IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.x, No.x, July xxxx, pp. 1~5
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: 10.22146/ijccs.xxxx

Optimizing Coral Fish Detection: Faster R-CNN, SSD MobileNet, YOLOv5 Comparison

Syifa Afnani Santoso*[1], Indra Jaya[2], Karlisa Priandana[3]
[1,2]Department of Marine Science and Technology, FPIK, IPB University, Bogor, Indonesia
[3]Department of Computer Science, FMIPA, IPB University, Bogor, Indonesia
e-mail: ***[1]afnani_syifa@apps.ipb.ac.id**, [2]indrajaya@apps.ipb.ac.id, [3]karlisa@apps.ipb.ac.id

Abstrak

Studi ini menyoroti peran kritis pengamatan keberlimpahan ikan Chaetodontidae yang akurat, terutama dalam menilai kesehatan terumbu karang. Melalui integrasi algoritma deep learning (Faster R-CNN, SSD-MobileNet, dan YOLOv5) ke dalam Autonomous Underwater Vehicle (AUV), penelitian ini bertujuan mempercepat identifikasi ikan di lingkungan akuatik. Evaluasi algoritma menunjukkan YOLOv5 memiliki akurasi tertinggi, disusul oleh Faster R-CNN dan SSD-MobileNet. Walaupun demikian, SSD-MobileNet menonjol dengan kecepatan komputasi superior yang memiliki mean average precision (mAP) sekitar 92,21% dan framerate sekitar 1,24 fps. Pemanfaatan Coral USB Accelerator juga meningkatkan kecepatan komputasi pada Raspberry Pi 4, memungkinkan pendeteksian secara real-time. Studi ini memasukkan pelacakan centroid untuk memfasilitasi perhitungan yang akurat dengan memberikan ID unik kepada objek teridentifikasi per kelas. Implementasi sistem secara real-time berhasil mencapai akurasi 87,18% dan presisi 87,54% pada 30 fps, memungkinkan AUV melakukan deteksi dan pelacakan ikan secara real-time. Temuan penelitian menunjukkan bahwa teknologi ini memiliki potensi untuk diterapkan di lingkungan bawah air, memungkinkan AUV mengumpulkan data ikan secara akurat dan efisien serta berkontribusi pada berbagai penelitian bawah air dan upaya konservasi.

***Kata kunci**—Deteksi objek, Faster R-CNN, SSD MobileNet, YOLOv5, pelacakan centroid*

Abstract

This study underscores the critical role of accurate Chaetodontidae fish abundance observations, particularly in assessing coral reef health. By integrating deep learning algorithms (Faster R-CNN, SSD-MobileNet, and YOLOv5) into Autonomous Underwater Vehicles (AUVs), the research aims to expedite fish identification in aquatic environments. Evaluating the algorithms, YOLOv5 emerges with the highest accuracy, followed by Faster R-CNN and SSD-MobileNet. Despite this, SSD-MobileNet showcases superior computational speed with a mean average precision (mAP) of around 92.21% and a framerate of about 1.24 fps. Furthermore, employing the Coral USB Accelerator enhances computational speed on the Raspberry Pi 4, enabling real-time detection capabilities. This study incorporates centroid tracking, facilitating accurate counting by assigning unique IDs to identified objects per class. Ultimately, the real-time implementation of the system achieves 87.18% accuracy and 87.54% precision at 30 fps, empowering AUVs to conduct real-time fish detection and tracking. The research findings show that this technology has potential for application in underwater environments, allowing AUVs to collect fish data accurately and efficiently and contribute to a variety of underwater research and conservation efforts.

