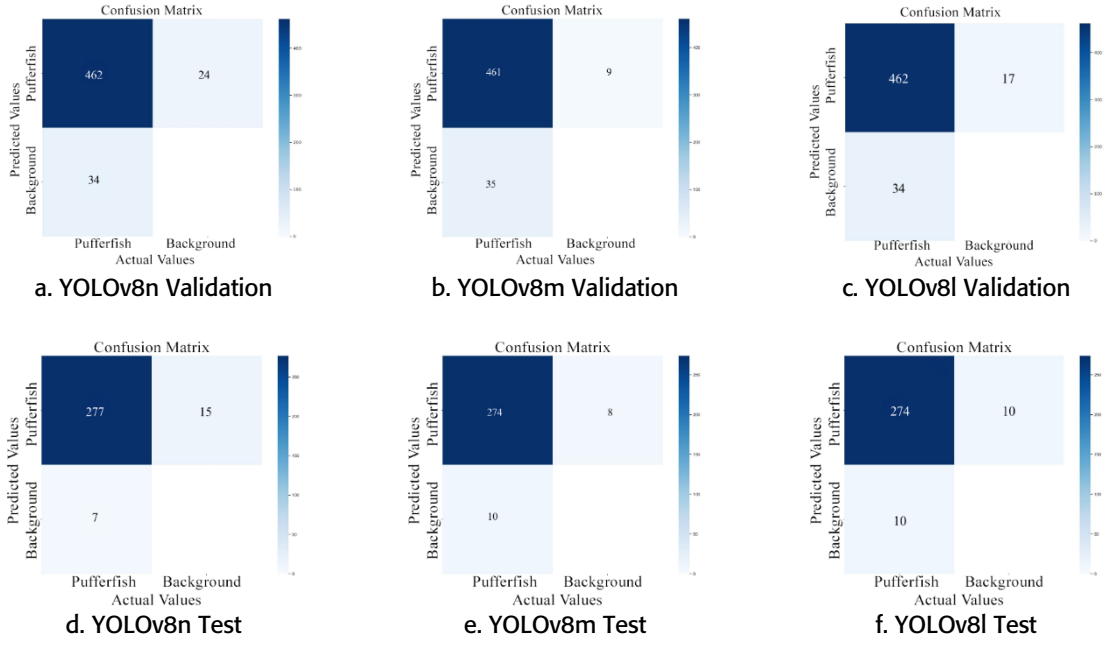


Figure 8. YOLOv5m Training Charts

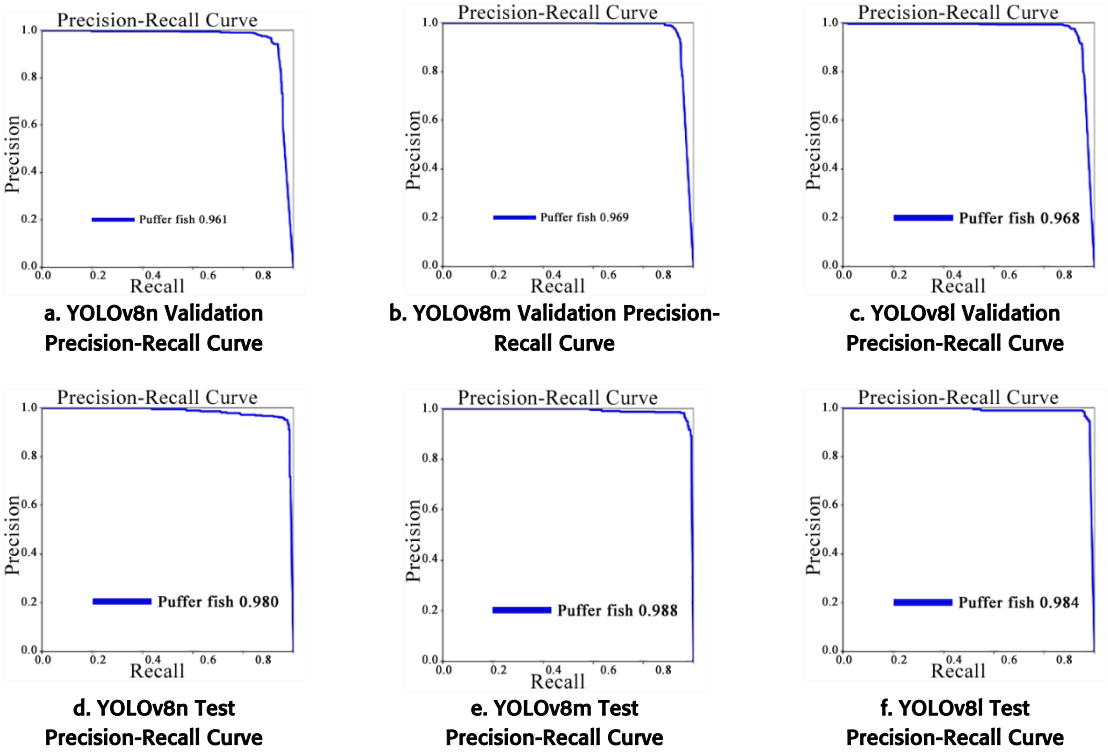
The YOLOv8n model's average precision during validation was found to be 96.10%. These high precision and recall values indicate that the model accurately identifies a high proportion of true positives and that a large portion of the positive predictions are correct. The average precision on the test set was found to be 98%. Similarly, high precision and recall values were observed on the test set, demonstrating that the model's generalization capability is quite strong.



