Coral Detection based on Optimised Lightweight YOLO Model

R E Saragih[*1], H S Husin[2], M K N Mursalim[3], Yodi[4]

[1,3]Department of Informatics Engineering, Faculty of Computer, Universitas Universal, Batam, Indonesia

[2]School of Computer Science, Faculty of Innovation and Technology, Taylor's University, Kuala Lumpur, Malaysia

[4]Department of Information Systems, Faculty of Computer, Universitas Universal, Batam, Indonesia

E-mail: raymondes@uvers.ac.id[1], husna.husin@taylors.edu.my [2], muh.khaerul.naim@uvers.ac.id [3], yodi@uvers.ac.id [4]

Abstract. Coral reefs are essential marine ecosystems that face significant threats due to climate change, pollution, and overfishing. Effective monitoring is crucial for conservation efforts, but traditional methods are labor-intensive and inefficient. This study proposes a deep learning-based coral detection model built based on the YOLOv8 architecture, specifically for nano and small. In addition, the Ghost modules and Ghost bottlenecks were utilized to modify the original YOLOv8 small. The proposed model was trained on an underwater coral dataset and evaluated in terms of precision, recall, and mean Average Precision (mAP) metrics. Experimental results demonstrate that the YOLOv8 small model and YOLOv8 small model with Ghost modules achieved a mAP of 53.675% and 55.88%, respectively, while maintaining a compact model size. This work contributes to developing efficient and lightweight coral detection systems to support conservation efforts.

Keywords: Coral detection; Deep learning; YOLOv8; Ghost module.

1. Introduction

Climate change, overfishing, and pollution threaten the coral reefs and are the causes of the decline of coral reefs worldwide [1], [2], [3], [4]. The importance of coral reefs varies, such as protecting the coastal communities from hydrodynamic activities, supporting the continuity of recreational activities and biodiversity, and enhancing fisheries productivity [5]. Saving the coral reefs is one of the actions towards preserving the ecosystem under the sea [1] and maintaining the sustainable coastal development of people who depend on the coral reef ecosystem [3].

Several actions have been taken to mitigate coral reef decline, including monitoring. Coral reef monitoring is crucial for detecting the ecological changes affecting coral reefs, quantifying and documenting the coral losses that have already occurred, and collecting data that serves as a foundation for research and reef restoration initiatives [6]. Monitoring the health of coral reefs can help researchers understand their current health and the causes of damage to them [7]. Monitoring coral reefs can be done through several approaches, such as photographic and satellite-based data [8]. Such data can be used for further analysis regarding protecting and restoring coral reefs.

Aside from aiding humans in this era of technology, we can utilize technology to tackle environmental threats. Artificial intelligence has emerged to help us with daily tasks for various purposes, such as in industry [9], education [10], agriculture [11], and the monitoring of marine life [12]. Image data analysis offers a valuable tool for monitoring coral reef health. However, while such monitoring is essential, current methods face significant challenges. These include the labor-intensive nature of manually analyzing images and the time-consuming process of evaluating large or expansive reef areas [13], [14]. Therefore, a solution that utilizes artificial intelligence can be proposed to tackle these challenges.

Deep learning is among the most influential technological advancements, particularly in applications like computer vision [13]. Computer vision can speed up the detection of coral reefs and reduce the need for human labor [13], [15] and act as a non-invasive approach for coral monitoring [16]. In recent years, CNN-based object detection algorithms have emerged as powerful tools, renowned for their performance and adaptability across diverse applications. Researchers can leverage this advancement to develop a robust and automated coral detection system [14], [15], paving the way for future integration into autonomous underwater vehicles (AUVs) [17].

Inspired by the possibility of utilising a deep learning-based object detection model to detect corals, this work proposes a model to detect corals within underwater imagery. The proposed model is based on the renowned YOLOv8 architecture [18]. However, we propose utilising the Ghost module and the Ghost bottleneck proposed by [19] and making changes to the original YOLOv8 architecture. The Ghost module reduces the computational cost by adopting cheap linear operations, which hopefully will reduce the size of the trained model. This paper also aims to contribute to conserving and monitoring coral reefs by developing an efficient yet well-performed coral detection model.

2. Related Works

Several works have proposed monitoring and protecting marine ecosystems, specifically coral reefs. Lu et al.'s work proposed a coral detection model based on the YOLO architecture, specifically the YOLOv10 [15]. The work made several changes to the original YOLOv10 architecture, such as modifying the C2f module and proposing the Multi-Path Fusion Block (MPFB) to optimise feature extraction. Another key change to the architecture is integrating GSConv and VoV-GSCSP modules into the neck, thus improving feature fusion and multi-scale detection. The proposed architecture was trained and tested using a custom coral detection dataset and achieved an mAP50 of 81.9. This result outperforms YOLO architectures, such as YOLOv5, YOLOv6, YOLOv8, YOLOv9, and the original YOLOv10.