***Keywords—**Object detection, Faster R-CNN, SSD MobileNet, YOLOv5, centroid tracking*

Received June 1st,2012; Revised June 25th, 2012; Accepted July 10th, 2012

1. INTRODUCTION

Coral reef is a complex ecosystem that provides numerous benefits in coastal areas [1]. From the ecological aspect as a habitat and food provider for many marine organisms to the economical aspect as a place for recreation and tourism [2]. Unfortunately, some economic activities such as fisheries are not environmentally friendly, which destroys coastal ecosystems [3]. The damage can be identified from several indicators, including physical, chemical, and biological indicators. One of the parameters of the biological indicators is the presence of fish species from the Chaetodontidae family [4]. Chaetodontidae is one of the most important fish groups in the coral reef ecosystem because their diversity and abundance are highly associated with the diversity and condition of coral reefs [5]. A coral reef ecosystem with high reliance and good health conditions usually has a high species richness of Chaetodontidae [6]. Therefore, it is necessary to research the abundance of reef fish concerning monitoring the health of coral reef ecosystems in the waters.

The method commonly used for collecting reef fish data is the visual observation method (underwater visual census, UVC), which records all fish species found while diving on a 50 meters line transect and stretches 2.5 meters to the left and 2.5 meters to the right of the line transect so that the observation area is 250 m2 [7]. This method has several disadvantages, such as requiring expert divers who can identify and count the number of fish species among hundreds of individual fish, taking a long time, and limitations by the depth, duration of the dive, and the ability of divers to distinguish similar species [8].

To overcome these limitations, various technologies have been developed, including high-resolution video recording for later identification [9]. Recently, Deep Learning methods, particularly Convolutional Neural Networks (CNNs), have been employed for fish identification from underwater photos and videos [8][10]. While CNNs have demonstrated high accuracy (95%), they still require trained specialists and entail significant time and cost [8].

In contrast, recent studies have explored the use of advanced Deep Learning models, such as Fast R-CNN, Faster R-CNN, and YOLOv3, for efficient fish detection from underwater videos [11]. These studies have highlighted trade-offs between accuracy and speed, with Faster R-CNN exhibiting superior accuracy and YOLOv3 demonstrating faster detection. The choice of model depends on the specific requirements of the application. For instance, YOLOv3 was successfully implemented in a Remotely Operated Vehicle (ROV) for effective underwater object detection [11].

Moreover, Autonomous Underwater Vehicles (AUVs) equipped with cameras and sensors offer promising opportunities for underwater surveys [12]. By integrating optimal Deep Learning models and devices, AUV can conduct real-time observations automatically, thereby enhancing efficiency and accuracy in fish identification and quantification. Therefore, this research aims to identify the essential model and other recommendations for implementing fish identification and quantification using AUV.

2. METHODS

2.1 Dataset Collection and Labelling

The dataset used as training material for this study is a collection of images of 3 species of fish in the Chaetodontidae family, namely Chaetodon trifasciatus, Chaetodon auriga, and Chaetodon semilarvatus. The video used as material was recorded using a webcam camera mounted on an AUV from various directions to obtain varied data, then converted into a set of images to be used as a dataset.

The next step is image labeling, which is done using labelimg software by creating a box (ground truth box) that marks an object to be identified and labeling it according to the

name of the object's species. This process is important so that the objects contained in the image can be recognized during the training process. The result of image labeling is a file with *.xml and *.txt formats which contains label information and the position of the object to be identified.

To facilitate model training and evaluation, the set of images and labels was divided into 10 equal subsets. One of these subsets was designated as the test set, while the remaining nine subsets were used as the training set. This division was repeated ten times, each time with a different combination of test and training sets, resulting in a total of 10 datasets. This method is called 10-fold cross-validation [13].

2.2 Algorithms

In this study, 3 types of detection algorithms will be compared, namely Faster Region-based Convolutional Neural Networks (Faster-RCNN), Single Shot Detector with MobileNet (SSD-MobileNet), and You Only Look Once version 5 (YOLOv5). Faster R-CNN is a multi-stage detector algorithm that separates the process for predicting object presence scores and object classes into two different stages, while SSD and YOLO are single-stage detector algorithms where images are passed once through the network to predict object presence scores. and object class at the same time. Faster R-CNN architecture has two main components, namely RPN network to identify ROI and estimate its location and Fast-RCNN network to classify objects and refine their location. SSD algorithm uses a sophisticated CNN architecture as the basic network for feature extraction with an additional convolution layer to produce a smaller feature map to detect objects with different scales [14]. In comparison, YOLOv5 architecture consists of three parts, namely the back (CSP Darknet), neck (PANet), and head (YOLO). On the head, YOLO layer generates 3 different sizes (18x18, 36x36, and 72x72) enabling the model to perform multi-scale predictions [15].