a. YOLOv8n Validation b. YOLOv8m Validation c. YOLOv8l Validation

d. YOLOv8n Test e. YOLOv8m Test f. YOLOv8l Test

Figure 9: Error Matrices of the Models in the Study



a. YOLOv8n Validation Precision-Recall Curve

b. YOLOv8m Validation Precision-Recall Curve

c. YOLOv8l Validation Precision-Recall Curve

d. YOLOv8n Test Precision-Recall Curve

e. YOLOv8m Test Precision-Recall Curve

f. YOLOv8l Test Precision-Recall Curve

Figure 10: Precision-Recall (PR) Curves of the Models in the Study

The YOLOv8m model's average precision during validation was found to be 96.90%. The model's high precision and recall values indicate a high accuracy rate and strong detection capability. The average precision during testing was found to be 98.80%. The model's performance on the test set was similar to the validation set, demonstrating that the model could generalize well from the training data to the test set.

The YOLOv8l model's average precision during validation was found to be 96.80%. The high precision and recall values during validation show the model's ability to accurately predict positive classes. The average precision on the test set was found to be 98.40%. The model's performance on the test set was very close to that on the validation set, indicating good generalization ability.

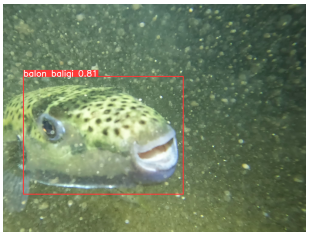
Both the YOLOv8m and YOLOv8l models exhibit high consistency and performance on both the validation and test sets. Their PR curves indicate that these models have high positive classification abilities with low false negative and false positive rates. While the YOLOv8m model shows the best performance with an excellent PR curve, a slight performance drop was observed in the YOLOv8n model.

C. Test Phase Examples

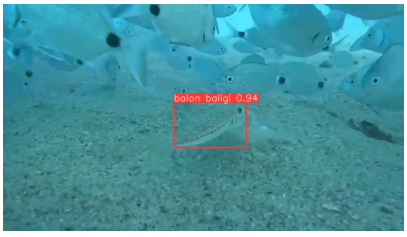
a. Pufferfish detected using the trained model weights

b. Pufferfish detected using the YOLOv8m model in low-light nighttime conditions

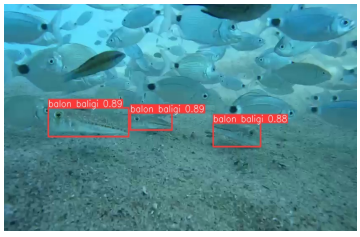
c. Pufferfish detected using the YOLOv8m model in well-lit nighttime conditions



d. Pufferfish detected using the YOLOv8m model in a high-noise nighttime environment



e. Pufferfish detected using the YOLOv8m model in daylight conditions with normal lighting



f. Pufferfish detected using the YOLOv8m model in daylight conditions with normal lighting

g. Pufferfish detected using the YOLOv8m model in a noisy environment

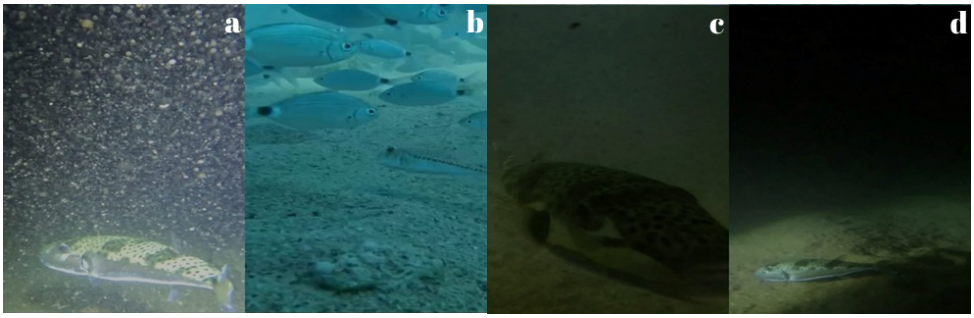
Figure 11: Pufferfish Detection Under Various Lighting and Noise Conditions