Zhang et al. propose a CNet, a coral segmentation model. The architecture comprises two symmetric input branches and a fused input branch based on ACNet [20]. A pre-trained ResNet encoder was employed as the RGB branch, and a VGG encoder was employed as the depth branch. Another distinction made within the architecture is the use of ShapeConv, which facilitates the integration of shape features from the RGB and depth images. The proposed architecture achieved a mIoU of 81.33 on the custom coral dataset.

Rajan and Damodaran proposed an MAFFN_YOLOv5 to detect coral health conditions [13]. As the name implies, the proposed architecture was based on the YOLOv5 architecture. The key change is to modify the neck section of the architecture by utilizing a multi-scale feature fusion attention network. The model achieved a mAP of 90.72% on the collected dataset, outperforming YOLOX, YOLOR, and the original YOLOv5.

Lastly, Ouassine et al. propose a model based on YOLOv5 to detect coral within underwater images [14]. The work compares the performance of four variants of YOLOv5: YOLOv5 nano, small, medium, and large. The four variants of YOLOv5 were trained using a custom underwater image dataset of coral. The highest performance in terms of mAP was achieved by the YOLOv5-large (YOLOv5l), which was 47.4. However, the model that achieved the highest mAP is relatively large (92.8 MB).

This work was inspired by Ouassine et al. [14]. Previous work has not explored using other YOLO architectures, such as YOLOv6 [21], YOLOv7 [22], YOLOv8 [18], YOLOv9 [23], and YOLOv10 [24];

therefore, this work aims to continue experimenting with utilising these architectures. We also propose the YOLOv8 with Ghost modules and bottleneck, and compare it with the other architectures.

3. Materials and Methods

This work utilises the coral dataset proposed by Ouassine et. al [14]. The original dataset consists of 575 underwater coral images. The coral species in the image are in the Reunion and Scattered Islands, and the photos were taken under different lighting conditions and depths. The dataset consists of only one label, coral, without further distinguishing different species of coral in the provided labels. The dataset consists of train (478 images), validation (64 images), and test split (33 images). We employ several augmentations to train our model: rotation, flipping, increasing or decreasing brightness, and blurring. The rotation was performed clockwise or counterclockwise with a degree of 15º. The flipping was performed horizontally or vertically, each with a probability of 50%. The brightness was increased or decreased by 15%, and the blurring was performed up to 1.5 pixels. Each of the augmentations was done randomly by utilizing the tools in the Roboflow platform. The resulting train images were expanded into a total of 1230 images.

The leading architecture we use in this work is the YOLOv8 [18]. YOLOv8 is a renowned object detection architecture that has achieved state-of-the-art performance with real-time detection capability and presents promising applications [25]. YOLOv8 proposed a new C2f module to enhance the performance of the previous YOLO versions and offer efficient architecture [25], [26]. Figure 1 shows the YOLOv8 architecture. The CBS module refers to a group of Convolutional Layers, Batch Normalization (BN), and Sigmoid Linear Unit (SiLU) activation. The proposed C2f module denotes a cross-stage partial bottleneck with two convolutional layers [27].

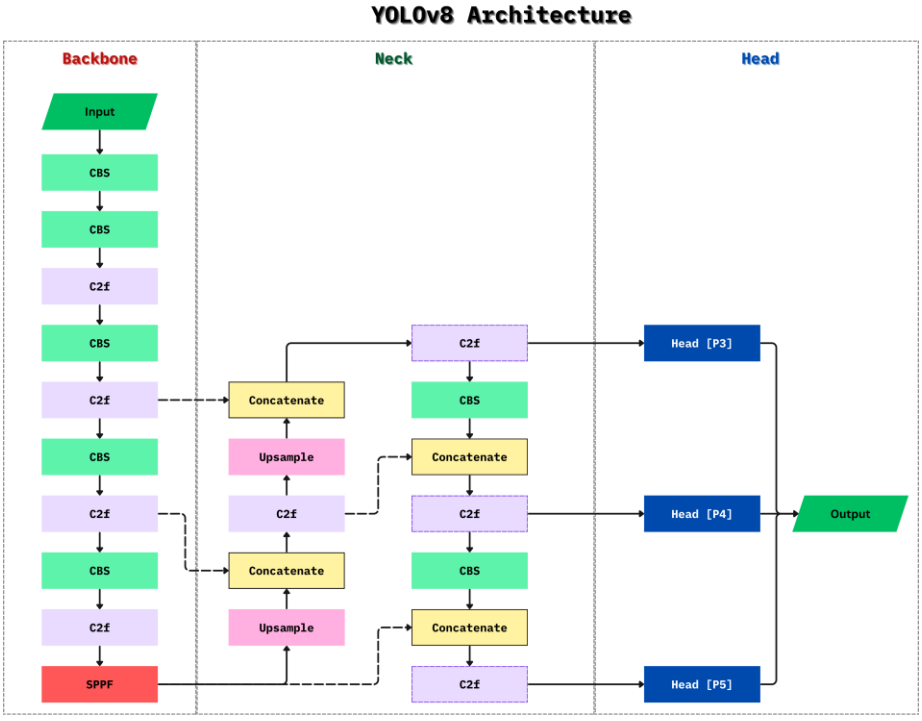

Figure 1. The architecture of YOLOv8 [18]