Object quantification is performed using the Centroid Tracking algorithm. This algorithm utilizes bounding box (x,y) coordinates obtained from detecting objects in the frame and then calculates the coordinates of the center point (centroid) and assigns a unique ID to each of these boxes. We assume that the same object will move to the centroid that has the minimum distance compared to other centroids in the next frame so that a pair of centroids that have the minimum distance in the next frame is considered the same object and is given the same ID. If the number of objects detected is more than the number of objects in the previous frame, then the centroid with the farthest distance compared to other centroids will be registered as a new unique ID [16].

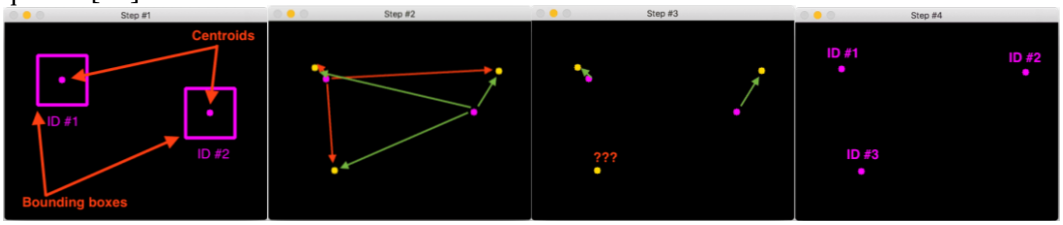

Figure 1 Illustration of centroid tracking method

2.3 Dataset Training

The training process is carried out to recognize objects by training the algorithms. There are two stages in the training process, namely forward propagation and backward propagation. In the forward propagation stage, the deep learning model will pass the input image through a set of layers in the CNN network consisting of the convolution layer, pooling layer, and fully connected layer (FC), then apply an activation function to determine the object class with probability values between 0 to 1 [17].

The convolution layer will extract features by performing dot product operations between the input image matrix and the filter matrix. Each filter performs a different operation, such as edge detection, sharpening, or smoothing. The output of this convolution process is

called a feature map. The pooling layer reduces the dimensions of each feature map while retaining important information. This pooling process is done by taking the largest value in some data in the feature map. The feature map generated after passing through a set of convolution layers and pooling layers is then converted into a one-dimensional vector which will be passed into the FC. In the FC layer, each value in the vector is multiplied by the weight matrix, and the result is entered into the activation function to predict the object. The prediction results are then compared with the actual results to calculate the loss value.

The model will try to reduce the loss value through the backward propagation stage. This process is done by updating the value of the weight and filter based on the loss value. Then the forward propagation stage will run again using the updated filter and weight. These two stages of training take place continuously to obtain the output with the smallest loss value. Finally, the training model is saved in the form of a prototype file (*.pb).

As previously mentioned, the dataset was divided into ten equal subsets to facilitate model training and evaluation. Each iteration of training involved using nine of these subsets for training the model, while the remaining subset served as the validation set. This process was repeated ten times, with each subset being used as the validation set exactly once.

By employing 10-fold cross-validation during training, the models were trained and evaluated on multiple partitions of the dataset, enhancing the robustness of the training process and ensuring that the models generalize well to unseen data. This approach helped mitigate the risk of overfitting and provided a more reliable estimate of the models' performance, contributing to the credibility of the study's findings [13].

2.4 Validation Test

The validation test process is carried out to check the accuracy and precision of the model. All images included in the test set will run through forward propagation with the weight obtained from the training process. The output of this process is an image with the prediction results in the form of a box and a description of the label and confidence score. In binary classification, data is separated into positive (object) and negative (non-object). Then the binary classifier will classify all the sample data as positive or negative. This classification produces four outputs: two types of true classification, True Positive (TP) and True Negative (TN); and two types of incorrect classification, False Positive (FP) and False Negative (FN). The 2x2 table contains these four outputs called the confusion matrix which can be used to calculate the accuracy or precision of the training model [18].

