IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 19, 2022, accepted May 9, 2022, date of publication May 12, 2022, date of current version May 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174583

Optimizing the Hyperparameter Tuning of YOLOv5 for Underwater Detection

IZA SAZANITA ISA[1], (Member, IEEE), MOHAMED SYAZWAN ASYRAF ROSLI[1], UMI KALSOM YUSOF[2], MOHD IKMAL FITRI MARUZUKI[1], (Member, IEEE), AND SITI NORAINI SULAIMAN[1], (Senior Member, IEEE)

[1]Center of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, Cawangan Pulau Pinang, Permatang Pauh, Penang 13500, Malaysia
[2]School of Computer Sciences, Universiti Sains Malaysia, Gelugor, Penang 11800, Malaysia

Corresponding author: Iza Sazanita Isa (izasazanita@uitm.edu.my)

This work was supported in part by the Grants from the Ministry of Higher Education (MOHE), Malaysia under the Fundamental Research Grant Scheme (FRGS), under Grant FRGS/1/2019/TK04/UITM/02/19; in part by the Research Management Institute; and in part by Universiti Teknologi MARA (UiTM), Cawangan Pulau Pinang.

ABSTRACT This study optimized the latest YOLOv5 framework, including its subset models, with training on different datasets that differed in image contrast and cloudiness to assess model performances based on quantitative metrics and image processing speed. The hyperparameter in the feature-extraction phase was configured based on the learning rate and momentum and further improved based on the adaptive moment estimation (ADAM) optimizer and the function reducing-learning-rate-on-plateau to optimize the model's training scheme. The optimized YOLOv5s achieved a better performance, with a mean average precision of 98.6% and a high inference speed of 106 frames per second. The ADAM optimizer with a detailed learning rate (0.0001) and momentum (0.99) fine-tuning yielded a sufficient convergence rate (0.69% at 55th epoch) to assist YOLOv5s in attaining a more precise detection for underwater objects.

INDEX TERMS Image processing speed, object recognition, optimization model, tuning hyper-parameter, underwater imaging.

I. INTRODUCTION

Analyzing captured images in video-sequence framing, an image processing of real-time or recorded underwater video becomes imperative for extracting underwater features. However, real-time video processing is technically laborious and time-consuming when performed on a serial processor due to several factors. These factors include images comprising a large data set, and complex operations (e.g., design space complexity and require a huge amount of labelled data) are needed to process these images [1]–[4].

Meanwhile, data-driven classification models like neural networks meet the requirement of automatic system and non-destructive method as a tool for managing underwater biodiversity [5]. However, unlike the atmospheric condition, the underwater scenes comprise degraded illumination, low contrast, and changes in visibility due to turbidity [6]–[8] as depicted in Fig. 1. Therefore, it is crucial to overcome such limitations and challenges for the underwater vision system (UVS) by reducing the constrained scenes.

The associate editor coordinating the review of this manuscript and approving it for publication was Yongjie Li.

II. UNDERWATER OBJECT DETECTION AND CHALLENGES

Although underwater object detection using image capturing is the most accessible approach, this method yielded several challenges on extraction the detailed imaging information through the integrations of marine vessels and robots with advanced imaging technologies. Certain factors contributed by the colors absorbance and scattering in underwater, i.e., water properties and impurities, affected the quality of the photographs captured by the underwater imaging devices [48]. The water light attenuations may include and thus processing of sea imaging data becomes more challenging. Certain studies showed that the existence of certain intrinsic deficiencies is attributed to the appearance of objects and ambient noise in underwater images [49]–[51]. Consequently, it is difficult in a real-time system to distinguish objects from their surroundings in these images.

A. OVERCOME THE CHALLENGES OF UNDERWATER ENVIRONMENT

Complex nature of underwater environment poses biggest challenge towards object detection and recognition of

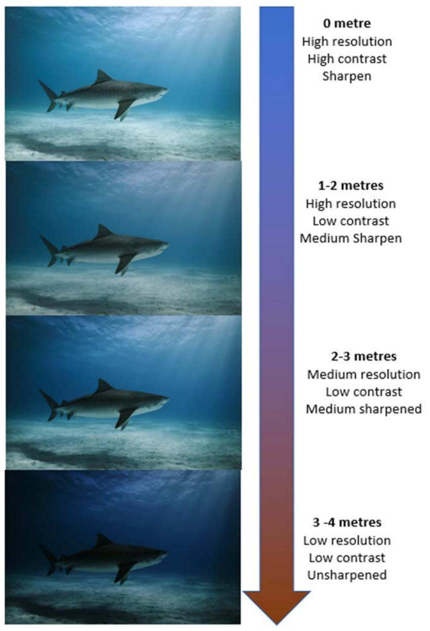


FIGURE 1. The quality of object detection in various luminosities at different distances.