The YOLO architecture, as illustrated in Figure 1, has three sections: backbone, neck, and head. Each section is intended for a specific reason. The backbone is designed for feature extraction, the neck section for feature fusion, and the head section for outputting the prediction. The feature extraction section extracts scaled features from an image, utilizing the C2f and SPPF (Spatial Pyramid Pooling Fast)

modules. The extracted features are then fused to combine high-level and low-level features, thus enhancing the capability to detect objects at different scales. Lastly, the head section, which consists of three parts, is intended for multi-scale prediction to achieve accurate predictions for small, medium, and large target objects. The output of the YOLOv8 architecture is the position of the objects (labelled by a bounding box), the corresponding classes, and confidence scores. The YOLOv8 architecture comes in four different varieties based on their parameters: nano, small, medium, large, and extra-large. This work focuses on utilizing the nano and small models. The YOLOv8 nano (3.156 million parameters) and small (11.167 million parameters) architectures offer ease of deployment on mobile or CPU-only devices, which have limited computational resources, such as a GPU [28].

However, this work explores the utilization of the ghost module [19] within the current architecture of YOLOv8, specifically the small variant. This choice was intended to reduce the model size of YOLOv8 small, which is relatively bigger than other modern architectures such as YOLOv9 and YOLOv10 small.

The ghost module was proposed by [19], reducing the computational cost by adopting cheap linear operations. Several ghost modules are then stacked to create a ghost bottleneck. A deep neural network called GhostNet, which consists of ghost bottlenecks, achieved higher performance than MobileNetV3, MobileNetV2, EfficientNet, and ShuffleNetV2, while maintaining efficiency and being lightweight [29]. The structure of the ghost module and ghost bottleneck are shown in Figures 2 and 3, respectively.

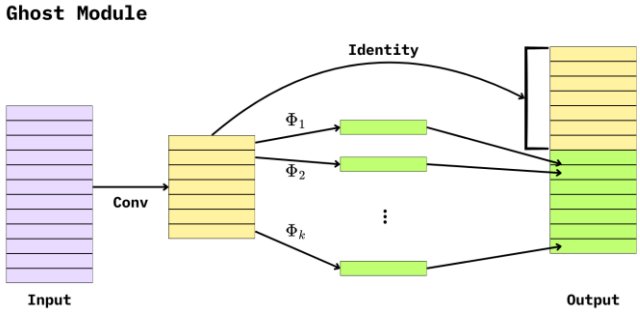

Figure 2. The structure of the Ghost Module [19]. Φ denotes the cheap linear operation.

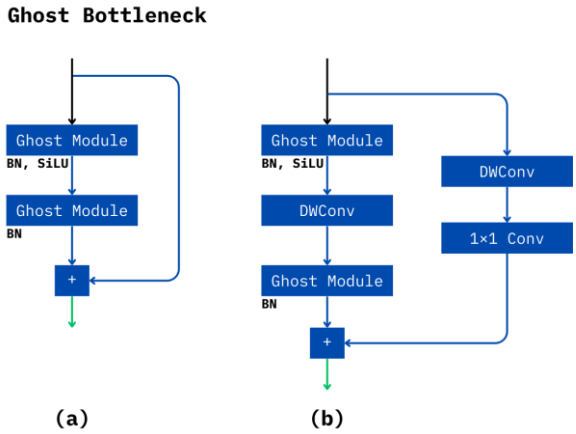

Figure 3. The structure of the Ghost Bottleneck. (a) The Ghost bottleneck with stride = 1 and (b) the Ghost Bottleneck with stride = 2 [19].

The cheap linear operation (Φ) within the Ghost Module (see Figure 2), uses depth-wise convolution with a fixed kernel size [19]. Thus, the proposed Ghost Module and Bottleneck (see Figure 3) reduce computational costs for deep neural networks. Figure 4 depicts the integration of the Ghost Module and Bottleneck within the YOLOv8 architecture. Within the YOLOv8 architecture, the C2f module is

exchanged with the C3Ghost module. The C3Ghost is a cross-stage partial module with a ghost bottleneck [30]. Aside from using the ghost module and C3Ghost, the other configuration is similar to the original YOLOv8 architecture.