According to reference [18], model accuracy/precision is obtained from the percentage of labels that are predicted to be positive compared to the overall positive predicted results using the following calculation formula:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (1)$$

$$Precision = TP/(TP + FP) \quad (2)$$

The accuracy values are calculated for each label by running the performance metrics created by [19]. The average percentage of the precision value of each label is expressed by the mean average precision (mAP).

Each trained model underwent evaluation using the 10-fold cross-validation method to ensure reliable performance metrics. The primary purpose of employing this method is to evaluate the generalization performance of the trained models across different subsets of the dataset. By rotating through all 10 datasets and using each one as the test set while training on the remaining nine, the models are tested on diverse data samples.

Additionally, the use of 10-fold cross-validation facilitates the selection of the most suitable model for implementation. By comparing the performance of the models across multiple iterations and datasets, informed decisions can be made regarding which model best balances accuracy, precision, and computational efficiency [13].

2.5 Computation Speed Test

This process is done by running a Python program that will turn on the webcam camera on the laptop and identify the recorded object in real-time. This process is carried out to determine the computational speed of the trained model. The computational speed of a model is indicated by frames per second (fps), which is the number of images or frames that the system can process in 1 second. A greater framerate suggests that the detector can handle a larger number of images within a brief timeframe, making it more efficient. In real-time scenarios, a high frame rate is crucial to ensure the detector can match the pace of the incoming video stream [20].

2.6 Comparison of Model Accuracy and Speed

One of the objectives of this research is to determine the most optimal model to be implemented in AUV. The model must have high accuracy (more than 90%) and good computational speed. Therefore, we use the objective function which we will find the maximum value as follows:

$$f(a, fr) = a \times fr; \quad a > 90\% \tag{3}$$

where:

a = model accuracy (mAP)

fr = framerate (fps)

The model with the highest value will be selected to be implemented on the Raspberry Pi 4.

2.7 Model Implementation

The model that has been selected will then be converted to the Tensorflow Lite or TFLite (*.tflite) format using the TensorFlow Lite Optimizing Converter (TOCO). Tensorflow Lite is a framework optimized for implementing lightweight deep learning models on resource-constrained devices [21]. The converted model is then implemented on Raspberry Pi using Python programming language. Before carrying out the implementation in real-time, identification and quantification of underwater fish videos are carried out to ensure that the program runs well.

Next is to carry out two treatments for identification in real-time using a webcam connected to the Raspberry Pi, namely implementation with and without the addition of the Coral USB Accelerator device. Coral USB Accelerator adds an Edge TPU coprocessor to the system so it can run high-performance, low-power models. What the Raspberry Pi needs to be able to use the Coral USB Accelerator is to install the libedgetpu library.

The framerate values of these two treatments will be compared to determine whether the addition of the Coral USB Accelerator is needed to run the implementation process more quickly and in real-time. Reference [22] states that the use of Coral USB Accelerator can increase the framerate value of TFLite model running on the Raspberry Pi up to 75.53 fps, much different from without using the device which is only 3.31 fps. This framerate value allows the process of identifying and counting fish using AUV to run in real-time.

3. RESULTS AND DISCUSSION

3.1 Trained Model Accuracy

In the training process, the three algorithms (Faster R-CNN, SSD-MobileNet, and YOLOv5) are trained using input from the 10-fold cross-validation datasets, resulting in 30 training models. The accuracy value is obtained by calculating the mean average precision (mAP) value for each of these models and is presented in Table 1 below.

Table 1 Accuracy comparison for Faster-RCNN, SSD-MobileNet, and YOLOv5 algorithms with 10-fold cross-validation

Model	Faster R-CNN	SSD-MobileNet	YOLOv5
Dataset 1	97.54	84.34	98.5
Dataset 2	97.03	90.59	99.3
Dataset 3	97.33	92.08	**99.6**
Dataset 4	97.35	86.8	99
Dataset 5	97.01	**92.21**	99
Dataset 6	98.37	87.1	98.9
Dataset 7	**99.31**	88.73	99.4
Dataset 8	97.07	86.92	99.5
Dataset 9	98.58	83.75	99
Dataset 10	98.05	79.51	98.8
Average	**97.764**	**87.203**	**99.1**