underwater images [55]–[57]. Advancements in camera and video technologies have increasingly evolved into broad applications with high quality and better resolution images, leading to efficient and precise underwater analysis. However, manual extraction and analysis of each frame sequence in the recorded video are labor-intensive, cost ineffectiveness, and prone to fatigue error [2]–[4]. Main challenging in underwater imaging is the limited availability of light, which causes high variability of light intensity while yielding poor luminosity, distortion, and light attenuation [6]–[9]. Other challenges include water murkiness and background confusion of the underwater floor with marine organisms [4], which decrease the accuracy of visual perception on the recorded images. To overcome the challenges, deploying a machine-learning algorithm of a computer vision system could enhance the resolution of underwater imageries due to high turbidity in the underwater environment [10], [42]. Other than improving the quality of underwater images, some of computational algorithms could accelerate the automatic detection in machine learning that will enhance the efficiency of monitoring and analysis [3]. Several imaging methods have been devised to specifically improve the imaging range and quality of underwater imaging systems [58]–[60]. As example, hyperspectral imaging method[61], [62] has been used in underwater object detection since the spectral images available in different bands can provide researchers with a better understanding of image information. Meanwhile, to solve distorted underwater image due to scattering, absorption, color loss, diffraction, polarization or light attenuation, image restoration method has been suggested by [63]. Furthermore, to cater water quality problem in underwater environment, deep learning-based method [64] has seen promising results especially for monitoring a large number of mariculture fish. Therefore, a computer-based methods is highly recommended to solve the underwater environment issues in the exploration of underwater research and engineering.

B. MACHINE LEARNING AND IMAGE PROCESSING FOR UNDERWATER IMAGING

The underwater imaging deals with detecting instances in an image or video, locating their position in a particular frame. However, detection and position location are particularly challenging of accurate underwater classes in recorded images of low resolutions. Several authors adapted the convolutional neural network (CNN) models for developing smart UVSs and achieved satisfactory performance [10]–[14]. These models comprise a stack of distinct layers of convolutional layers, Rectified Linear Unit (ReLU) layers, pooling layers and a fully connected layer that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. By contrast, most of the existing systems are designed for shallow waters or well-illuminated areas in underwater environments [7], [13], [14], where objects are visible.

To date, only a few researchers are investigating to resolve issues of low light intensity and murky water while enhancing image quality processing time to produce a more accurate UVS. However, more computational capacity in classifying object detection and algorithm processing is needed to resolve these issues. Therefore, this study aimed to develop a new optimized model using one of the network architectures for deep learning, i.e., the features extraction stage. In this proposed architecture, features would learn automatically from the input data, eliminating the requirement and engineering effort for hand-crafted feature selection and extraction.

C. UNDERWATER IMAGING USING THE DEEP-LEARNING METHOD

Currently, the most frequently used algorithm for object detection is the model known as You Only Look Once (YOLO) due to its high efficacy and accuracy [15]–[18]. As a subset of the CNN model, YOLO employs a single forward propagation through a neural network to detect objects in real-time, i.e., the entire image is predicted in a single algorithm run for training and validation [43]. This study used the CNN model to predict various class probabilities and bounding boxes simultaneously for two reasons. First, the latest version of algorithm-based YOLO, particularly version 3 or higher, is appropriate for improving object detection with more accurate positioning, faster speed, and more accurate classification. Second, comparative studies on these models for object detection under different underwater environments are yet available.

However, an improvement on YOLO models is more significant based on the tuning parameters of the model's optimizer. Tunable parameters that can improve the YOLO performance are the image of input size, number of epochs, batch size, learning rate, momentum, and activation function.

Also, tuning hyper-parameters, such as learning rate and momentum, during training algorithm would significantly reduce training time and improves performance of the model [44]. Otherwise, the model would be underfitting or overfitting as the effect of poor hyper-parameter tuning. (i.e. increase regularization, increase training speed, cause instability). Thus, this study aimed to generate a robust YOLO model for underwater detection by improving its optimizer, learning rate, and momentum tuning. In this study, among all YOLO series, YOLOv5 has been selected for the model optimization due to several reasons:

i the most advanced target detection algorithm with two content security policy (CSP) structures (CSP1_X and CSP2_X) [52] that able to extract generic features particularly in underwater image,

ii able to adaptively change the depth and width of the network by changing parameters to its own data volume scale [53] (self-adaptation to small underwater objects),

iii can guarantee good training result [54] of highest detection accuracy as proved in this study (Table 5).