Each model was evaluated based on the test set regarding precision, recall, and mean Average Precision (mAP). The precision, recall, and mAP were calculated using equations (1), (2), and (3), respectively. TP corresponds to the number of true positives, FP corresponds to the number of false positives, FN corresponds to the number of false negatives, and AP_i corresponds to the average precision for class i.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3}$$

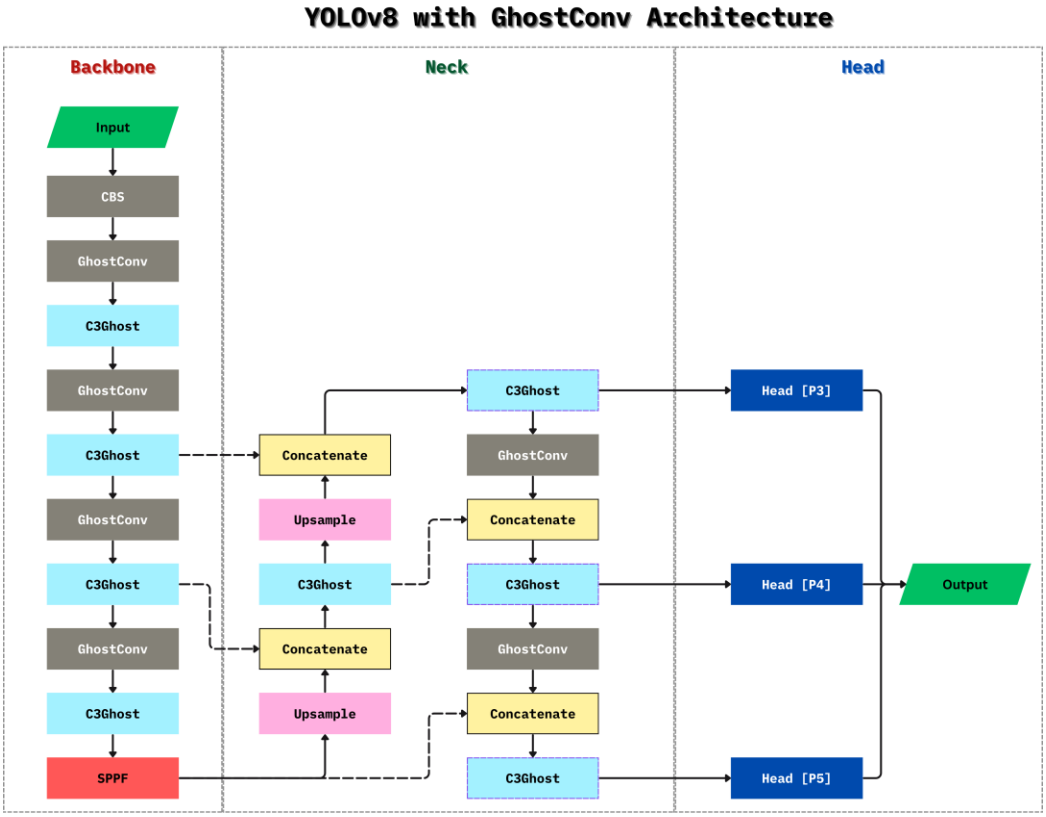

Figure 4. The architecture of YOLOv8 with the ghost module [18], [19]

4. Results and Discussion

This section discusses the training results and evaluates each model. Each model was trained for 300 epochs. The Stochastic Gradient Descent (SGD) was the optimizer, with a learning rate of 1e-2, weight decay of 5e-4, and momentum of 0.937. The size of each image is set to 640 × 640 pixels, whether for the training, validation, or test set. The training was done on the Kaggle platform utilizing the Tesla P100

GPU with a memory capacity of 16 GB. A comparison was made between the YOLOv8 nano and small, a modified YOLOv8 small with the Ghost module and bottleneck, YOLOv6 small, YOLOv7, YOLOv9 small, and YOLOv10 small. Figure 5 shows each model's training results in terms of the mAP50 using the hyperparameters mentioned earlier.