Based on Table 1, the sequence of algorithms with the highest to lowest average accuracy is YOLOv5 with an average mAP value of 99.1%, Faster R-CNN with an average mAP value of 97.764%, and SSD-MobileNet with an average mAP value of 87.203%. Out of 10 trained models for each algorithm, one model that has the best accuracy and meets the predetermined objective function condition (accuracy value above 90%) would be selected. For Faster R-CNN, dataset model 7 was selected because it has the highest accuracy value compared to the other nine models with a mAP value of 99.31%. For SSD-MobileNet, dataset model 5 was selected with a mAP value of 92.21%, and for YOLOv5, dataset model 3 was selected with a mAP value of 99.6%.

3.2 Model Computation Speed

The three selected models are run using the Python program on Anaconda Prompt. This program intends to identify objects in real-time by utilizing a webcam attached to the laptop. While the program is running, there is a window that displays the results such as bounding boxes and labels that mark the identified object, and the framerate that indicates the computational speed of the model being run. A comparison of the framerate values of the three models can be seen in Table 2.

Table 2 Overall comparison of Faster-RCNN, SSD-MobileNet, and YOLOv5 algorithms

Model	Framerate, fr (fps)	Accuracy, a (maP)	f(a,fr) = a + fr
Faster R-CNN	0.10	0.9931	1.0931
SSD-MobileNet	1.24	0.9221	**2.1621**
YOLOv5	0.65	0.996	1.646

The order of the models with the highest to lowest computing speed is SSD-MobileNet with a framerate of around 1.24 fps, followed by YOLOv5 with 0.65 fps, and lastly Faster R-CNN with 0.10 fps. This framerate is then used as an input value for the objective function that has been determined together with accuracy from the previous analysis. The results of this analysis show that the SSD-MobileNet model has the best accuracy and speed, so it was chosen as the recommended model to be implemented on Raspberry Pi 4. By selecting the SSD-MobileNet model for implementation, this research ensures a balance between accuracy and computational speed, optimizing the performance of real-time object identification in underwater environments using AUV systems.

3.3 Increasing Computational Speed

The bar graph in Figure 2 compares computation speed (framerate) between running the SSD-MobileNet model on the Raspberry Pi 4 with and without the addition of a Coral USB

Accelerator device. The graph highlights the significant increase in computing speed achieved by incorporating the Coral USB Accelerator.

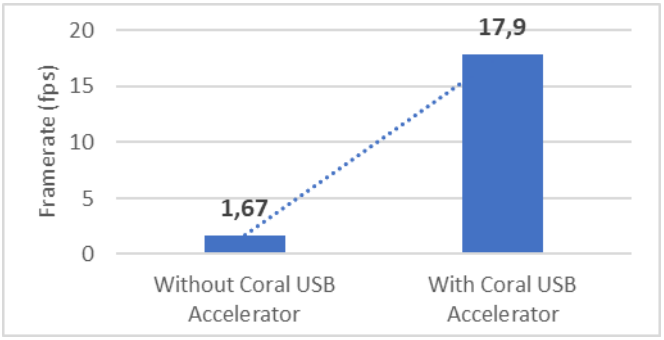

Figure 2 Comparison between computational speed with and without Coral USB Accelerator

Without Coral USB Accelerator, SSD-MobileNet model achieved a framerate of around 1.67 frames per second (fps). However, with Coral USB Accelerator device, the framerate increased significantly to around 17.9 fps. The incredible increase in computing speed, about 10-fold, shows the efficacy of Coral USB Accelerator in improving the performance of TFLite models on Raspberry Pi 4.

This analysis concludes that integrating the Coral USB Accelerator with the Raspberry Pi 4 and SSD-MobileNet TFLite models allows the system to meet real-time processing needs. This combination ensures that fish identification and quantification can be carried out accurately and quickly, thereby facilitating efficient data collection and analysis in underwater environments.