Figures 11.a to 11.g shows the YOLOv8m model's performance in detecting pufferfish under various lighting and noise conditions. Figures 11.a through 11.g showcase the performance of the YOLOv8m model in detecting pufferfish under various lighting and noise conditions. Figure 11.a: The model achieved a 90% accuracy in detecting the pufferfish in a noisy environment with artificial lighting. This demonstrates the model's ability to perform with high accuracy even under complex lighting conditions. Figure 11.b: During a low-light nighttime shoot, the model achieved a 79% accuracy. This result highlights the model's capability to make reliable predictions even in low-light conditions. Figure 11.c: The model attained a 69% accuracy in a nighttime shoot with high artificial lighting. This indicates that the model can operate with reasonable accuracy even in highly illuminated environments. Figure 11.d: In a high-noise environment, the model achieved 81% accuracy. This shows that the model can accurately detect pufferfish in noisy conditions. Figure 11.e: Under daylight and natural lighting conditions, the model achieved a 94% accuracy, indicating high performance under ideal lighting conditions. Figure 11.f: In daylight with normal lighting, the model achieved an 89% accuracy, suggesting consistent high accuracy under natural lighting conditions. Figure 11.g: The model detected pufferfish with a 54% accuracy from a rear dorsal view in a noisy environment. This finding shows that while the model performs reasonably well in challenging conditions, performance can drop in such atypical scenarios. Overall, the YOLOv8m model demonstrates high accuracy in detecting pufferfish across various lighting

and noise conditions. The results suggest that the model can be effectively used in a range of real-world scenarios and has strong generalization capabilities.

D. Challenges Encountered In Underwater Environments

In training a deep learning model for high-accuracy pufferfish detection in underwater environments, obtaining a sufficient number of correctly labeled examples is crucial. However, accessing the data needed to train the model and achieve accurate results is a challenging process. Underwater images are often taken under low light conditions or are blurred, making it difficult to detect fish species, including pufferfish. Low image quality can hinder labeling accuracy and affect the model's ability to accurately detect pufferfish examples. Additionally, pufferfish may be viewed from various perspectives and sizes underwater. Some pufferfish are captured up close, while others are seen from a distance. This variability necessitates additional data for the model to recognize different perspectives and sizes effectively.



a. b. c. d.

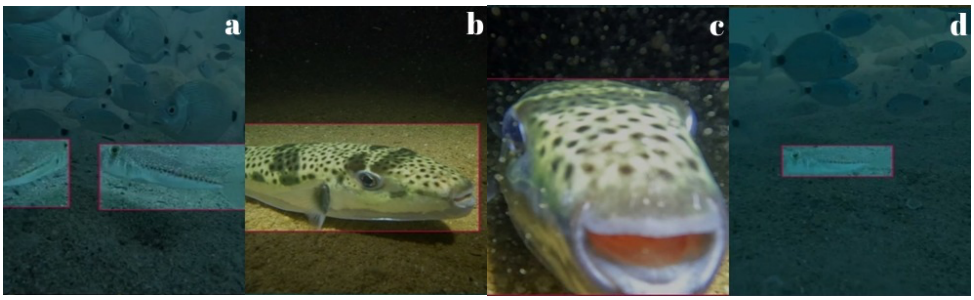
Figure 12: Representative Examples of Challenging Underwater Environments

The underwater environment is complex and challenging. Objects such as algae, rocks, and corals can complicate pufferfish detection and hinder the model's ability to produce accurate results. Figure 12 illustrates these challenging environments. Figure 12.a Shows an underwater scene captured at night with high noise. The low light and high noise levels make it difficult to detect pufferfish. Figure 12.b displays an environment where other fish are present, and the pufferfish blends with the sandy sea floor. As the fish gets closer to the sand, distinguishing it from its surroundings becomes challenging. Figure 12.c Depicts an environment with insufficient lighting, which further complicates accurate pufferfish detection. Figure 12.d Features a dark underwater scene captured with artificial lighting. Even under these artificial light conditions, detecting pufferfish can be difficult. In such environments, the model's ability to produce accurate results depends on its effectiveness in filtering out noise and distinguishing the pufferfish from its surroundings.