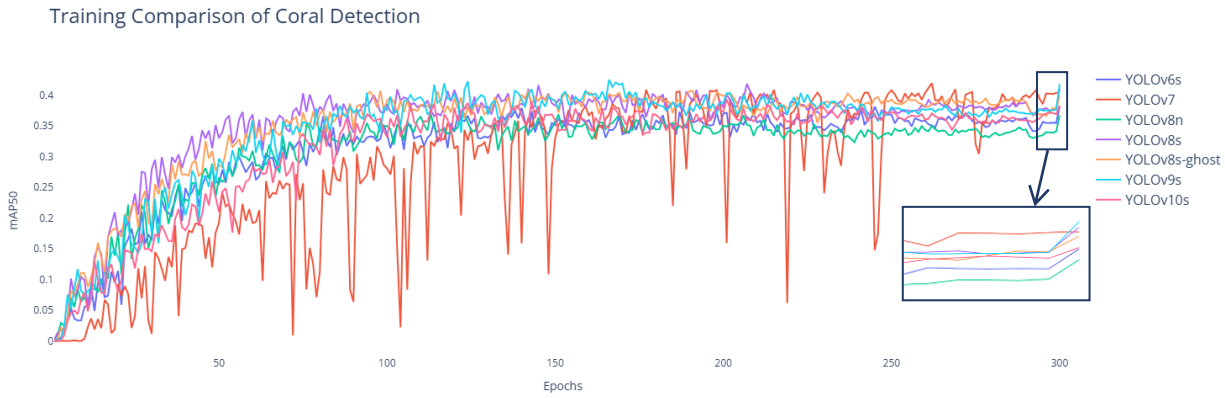

Figure 5. The results of training each model.

The results shown in Figure 5 indicate that each model generally improves its performance. It is evident that at the first 50 epochs, each model, except YOLOv7, experienced an increase from nearly 0 to 0.2. From there onwards, the mAP of each model increased and steadily stayed in the range of 0.3 to slightly above 0.4. The YOLOv7 model exhibits significantly fluctuating training progress compared to the other architectures. However, YOLOv8n can be seen as underperforming in this case, followed by YOLOv6 small. YOLOv9 achieved the highest result, followed by YOLOv8 small, YOLOv7, YOLOv8 small with ghost module, and YOLOv10 small. In addition to the training results, each model is evaluated using the validation and the test set. Table 1 shows these results and the size of each model.

Table 1. The results from the validation and test sets, along with the size of each model.

Model	Validation Set			Test Set			Model Size (MB)
	Precision	*Recall*	*mAP50*	*Precision*	*Recall*	*mAP50*	
YOLOv6small	46.841	41.65	38.153	52.612	53.11	51.387	32.9
YOLOv7	45.881	**45.972**	40.464	**69.5**	42.1	45.6	74.8
YOLOv8nano	47.001	41.454	36.701	53.892	45.933	44.919	**6.3**
YOLOv8small	55.742	39.293	41.057	59.815	**54.067**	53.675	22.6
YOLOv8small Ghost	52.08	39.927	39.85	65.238	50.284	**55.88**	12.2
YOLOv9small	**57.021**	40.668	**41.829**	55.927	45.933	48.366	15.3
YOLOv10small	45.412	44.782	38.328	58.127	38.525	42.511	16.6

Starting with the YOLOv6 small model, it achieved a precision of 46.841% on the validation set, a recall of 41.65%, and an mAP50 of 38.153%. On the test set, it achieved performances with a precision of 52.612%, a recall of 53.11%, and an mAP50 of 51.387%. YOLOv6 small has a size of 32.9 MB, making it relatively lightweight. The YOLOv7 achieved a precision of 45.881%, a recall of 45.972%, and an mAP50 of 40.464%. Test set performance of the YOLOv7 is significantly higher in precision (69.5%), but it has lower recall (42.1%) and mAP50 (45.9%). YOLOv7 (74.8 MB) is the largest among other

models. With such a size, deploying a relatively large model such as YOLOv7 may impact deployment efficiency.

The YOLOv8 nano, with a size of only 6.3 MB, is the smallest model among others. It has relatively good performance and can compete with other larger models. YOLOv8 nano achieved precision, recall, and mAP of 47.001%, 41.454%, and 36.701% on the validation set. The test set results achieved by the YOLOv8 nano are good as well, with a precision of 53.892%, a recall of 45.933%, and an mAP50 of 44.919%. Having such a miniature size, YOLOv8 nano might be ideal for resource-constrained environments.

The YOLOv8 small shows strong performance compared to others. On the validation set, it achieved the highest precision (55.742%) and mAP50 (41.057%) among all models, with a recall of 39.293%. On the test set, the performance of the YOLOv8 small remained strong, with a precision of 59.815%, a recall of 54.067%, and a mAP50 of 53.675%. The size of the YOLOv8 small (22.6 MB) is significantly higher than that of the YOLOv8 nano. However, with such performance, YOLOv8 small shows a balance between performance and efficiency.

The YOLOv8 small with Ghost modules achieved a precision of 52.08%, a recall of 39.927%, and a mAP50 of 39.85% on the validation set. On the test set, it achieved the highest mAP50 (55.88%) among all models, with precision at 65.238% and recall at 50.284%. The YOLOv8 small with Ghost module is 12.2 MB, which is significantly smaller (10.4 MB smaller) than the original YOLOv8 small. Therefore, it offers a compact yet high-performing alternative.

Compared to other models, YOLOv9 small achieves the highest mAP50 (41.829%) and precision (57.021%) on the validation set. However, on the test set, it achieves a mAP50 of 48.366%, significantly lower than both YOLOv8 small with the Ghost module. Lastly, YOLOv10 small achieves a mAP50 of 38.328% on the validation set, which is lower than most of the models except YOLOv6 and YOLOv8nano.