3.4 System Implementation on Video

The selected SSD-MobileNet model is then implemented into an object detection and tracking algorithm to test the fish quantification system. This algorithm is compiled using Python programming language. The object tracking process uses the centroid tracking algorithm which uses inputs such as bounding boxes and other object details from the results of fish detection using the selected model. It then assigns a unique ID to each detected fish. These IDs for C. trifasciatus, C. auriga, and C. semilarvatus fish are TR_ID, AU_ID, and SE_ID, respectively; followed by the sequence number of the fish detected per class.

This system is then run on videos sourced from the internet. The outputs of this process are the detection result video and a CSV file that summarizes the detection results. In the video output, a timestamp is displayed showing the seconds of the video that is running on the top left side, and the detection results such as bounding boxes, labels, and confidence scores, as well as the ID given to each detected object. An example of the detection result video display can be seen in Figure 3.

Figure 3 The system detected two different C. trifasciatus marked with TR_ID1 and TR_ID2

Besides being saved in video form, the detection results are also summarized in the form of a CSV file. There are two columns in the file, namely "Object Name" or the class name of the detected object, and "Total" or the number of objects from that class detected while the video is running. An example of a CSV file that is saved after running the system on the same video as Figure 3 can be seen in Figure 4.

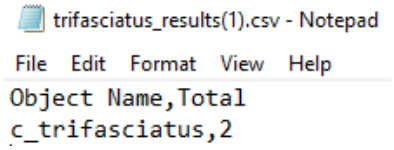

Figure 4 The CSV file after running the identification and quantification system on a video

The combination of detection video and CSV files provides comprehensive output for analyzing and processing fish quantification results. The video allows visual inspection of detected fish and their trajectories, while the CSV file offers a quantitative summary of detected fish per class. This output can also facilitate further analysis of fish behavior, population dynamics, and other relevant research investigations in underwater environments.

3.5 System Implementation in Real-Time

The system that has been tested on video for fish identification and quantification is now implemented on a Raspberry Pi 4 device in real-time. This implementation uses a webcam and Coral USB Accelerator to improve performance. System testing is carried out by displaying several images alternately in front of the camera. There were a total of 34 images used in this test, consisting of 12 images of C. trifasciatus, 12 images of C. auriga, and 10 images of C. semilarvatus. This process allows evaluation of the system's ability to recognize differences between species and measures its level of accuracy and effectiveness.

Similar to the previous discussion, each detected object is annotated with a bounding box, label, and unique ID for each class. Additionally, the top left corner of the screen shows the framerate, date, and time during recording. The recording results are then saved in avi video format. An illustration of the results of real-time fish identification and quantification is presented in Figure 5.

Figure 5 Fish detection results in real-time on Raspberry Pi 4 using a webcam and Coral USB Accelerator

In addition to the video output, the detection results are also summarized and saved in a CSV file. This file follows the same format as the output obtained from the system implementation on video. Figure 6 illustrates an example of a CSV file saved after running the system in real-time.

```
Object Name,Total
c_trifasciatus,12
c_auriga,14
c_semilarvatus,13
```

Figure 6 The CSV file after running the identification and quantification system in real-time

In Figure 5, the computing speed in testing the identification and quantification system reaches around 30 frames per second (fps) on a video with a resolution of 640x480 pixels. This speed is considered very good for identifying underwater fish in real-time.

Apart from that, Figure 6 shows that the system succeeded in correctly detecting all the objects displayed in front of the camera, but there were still errors in determining the number.

Errors in object tracking can occur for two main reasons. First, the existence of crossings with similar objects makes it difficult for the algorithm to distinguish whether it is the same object or a new object that appears after the crossing. Second, setting the detection threshold too high (90%) can cause moving objects to be considered 'missing' when detected with a low confidence score. So, when an object 'appears' again with a confidence score above 90%, the algorithm can incorrectly assume it is a new object.