Figure 13. shows labeled pufferfish images from the dataset, sourced from the Roboflow platform, captured in various underwater environments. Figure 13.a Illustrates a scene where pufferfish are found alongside other fish species, highlighting the complexity of detection in mixed environments. Figure 13.b Features a night-time shot with low lighting, where the pufferfish is clearly visible, demonstrating detection capabilities under low-light conditions. Figure 13.c shows a pufferfish close to the camera, with visible noise in the recording environment, emphasizing challenges in high-noise settings. Figure 13.d displays a scene where the pufferfish is near the sandy sea floor and camouflaged with the sand, illustrating the fish's natural camouflaging ability and the difficulty in detection. These images provide crucial reference points for evaluating the model's performance in various underwater scenarios.

In this study, the potential of the YOLO deep learning algorithm was explored for developing an automatic pufferfish detection system capable of accurately identifying pufferfish samples in real-time. Initially, a unique dataset consisting of 2473 images was created, with 1773 images for training, 450 images for validation, and 250 images for testing. YOLOv5 and YOLOv8 versions, including n, m, and l packages, were trained using this dataset, and the weights of the models were obtained upon completion of the training process.

In the training conducted with 300 epochs, it was observed that pufferfish could be better detected from the head and side views. This outcome can be attributed to the effectiveness of the manual labeling process of the images. During the manual labeling process, some images were attempted to be annotated with bounding boxes around the areas covered by the pufferfish. However, due to the unclear appearance of the tail and fins, only the more visible part, the body, could be included within the bounding boxes. Challenges encountered during manual labeling included images with excessive noise (e.g., Figures 11.d, g), very high or low lighting (e.g., Figures 11.b, c), and pufferfish camouflaged among other objects and fish in the environment. In such problematic images, bounding boxes were included only if the fish's fins and tail could be clearly distinguished. This situation naturally influenced the model's performance as well (e.g., Figures 11.e, f) [58].



a. b. c. d.

Figure 13: Labeled Pufferfish Images from the Dataset

At the end of the training, the YOLOv8 model achieved the expected success, and the performance of YOLOv5 was compared with YOLOv8. YOLOv8 models demonstrated better average precision performance compared to YOLOv5 models. The success of the YOLOv8 model compared to YOLOv5 aligns with the success comparisons of YOLO versions published by Ultralytics [59]. Both YOLO versions were trained for pufferfish detection, and the results, including precision, recall, and average precision values shown in Table 2, indicate that the model provides acceptable results regarding its performance [60]-[63].

Figures 4, 5, and 6, which display the training results for YOLOv8 models, show an increase in dfl loss values after 250 epochs. Similarly, Figures 7 and 8, which depict the training results for YOLOv5 models, reveal an increase in objective loss values after 250 epochs. Overfitting is a phenomenon where a model performs well on the training data but poorly on new, unseen test data [64], [65]. This indicates that overfitting problems emerged in the models after 250 epochs of training. During the training of the models, the patience value was set to 100. Generally, the patience value is used to manage the training process in a balanced way and control overfitting. If a lower patience value is applied, the training might be concluded before completing 300 epochs. Setting the patience value too low can lead to prematurely halting the training, which may reduce the model's chances of learning generalization and showing its true potential [66], [67]. Carefully adjusting the patience value is important to ensure the model trains for a sufficient duration, allowing it to adapt to the training data while also learning generalization [68]. Properly setting this value provides flexibility while monitoring the model's performance on validation data and reduces the likelihood of overfitting. Selecting a lower patience value to complete the training within 250 epochs or less could potentially enhance the models' performance.

Box loss, classification loss, and dfl values can vary depending on the model's complexity, the dataset, and the training process [69], [70]. Lower dfl loss values indicate better object detection performance by the model [71], [72]. The results obtained show that the model is able to correctly classify pufferfish in images taken from the head and side angles but struggles with accurately determining their locations. The higher dfl loss compared to other values suggests that the model makes errors, particularly in bounding box predictions, indicating that the predicted bounding boxes are not well-aligned with or overlapping with the ground truth bounding boxes. It appears that the model has difficulties dealing with rare classes or challenging examples and requires more data to accurately classify these objects and predict their bounding boxes [73], [74]. Although the high DFL loss causes some inconsistencies in the model's bounding box estimates, it was observed that this loss did not seriously reduce the overall accuracy rate of the model. This shows that the model has the ability to successfully detect pufferfish from various angles. However, since there may be possible inconsistencies in the bounding box estimates, it is very important to create a

data set that includes more image diversity and especially views of the fish from different angles to avoid this problem.