Overall, the trained models performed well. However, YOLOv8small achieves a good balance between precision and recall on both validation and test sets. YOLOv8small Ghost stands out with the highest mAP50 on the test set (55.88%) and a relatively small size (12.2 MB). However, on the test set, YOLOv10 small achieve a relatively good mAP50 of 42.511%. In terms of model size, YOLOv9 small (15.3 MB) and YOLOv10 small (16.6 MB) are significantly smaller than YOLOv8 small. However, after modification, YOLOv8 small with the Ghost module was reduced in size.

This work surpasses the original study [14], which utilized the YOLOv5 large architecture to achieve a mAP of 47.4%. By contrast, our implementation of the YOLOv8 small model with a Ghost module attains a significantly higher mAP of 55.88%, marking an improvement of 8.48 percentage points over prior results.

We compare the performance results of our work and model size with those of previous research and the original work. Table 2 shows this comparison in terms of mAP, model size (in MB), and parameters (in millions).

Table 2. Performance comparison on coral detection

Authors	Methods	Results (%)	Model Size (MB) / Params (Millions)
Lu et al. [15]	YOLOv10 with Multi-Path Fusion Block (MPFB) GSConv, and VoV-GSCSP modules	81.9	- / 2.4
Rajan and Damodaran [13]	MAFFN_YOLOv5	90.72	89 / 8.3
Ouassine et al. [14]	YOLOv5	47.4	92.8 / -
Ours	YOLOv8 small	53.675	22.6 / 11.1
	YOLOv8 is small with the Ghost modules	55.88	12.2 / 5.9

Compared with previous works, our work exceeds the original work of Ouassine et al. [14] in terms of mAP, with a difference of 8.48%. Our model is significantly smaller, with only 12.2 MB for YOLOv8 small with the Ghost modules and 22.6MB for the original YOLOv8 small, compared to 92.8 MB of YOLOv5. However, compared to the work of Lu et al. [15] and Rajan and Damodaran [13], the mAP of this work differs greatly since the dataset used within this work differs from that of the other work. Regarding model size and parameters, the YOLOv8 small with Ghost modules is significantly smaller than the MAFFN_YOLOv5. However, compared to the model proposed by Lu et al. [15], our model still has significantly more parameters. Figure 6 visualizes detection results from each model on a representative test set, paired with their ground truth annotations.

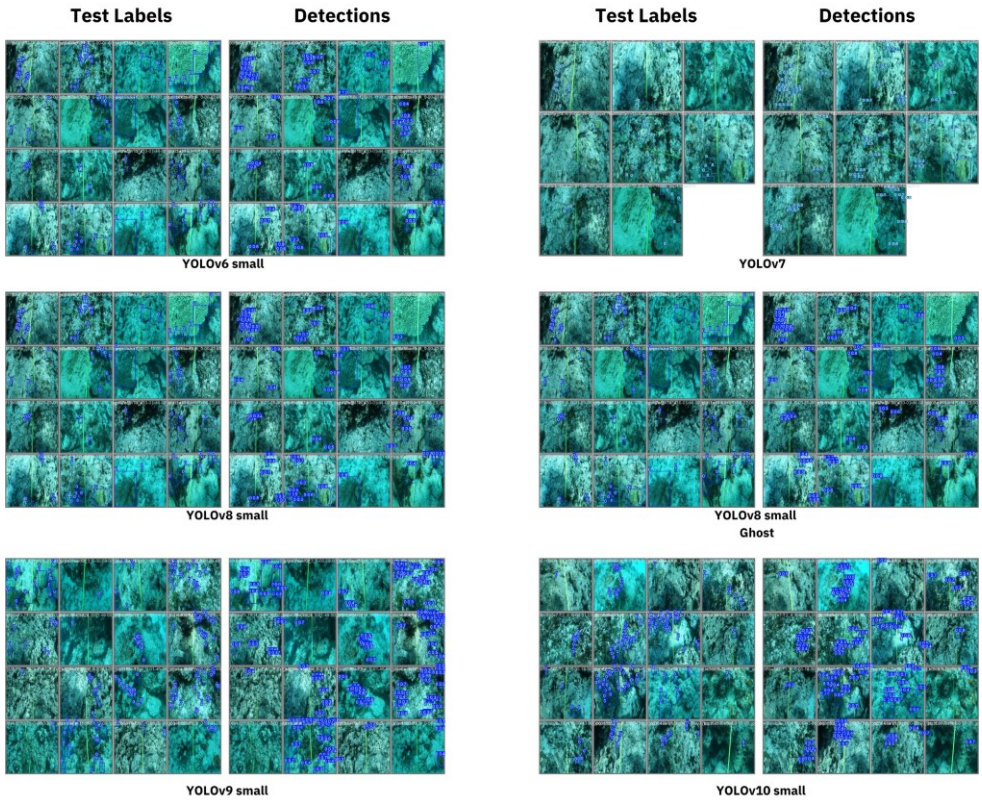

Figure 6. Detection results from each model on a representative test image with the corresponding ground truth annotations.