These results can be measured using accuracy and precision metrics calculated from the number of correctly detected objects compared to the overall number of expected objects. With this accuracy and precision, it can be indicated that the system provides good results in recognizing and classifying fish based on their species.

$$Accuracy = \frac{34}{39} \times 100\% = \mathbf{87,18\%}$$

$$TR\ Precision = \frac{12}{12+0} \times 100\% = 100\%$$

$$AU\ Precision = \frac{12}{12+2} \times 100\% = 85{,}71\%$$

$$SE\ Precision = \frac{10}{10+3} \times 100\% = 76{,}92\%$$

$$Avg\ Precision = \frac{100 + 85{,}71 + 76{,}92}{3}\% = \mathbf{87,54\%}$$

The resulting video output with bounding boxes, labels, and unique IDs provides visual evidence of the fish species detected, facilitating further analysis and documentation. Additionally, the CSV file summary offers a quick overview of detected objects and their number per class, allowing easy extraction and analysis of data for statistical purposes.

4. CONCLUSIONS

This study conducted a comprehensive evaluation of three deep learning algorithms (Faster R-CNN, SSD-MobileNet, and YOLOv5) using the 10-fold cross-validation method. YOLOv5 showed the highest accuracy at 99.6%, followed by Faster R-CNN with 99.31%, and SSD-MobileNet with 92.21%. In terms of computing speed, SSD-MobileNet outperforms other models, achieving a framerate of 1.24 fps.

For implementation, the SSD-MobileNet detection model and centroid tracking are implemented on a Raspberry Pi 4, coupled with a Coral USB Accelerator device. The test results show that the object identification and quantification system can run at a speed of 30 fps with an accuracy value of 87.18% and a precision of 87.54%. The recording results are saved in video and CSV format, providing visual evidence and a summary of the number of fish detected per class.

The research findings show that this technology has potential for application in underwater environments, allowing AUVs to collect fish data accurately and efficiently and contribute to a variety of underwater research and conservation efforts.

REFERENCES

[1] A. J. Woodhead, C. C. Hicks, A. V. Norstrom, G. J. Williams, and N. A. J. Graham, “Coral reef ecosystem services in the Anthropocene”, Coral Reef Functional Ecology in the Anthropocene, vol. 33, no. 6, pp. 1023-1034, Mar. 2019 [Online]. Available: https://doi.org/10.1111/1365-2435.13331

[2] A. Tuwo, and J. Tresnati, "Coral reef ecosystem" in Advances in Biological Sciences and Biotechnology, vol. 1, Y. Singh, India: Integrated Publications, 2020, pp. 75-104. Available: https://www.researchgate.net/publication/349485597_Coral_Reef_Ecosystem

[3] L. Fudjaja et al., "Anthropogenic activity and the destruction of coral reefs in the waters of small islands", in IOP Conf. Series: Earth and Environmental Science, 2020, pp. 1-8. Available: https://iopscience.iop.org/article/10.1088/1755-1315/575/1/012057

[4] R. Iskandar, K. Z. Soemarno, and D. G. R. Wiadnya, "Coral reef condition with Chaetodontidae fish as the indicators in the waters of the Samber Gelap island of Kotabaru, South Kalimantan", RJOAS, vol. 11, no. 107, Nov. 2020 [Online]. Available: https://doi.org/10.18551/rjoas.2020-11.10

[5] E. S. Reese, "Predation on Corals by Fishes of the Family Chaetodontidae: Implications for Conservation and Management of Coral Reef Ecosystems", Bulletin of Marine Science, vol. 31, no. 3, pp. 594-604, Jul. 1981 [Online]. Available: https://www.ingentaconnect.com/content/umrsmas/bullmar/1981/00000031/00000003/art00011

[6] K. Wibowo, M. Adrim, and P. C. Makatipu, "Community structure of Chaetodontidae in the west of Banda Sea", Mar. Res. Indonesia, vol. 38, no. 1, pp. 1-8, Feb. 2013 [Online]. Available: http://dx.doi.org/10.14203/mri.v38i1.51

[7] S. English, C. Wilkinson, and V. Baker, "Survey manual for tropical marine resources," Townsville: Australian Institute of Marine Sciences, 1997. Available: https://www.aims.gov.au/sites/default/files/Survey%20Manual-sm01.pdf

[8] S. Villon et al., "A deep learning method for accurate and fast identification of coral reef fishes in underwater images," Ecological Informatics, vol. 48, pp. 238-244, Nov. 2018 [Online]. Available: https://doi.org/10.1016/j.ecoinf.2018.09.007