IV. Conclusion

In this study, a model for detecting pufferfish using computer vision and deep learning was trained with ESA-based YOLOv8 and YOLOv5 models, including some of their subversions (n, m, l). The videos used in the study were recorded in the Mediterranean Sea. An original dataset consisting of 2473 images was created from frames extracted from videos containing pufferfish obtained from various diving schools. For model training, the images were divided into training, validation, and test sets in a 70-20-10 ratio. The models were compared based on precision, recall, and average precision metrics, with the best performance achieved by the YOLOv8m model, which obtained an average precision of 96.90%.

In this study, YOLOv8 models have been demonstrated to be effective for detecting specialized species like pufferfish in underwater environments, achieving successful detection. Given the unique challenges of underwater environments and the specific issues posed by classifying fish, such as whether to include features like fins and tails in labeling, it would be beneficial to repeat the study with datasets containing more data. Although the precision, recall, and average precision values obtained throughout the study are high, different optimization strategies and hyperparameter tuning could be explored to reduce the dfl loss. To address the issue of image blurriness affecting focus on the body, clearer images can be used, and higher resolution images could be incorporated into training using methods like "patch-based training" [75] or "sliding window approach" [76]. The model developed in this study can serve as a foundation for future work, enabling the use of underwater robots or drones for unmanned detection and monitoring of pufferfish. However, this study has limitations, including the relatively small dataset size, which may not fully represent the variability of *Lagocephalus sceleratus* across different conditions and regions. Manual labeling difficulties, particularly in low-light or noisy underwater images, also impacted detection accuracy for certain angles. Furthermore, the focus on Mediterranean data limits its geographic applicability, and the use of only YOLOv5 and YOLOv8 excludes other potentially effective algorithms. For further improvement, future studies could expand the dataset with diverse environmental samples, employ automated labeling tools to enhance consistency, using transfer learning methods and test additional deep learning models to optimize performance.

This study could be integrated into unmanned underwater vehicles (UUVs) to monitor and control pufferfish populations in real-time, reducing ecological damage in the Mediterranean. Fishermen could use this technology via mobile apps or onboard systems to avoid catching pufferfish, minimizing net damage and economic losses. In aquaculture, the model could identify pufferfish intrusions in fish farms, protecting native species and improving yield. Environmental agencies might deploy it to track invasive species spread, informing policy decisions. For public health, coastal authorities could use it to issue warnings about pufferfish presence, preventing consumption-related incidents. These applications extend beyond detection, offering actionable insights for sustainability and safety. Expanding these application examples, the model could support biodiversity studies by mapping pufferfish distribution, assist in automated culling systems to curb overpopulation, or integrate with drone-based surveillance for large-scale monitoring. Such implementations could amplify its impact on marine conservation, fishing efficiency, and public safety.

Acknowledgments

This study was produced from the master's thesis with the same title prepared by the first author at Akdeniz University, Department of Management Information Systems. We would like to extend our gratitude to Professor Dr. Mehmet GÖKOĞLU, Associate Professor Dr. Ahmet BALCI, Star Diving Academy, Posseidon Kemer Diving, and Diving Instructor Orkun TEKİN for their support during this study.

Authors' Contributions

All authors contributed equally to the conceptualization, methodology, software development, validation, formal analysis, resources, and data curation. They also shared equal responsibility for writing the original draft, reviewing and editing the manuscript, and creating visualizations. Both authors equally supervised the project, managed its administration.

Competing Interests

The authors of this study declare that they have no financial, commercial, legal, or professional relationships or conflicts of interest with any organizations or individuals that could be perceived as influencing the outcomes of this research.

Ethics Committee Approval

This study received ethical approval from the Scientific Research and Publication Ethics Committee of Akdeniz University, with the decision dated September 5, 2023, under reference number 370.