The label 0 indicates coral class. Each model can detect most of the coral within the representative test image with a relatively acceptable degree of confidence, ranging from 30% to 90%. There are cases where coral is challenging to detect. These challenges arise from the small size of the corals or their considerable distance from the camera, both of which reduce their visibility in the image. Although the models could detect several small-sized corals, not every small coral can be detected. Another challenge is regarding the corals that are mainly covered by shadow. This condition tends to make the model unable to detect those corals properly. Aside from these conditions, the models made several missed detections (false negatives) and false positive detections, thus affecting the performance evaluation.

5. Conclusion

This study explores the development of a deep learning-based coral detection for underwater imagery. The proposed model is based on the renowned YOLOv8 architecture for object detection. Aside from

utilising the original YOLOv8 architecture, we proposed incorporating the Ghost modules and Ghost bottlenecks to replace the original modules within the YOLOv8 architecture, specifically the YOLOv8 small. The results of the training and evaluation of each model were satisfactory for the dataset used in this work. The highest mAP on the test set was achieved by the YOLOv8 small with ghost modules, which is 55.88%, although the original YOLOv8 small model offers a relatively balanced performance in precision, recall, and mAP. However, the modified YOLOv8 small with ghost modules is significantly smaller in model size and parameters than the original YOLOv8 small model, thus offering a compact yet high-performing model. Significant challenges were found in detecting corals within underwater imagery. The first is the size of the coral, which can be small due to the characteristics of the coral or being far from the underwater camera. The second challenge is regarding the coral being covered by shadows, which causes the model to miss detecting it. These problems were found and hopefully can be addressed in future works. A large dataset of coral detection may also be proposed to create a robust and well-performed model to be deployed within autonomous underwater vehicles (AUVs). Lastly, modifying other object detection architectures may be one of the drivers for future work.

6. References

[1] N. M. D. Schiettekatte *et al.*, "Biological trade-offs underpin coral reef ecosystem functioning," *Nature Ecology & Evolution 2022 6:6*, vol. 6, no. 6, pp. 701–708, Apr. 2022, doi: 10.1038/s41559-022-01710-5.

[2] H. T. Pinheiro *et al.*, "Plastic pollution on the world's coral reefs," *Nature 2023 619:7969*, vol. 619, no. 7969, pp. 311–316, Jul. 2023, doi: 10.1038/s41586-023-06113-5.

[3] T. D. Eddy *et al.*, "Global decline in capacity of coral reefs to provide ecosystem services," *One Earth*, vol. 4, no. 9, pp. 1278–1285, Sep. 2021, doi: 10.1016/j.oneear.2021.08.016.

[4] J. A. Cardenas *et al.*, "A systematic review of robotic efficacy in coral reef monitoring techniques," *Mar Pollut Bull*, vol. 202, p. 116273, May 2024, doi: 10.1016/J.MARPOLBUL.2024.116273.

[5] D. R. Bellwood, S. J. Brandl, M. McWilliam, R. P. Streit, H. F. Yan, and S. B. Tebbett, "Studying functions on coral reefs: past perspectives, current conundrums, and future potential," *Coral Reefs 2024 43:2*, vol. 43, no. 2, pp. 281–297, Feb. 2024, doi: 10.1007/S00338-024-02474-Z.

[6] P. J. Edmunds, "Why keep monitoring coral reefs?," *Bioscience*, vol. 74, no. 8, pp. 552–560, Sep. 2024, doi: 10.1093/BIOSCI/BIAE046.

[7] J. Teague, D. A. Megson-Smith, M. J. Allen, J. C. C. Day, and T. B. Scott, "A Review of Current and New Optical Techniques for Coral Monitoring," *Oceans 2022, Vol. 3, Pages 30-45*, vol. 3, no. 1, pp. 30–45, Jan. 2022, doi: 10.3390/OCEANS3010003.

[8] A. Apprill *et al.*, "Toward a New Era of Coral Reef Monitoring," *Environ Sci Technol*, vol. 57, no. 13, pp. 5117–5124, Apr. 2023, doi: 10.1021/acs.est.2c05369.

[9] Z. Jan *et al.*, "Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities," *Expert Syst Appl*, vol. 216, p. 119456, Apr. 2023, doi: 10.1016/J.ESWA.2022.119456.

[10] H. Galindo-Domínguez, N. Delgado, D. Losada, and J. M. Etxabe, "An analysis of the use of artificial intelligence in education in Spain: The in-service teacher's perspective," *Journal of Digital Learning in Teacher Education*, vol. 40, no. 1, pp. 41–56, 2024, doi: 10.1080/21532974.2023.2284726.