[9] D. Pelletier, K. Leleu, G. Mou-tham, N. Guillemot, and P. Chabanet, "Comparison of visual census and high definition video transects for monitoring coral reef fish assemblages," Fisheries Research, vol. 107, no. 1-3, pp. 84-93, Jan. 2011 [Online]. Available: https://doi.org/10.1016/j.fishres.2010.10.011

[10] W. Xu and S. Watzner, "Underwater fish detection using deep learning for water power applications," in 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Nov. 2018, pp. 313-318 [Online]. Available: https://doi.org/10.1109/CSCI46756.2018.00067

[11] F. Han, J. Yao, H. Zhu, and C. Wang, "Underwater image processing and object detection based on deep CNN method," Journal of Sensors, vol. 2020, pp. 1-20, May 2020 [Online]. Available: https://doi.org/10.1155/2020/6707328

[12] A. M. Munoz, M.-J. Moron-Fernandez, D. Cascado-Caballero, F. Diaz-del-Rio, and P. Real, "Autonomous Underwater Vehicles: Identifying Critical Issues and Future Perspectives in Image Acquisition," Sensors, vol. 23, no. 10, pp. 1-27, May 2023 [Online]. Available: https://doi.org/10.3390/s23104986

[13] L. A. Yates, Z. Aandahl, S. A. Richards, B. W. Brook, "Cross validation for model selection: A review with examples from ecology," Ecological Monographs, vol. 93, no. 1, pp. 1-24, Sep. 2022 [Online]. Available: https://doi.org/10.1002/ecm.1557

[14] U. Alganci, M. Soydas, and E. Sertel, "Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images," Remote Sensing, vol. 12, no. 3, pp. 1-28, Feb. 2020 [Online]. Available: https://doi.org/10.3390/rs12030458

[15] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," Forests, vol. 12, no. 2, pp. 1-17, Feb. 2021 [Online]. Available: https://doi.org/10.3390/f12020217

[16] A. Bakliwal et al., "Crowd counter: an application of centroid tracking algorithm," International Research Journal of Modernization to Engineering Technology and Science, vol. 2, no. 4, pp. 1138-1141, April 2020 [Online]. Available:

https://www.irjmets.com/uploadedfiles/paper/volume2/issue_4_april_2020/868/1628083008.pdf

[17] Purwono, A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. Haq, "Understanding Convolutional Neural Network (CNN): A Review," International Journal of Robotics and Control Systems, vol. 2, no. 4, pp. 739-748, Dec. 2022 [Online]. Availabe: https://doi.org/10.31763/ijrcs.v2i4.888

[18] M. M. Fahmy, "Confusion Matrix in Binary Classification Problems: A Step-by-Step Tutorial," Journal of Engineering Research, vol. 6, no. 5, pp. T1-T12, Dec. 2022 [Online]. Available: https://dx.doi.org/10.21608/erjeng.2022.274526

[19] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Jul. 2020, pp. 237-242. Available: https://doi.org/10.1109/IWSSIP48289.2020.9145130

[20] S. Xu, H. Tang, J. Li, L. Wang, X. Zhang, H. Gao, "A YOLOW Algorithm of Water-Crossing Object Detection," Applied Sciences, vol. 13, no. 15, pp. 1-15, Jul. 2023 [Online]. Available: https://doi.org/10.3390/app13158890

[21] D. Choe, E. Choi, and D. K. Kim, "The real-time mobile application for classifying of endangered parrot species using the cnn models based on transfer learning," Deep Learning in Mobile Information System, vol. 2020, pp. 1-13, Mar. 2020 [Online]. Available: https://doi.org/10.1155/2020/1475164

[22] S. Sterckval, "Google Coral Edge TPU vs NVIDIA Jetson Nano: a quick deep dive into EdgeAI performance," 2019 [Online]. Available: https://medium.com/@samsterckval/google-coral-edge-tpu-vs-nvidia-jetson-nano-a-quick-deep-dive-into-edgeai-performance-bc7860b8d87a. [Accessed: 18-Jun-2023]