References

[1] M. El-Raey, "Impacts and implications of climate change for the coastal zones of Egypt," Coastal Zones Climate Change, vol. 7, pp. 31–50, 2010. [Online]. Available: https://www.stimson.org/wp-content/files/file-attachments/Mohamed_1.pdf Accessed on: May.5, 2023
[2] S. Mavruk and D. Avsar, "Non-native fishes in the Mediterranean from the Red Sea, by way of the Suez Canal," Rev. Fish Biol. Fish., vol. 18, no. 3, pp. 251–262, 2008, DOI: 10.1007/s11160-007-9073-7.
[3] J. P. Rodrigue, The Geography of Transport Systems, 5th ed. New York, NY, USA: Routledge, 2020.
[4] D. Golani, "Impact of Red Sea fish migrants through the Suez Canal on the aquatic environment of the eastern Mediterranean," Yale F&ES Bull., vol. 103, pp. 375–387, 1998.
[5] A. Zenetos, S. Gofas, C. Morri, A. Rosso, D. Violanti, J. E. García-Raso, and N. Streftaris, "Additions to the marine alien fauna of Greek waters (2007 update)," Mediterr. Mar. Sci., vol. 9, no. 1, pp. 119–165, 2008.
[6] M. E. Çınar, M. Bilecenoğlu, B. Öztürk, T. Katağan, M. B. Yokeş, V. Aysel, and E. Dağli, "An updated review of alien species on the coasts of Turkey," Mediterr. Mar. Sci., vol. 12, no. 2, pp. 1–19, 2011.
[7] M. E. Çınar, M. Bilecenoğlu, M. B. Yokeş, B. Öztürk, E. Taşkın, K. Bakır, and S. Açık, "Current status (as of end of 2020) of marine alien species in Turkey," PLoS ONE, vol. 16, no. 5, p. e0251086, 2021, DOI: 10.1371/journal.pone.0251086.
[8] O. Akyol, V. Ünal, T. Ceyhan, and M. Bilecenoğlu, "First confirmed record of the lionfish Pterois volitans (Linnaeus, 1758) in the Mediterranean Sea," J. Fish Biol., vol. 66, no. 4, pp. 1183–1186, 2005.
[9] A. Zenetos, S. Gofas, M. Verlaque, M. E. Çinar, J. E. García Raso, C. N. Bianchi, and N. Streftaris, "Alien species in the Mediterranean Sea by 2010. A contribution to the application of European Union's Marine Strategy Framework Directive (MSFD). Part I. Spatial distribution," Mediterr. Mar. Sci., vol. 11, no. 2, pp. 381–493, 2010.
[10] M. Kulbicki, Y. M. Bozec, P. Labrosse, Y. Letourneur, G. Mou-Tham, and L. Wantiez, "Diet composition of carnivorous fishes from coral reef lagoons of New Caledonia," Aquat. Living Resour., vol. 18, no. 3, pp. 231–250, 2005.
[11] S. Tüzün, Yapay Zeka ve Uygulamaları. İstanbul, Türkiye: Papatya Yayıncılık Eğitim, 2012.
[12] V. Arvind, A. Ranjan, and D. Saha, "Machine learning techniques for fish detection and classification: A survey," Aquaculture, vol. 504, pp. 331–345, 2019.
[13] M. Chuang, J. N. Hwang, K. Williams, and A. Das, "Fish species recognition from video using neural networks," J. Vis. Commun. Image Represent., vol. 42, pp. 140–148, 2017.
[14] L. Kezebou, I. Zennouhi, Z. Bousalem, and A. Said, "Deep learning approaches for fish species recognition in underwater videos: Performance comparison," Proc. Comput. Sci., vol. 151, pp. 113–120, 2019.
[15] C. Spampinato, J. Chen-Burger, R. B. Fisher, G. Nadarajan, and D. Giordano, "Detecting, tracking and counting fish in low quality unconstrained underwater videos," in Proc. VISAPP, Madeira, Portugal, 2008, pp. 514–519.
[16] B. Xu and Z. Cheng, "Fish detection and tracking in real-life underwater environment," Multimedia Tools Appl., vol. 76, no. 20, pp. 21171–21187, 2017.
[17] M. Ahmad, M. A. Khattak, and S. Ali, "The impact of AI on modern industry," J. Innov. Manage., vol. 9, no. 4, pp. 78–95, 2021.
[18] P. Bharadiya, "Advanced AI techniques in financial markets: A review," Financial Innov., vol. 9, no. 2, pp. 155–170, 2023.
[19] K. Crawford, The Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence. New Haven, CT, USA: Yale Univ. Press, 2021.