[11] M. A. Hamed, M. F. El-Habib, R. Z. Sababa, M. M. Al-Hanjor, B. S. Abunasser, and S. S. Abu-Naser, "Artificial Intelligence in Agriculture: Enhancing Productivity and Sustainability," 2024. Accessed: Jan. 31, 2025. [Online]. Available: https://philpapers.org/rec/HAMAII-2

[12] A. Mandal and A. R. Ghosh, "AI-driven surveillance of the health and disease status of ocean organisms: a review," *Aquaculture International*, vol. 32, no. 1, pp. 887–898, Feb. 2024, doi: 10.1007/s10499-023-01192-7.

[13] S. K. S. Rajan and N. Damodaran, "MAFFN_YOLOv5: Multi-Scale Attention Feature Fusion Network on the YOLOv5 Model for the Health Detection of Coral-Reefs Using a Built-In Benchmark Dataset," *Analytics 2023, Vol. 2, Pages 77-104*, vol. 2, no. 1, pp. 77–104, Jan. 2023, doi: 10.3390/ANALYTICS2010006.

[14] O. Younes *et al.*, "Automatic Coral Detection with YOLO: A Deep Learning Approach for Efficient and Accurate Coral Reef Monitoring," 2024, pp. 170–177. doi: 10.1007/978-3-031-50485-3_16.

[15] Z. Lu, L. Liao, X. Xie, and H. Yuan, "SCoralDet: Efficient real-time underwater soft coral detection with YOLO," *Ecol Inform*, vol. 85, p. 102937, Mar. 2025, doi: 10.1016/J.ECOINF.2024.102937.

[16] M. Li, H. Zhang, A. Gruen, and D. Li, "A survey on underwater coral image segmentation based on deep learning," *Geo-spatial Information Science*, pp. 1–25, May 2024, doi: 10.1080/10095020.2024.2343323.

[17] A. Chowdhury, M. Jahan, S. Kaisar, M. E. Khoda, S. M. A. K. Rajin, and R. Naha, "Coral Reef Surveillance with Machine Learning: A Review of Datasets, Techniques, and Challenges," *Electronics 2024, Vol. 13, Page 5027*, vol. 13, no. 24, p. 5027, Dec. 2024, doi: 10.3390/ELECTRONICS13245027.

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[19] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features from Cheap Operations," Nov. 2019, Accessed: Oct. 19, 2024. [Online]. Available: http://arxiv.org/abs/1911.11907

[20] H. Zhang, M. Li, J. Zhong, and J. Qin, "CNet: A Novel Seabed Coral Reef Image Segmentation Approach Based on Deep Learning," 2024. Accessed: Jan. 31, 2025. [Online]. Available: https://mooreaidea.ethz.ch

[21] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," Sep. 2022, Accessed: Sep. 07, 2023. [Online]. Available: https://arxiv.org/abs/2209.02976v1

[22] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022, Accessed: Sep. 07, 2023. [Online]. Available: https://arxiv.org/abs/2207.02696v1

[23] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," Feb. 2024, Accessed: Oct. 05, 2024. [Online]. Available: https://arxiv.org/abs/2402.13616v2

[24] A. Wang *et al.*, "YOLOv10: Real-Time End-to-End Object Detection," May 2024, Accessed: Jun. 02, 2024. [Online]. Available: http://arxiv.org/abs/2405.14458

[25] B. Zhao, Y. Zhou, R. Song, L. Yu, X. Zhang, and J. Liu, "Modular YOLOv8 optimization for real-time UAV maritime rescue object detection," *Scientific Reports 2024 14:1*, vol. 14, no. 1, pp. 1–12, Oct. 2024, doi: 10.1038/s41598-024-75807-1.

[26] M. Hussain, "YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO," *IEEE Access*, vol. 12, pp. 42816–42833, 2024, doi: 10.1109/ACCESS.2024.3378568.

[27] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction 2023, Vol. 5, Pages 1680-1716*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/MAKE5040083.

[28] X. Zhai, Z. Huang, T. Li, H. Liu, and S. Wang, "YOLO-Drone: An Optimized YOLOv8 Network for Tiny UAV Object Detection," *Electronics (Basel)*, vol. 12, no. 17, p. 3664, Aug. 2023, doi: 10.3390/electronics12173664.

[29] J. Sun, T. Xu, and Y. Yao, "An enhanced GhostNet model for emotion recognition: leveraging efficient feature extraction and attention mechanisms," *Front Psychol*, vol. 15, p. 1459446, Apr. 2024, doi: 10.3389/FPSYG.2024.1459446/BIBTEX.

[30] C. Y. Wang, H. Y. Mark Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 1571–1580, Nov. 2019, doi: 10.1109/CVPRW50498.2020.00203.