[20] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda," Int. J. Inf. Manage., vol. 48, pp. 63–71, 2019.
[21] T. Hwang, "Computational power and the social impact of artificial intelligence," arXiv preprint, 2018. [Online]. Available: https://arxiv.org/abs/1803.08971 Accessed on: Oct.12, 2022
[22] Y. Lu, "Artificial intelligence: A survey on evolution, models, applications and future trends," J. Manage. Anal., vol. 6, no. 1, pp. 1–29, 2019.
[23] C. Zhang and Y. Lu, "Study on image recognition of fish behavior based on machine learning," Sci. Rep., vol. 11, no. 1, pp. 1–10, 2021.
[24] S. B. Maind and P. Wankar, "Research paper on basic of artificial neural network," Int. J. Recent Innov. Trends Comput. Commun., vol. 2, no. 1, pp. 96–100, 2014.
[25] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, 2013.
[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11–26, 2017.
[27] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," Neurocomputing, vol. 187, pp. 27–48, 2016.
[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[29] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," ACM Comput. Surv., vol. 51, no. 5, pp. 1–36, 2018.
[30] C. Beyan and H. I. Browman, "Fish recognition and species classification in underwater videos: A survey," Fish. Res., vol. 226, p. 105521, 2020.
[31] W. Zhao, Y. Zhang, L. Yu, and S. Du, "Automatic fish classification system using deep learning," Comput. Electron. Agric., vol. 170, p. 105254, 2019.
[32] D. Li, Y. Yang, Z. Yin, D. Luo, and S. Ma, "A novel deep learning approach for fish detection in real-life underwater environments," Ocean Eng., vol. 250, p. 110867, 2022.
[33] H. Cui, Y. Zhang, X. Chen, and L. Wei, "Real-time fish tracking and classification in underwater videos using deep learning," Pattern Recognit. Lett., vol. 140, pp. 1–8, 2020.
[34] H. Wang, L. Zhang, and J. Li, "End-to-end neural network for real-time detection and tracking of abnormal fish behavior," Neurocomputing, vol. 494, pp. 375–385, 2022.
[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Las Vegas, NV, USA, 2016, pp. 779–788.
[36] X. Zhou, D. Wang, and P. Krähenbühl, "Deep learning for automatic fish species classification," Fish. Res., vol. 204, pp. 1–10, 2017.
[37] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767 Accessed on: Jun.8, 2023
[38] K. Kim, K. Kim, and S. Jeong, "Application of YOLO v5 and v8 for recognition of safety risk factors at construction sites," Sustainability, vol. 15, no. 20, p. 15179, 2023.
[39] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934 Accessed on: Jan.20, 2023
[40] Ultralytics, "YOLOv5," GitHub Repository, 2020. [Online]. Available: https://github.com/ultralytics/yolov5 Accessed on: Feb.5, 2023
[41] Thu-Mig, "YOLOv10," GitHub Repository, 2024. [Online]. Available: https://github.com/THU-MIG/yolov10 Accessed on: May.10, 2024
[42] E. M. Ditria, S. Lopez-Marcano, M. Sievers, E. L. Jinks, and R. M. Connolly, "Deep learning for underwater fish detection: Accuracy and efficiency of human and automated methods," Ecol. Inform., vol. 55, p. 101019, 2020.
[43] N. Wang, Y. Zhang, and J. Han, "Fish detection and classification in complex underwater environments using deep learning," Inf. Sci., vol. 578, pp. 298–314, 2022.
[44] X. Li, Y. You, J. Wu, and H. Zhou, "Fish species recognition in underwater video based on deep convolutional neural networks," J. Comput. Sci., vol. 10, pp. 124–133, 2015.
[45] S. Villon, D. Mouillot, M. Chaumont, and T. Claverie, "Automatic underwater fish species classification with limited data using deep learning," Ecol. Inform., vol. 48, pp. 124–131, 2018.
[46] D. Hridayami, Y. Tjahyadi, and W. Widhiarso, "Deep learning-based fish classification system for real-time underwater fish monitoring," Procedia Comput. Sci., vol. 157, pp. 447–453, 2019.

[47] V. Allken, N. O. Handegard, S. Rosen, T. Schreyeck, T. Mahiout, and K. Malde, "Fish species identification using a convolutional neural network trained on synthetic data," ICES J. Mar. Sci., vol. 76, no. 1, pp. 342–349, 2019.
[48] A. Jalal, M. Nasir, and K. Kim, "Fish detection and species classification in underwater environments using deep learning with temporal information," Sensors, vol. 20, no. 12, p. 3452, 2020.
[49] A. Salman, A. Jalal, and K. Kim, "Region-based convolutional neural network for fish detection in underwater environments," Sensors, vol. 20, no. 5, p. 1400, 2020.
[50] M. Hussain, A. Jalal, and K. Kim, "Modified AlexNet for fish classification in underwater environments," Comput. Electr. Eng., vol. 93, p. 107231, 2021.
[51] H. Wang, S. Zhu, M. Wang, and J. Ye, "Advances in deep learning: Applications in the healthcare sector," J. Healthcare Eng., vol. 2022, no. 1, pp. 1–14, 2022.
[52] B. Patro, S. Mukhopadhyay, and P. P. Roy, "Real-time object detection using YOLOv5 for fish monitoring in adverse environments," J. Ocean Technol., vol. 18, no. 2, pp. 61–73, 2023.
[53] Y. Eği, "YOLO v7 and computer vision-based mask-wearing warning system for congested public areas," J. Inst. Sci. Technol., vol. 13, no. 1, pp. 22–32, 2023.
[54] Z. Yücel and D. Çetintaş, "Yolov9 ile kan hücrelerinin otomatik tanımlanması: Optimizasyon ve öğrenme oranı etkileri," Adıyaman Üniv. Müh. Bilim. Derg., vol. 11, no. 22, pp. 125–135, 2024.
[55] D.-J. Shin and J.-J. Kim, "A deep learning framework performance evaluation to use YOLO in Nvidia Jetson platform," Appl. Sci., vol. 12, no. 8, p. 3734, 2022.
[56] C. Li, Y. Zhang, X. Wu, and Y. Yang, "Deep learning-based fish behavior analysis in underwater videos," Inf. Sci., vol. 579, pp. 92–105, 2022.
[57] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, pp. 318–327, 2020.
[58] J. Terven and D. Cordova-Esparza, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," arXiv preprint, 2023. [Online]. Available: https://arxiv.org/abs/2304.00501 Accessed on: Mar.15, 2024
[59] G. Jocher and M. R. Munawar, "Ultralytics YOLOv8," GitHub Repository, 2023. [Online]. Available: https://github.com/ultralytics/yolov8 Accessed on: Mar.15, 2024
[60] V. Viswanatha, R. K. Chandana, and A. C. Ramachandra, "Real time object detection system with YOLO and CNN models: A review," arXiv preprint, 2022. [Online]. Available: https://arxiv.org/abs/2208.00773 Accessed on: Mar.12, 2023
[61] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of Yolo algorithm developments," Procedia Comput. Sci., vol. 199, pp. 1066–1073, 2022.
[62] M. Hussain, "YOLOv1 to v8: Unveiling each variant–A comprehensive review of YOLO," IEEE Access, vol. 12, pp. 42816–42833, 2024.
[63] A. Vijayakumar and S. Vairavasundaram, "YOLO-based object detection models: A review and its applications," Multimedia Tools Appl., vol. 83, no. 35, pp. 83535–83574, 2024, DOI: 10.1007/s11042-024-18872-y.
[64] C. F. G. D. Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," ACM Comput. Surv., vol. 54, no. 10, pp. 1–25, 2022.
[65] S. Aburass, "Quantifying overfitting: Introducing the overfitting index," arXiv preprint, 2023. [Online]. Available: https://arxiv.org/abs/2308.08682 Accessed on: Oct.15, 2023
[66] L. Prechelt, "Early stopping — But when?," in Neural Networks: Tricks of the Trade. Berlin, Germany: Springer, 2012.
[67] J. Brownlee, "A gentle introduction to early stopping to avoid overtraining neural networks," Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/, Accessed on: Apr. 1, 2024.
[68] S. Paguada, L. Batina, I. Buhan, and I. Armendariz, "Being patient and persistent: Optimizing an early stopping strategy for deep learning in profiled attacks," IEEE Trans. Comput., vol. 74, no. 3, pp. 1–12, 2024.
[69] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "CMS-RCNN: Contextual multi-scale region-based CNN for unconstrained face detection," in Deep Learning for Biometrics. Cham, Switzerland: Springer, 2017.
[70] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, pp. 318–327, 2020.
[71] K. Su, L. Cao, B. Zhao, N. Li, D. Wu, and X. Han, "N-IoU: Better IoU-based bounding box regression loss for object detection," Neural Comput. Appl., vol. 36, no. 6, pp. 3049–3063, 2024.

[72] P. Liu, G. Zhang, B. Wang, H. Xu, X. Liang, Y. Jiang, and Z. Li, "Loss function discovery for object detection via convergence-simulation driven search," arXiv preprint, 2021. [Online]. Available: https://arxiv.org/abs/2102.04700 Accessed on: Apr.8, 2023

[73] P. Jayanthi, "Machine learning and deep learning algorithms in disease prediction: Future trends for the healthcare system," in Deep Learning for Medical Applications with Unique Data. Cambridge, MA, USA: Academic Press, 2022.

[74] P. D. K. Reddy, M. Margala, S. S. Shankar, and P. Chakrabarti, "Early fire danger monitoring system in smart cities using optimization-based deep learning techniques with artificial intelligence," J. Reliable Intell. Environ., vol. 10, no. 2, pp. 197–210, 2024.

[75] Z. Ding, L. Sun, X. Mao, L. Dai, and B. Xu, "Location-independent adversarial patch generation for object detection," J. Electron. Imaging, vol. 32, no. 4, p. 043035, 2023.

[76] V. Kshirsagar, R. H. Bhalerao, and M. Chaturvedi, "Modified YOLO module for efficient object tracking in a video," IEEE Latin Amer. Trans., vol. 21, no. 3, pp. 389–398, 2023.