D. YOLO VERSION 5 (YOLOv5)

The latest version of the YOLO family is YOLOv5, which is extended from YOLOv4 [34].

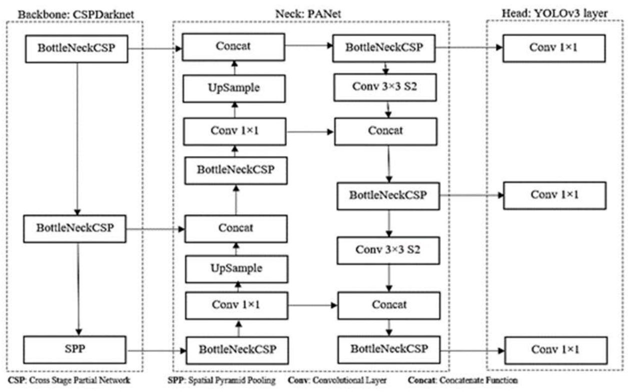


FIGURE 2. The framework and architecture of YOLOv5.

In general, the architecture of YOLOv5 and YOLOv4 is somewhat similar, especially in the backbone, neck, and head (Fig. 2). These two models use the same cross-stage partial, CSP connection in their backbone to generate a rich gradient combination while reducing computational usage. CSP portions the feature map of the base layer, splitting the gradient flow for propagating through different paths. Implementing the CSP connection in a deep network will enhance the learning ability of CNN and hence, improve the accuracy while being light-weighting [35]. Also, CSP clears the computational bottleneck by uniformly distributing the computation in each layer of CNN. Besides, CSP will help downsize the memory costs by using cross-channel pooling to compress the feature maps during the generation of the feature pyramid via the model neck. These feature pyramids help identify identical objects with different sizes and scales. In this respect, YOLOv5 uses the path aggregation network (PANet) as the model neck, and it is particularly beneficial for instance-segmentation in preserving the spatial information accurately. This accuracy helps locate the pixels correctly. Additionally, PANet provides a bottom-up path using clean horizontal connections from lower layers to the top ones (green dot) [36]. The path is called the "shortcut" connection, comprising ten layers only. In the section head, YOLOv5 uses a similar dense prediction as in YOLOv4 and YOLOv3. The final prediction consists of a vector containing the coordinates of the predicted bounding box and its confidence score, and the label. The output processing is carried out by getting rid of boxes with a low score (i.e., below the confidence threshold) and selecting only one box out of several that overlap with each other while detecting the same object based on the architectural variation. YOLOv5 comprises several models, namely YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, with alterations in the depth and width in each model. However, this study used only two models, namely, YOLOv5s and YOLOv5m. In general, each model was scaled for the portions of the network independently with two configurations tuned for each model, namely depth multiple and width multiple. Depth multiple represented the model's depth factor, while the width multiple constituted the layer channel multiple used to scale up the backbone and feature network of the model.

III. THE PROPOSED METHOD

In this study, all YOLO models were trained, validated, and tested using the same platform through Jupyter notebook in Google Collaboratory or Google Colab. This platform allows users to prototype machine-learning models on devices such as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) [22]. The training and testing of all YOLO models were performed using Nvidia Tesla T4 with a memory size of 16 GB Graphics Double Data Rate (GDDR6) and Compute Unified Device Architecture (CUDA) graphic features.

A. COMPARING AND SELECTING YOLO MODELS

This study compared the models of YOLOv3, YOLOv4, and YOLOv5 and selected the best-tested one over three open-source datasets to benchmark. The training and validation were performed using the default configuration of each model. Upon the completion of training, each model was evaluated using the test dataset to assess its performance towards never seen dataset. Table 1 shows the three public datasets used to train the YOLO architecture. These datasets were Open Image Dataset V6 [23], Aquarium Dataset [24], and The Brackish Dataset [25]. An Open Image Dataset V6 were employed in this study to investigate the YOLO models performance towards detection capability in different environments between underwater and non-underwater. The complexity and diversity characterized by all the images could provide ample challenge in testing each YOLO model's architecture in terms of different input image characteristics.

A non-underwater dataset of 2732, 341 and 342 images from the Open Image Dataset V6 used for training, validation and testing respectively. Whilst underwater dataset of 510, 64, 64 images from Aquarium Dataset used for training, validation and testing, meanwhile 11615, 1451 and 1452 image from The Brackish Dataset. The process of separating the dataset fraction was executed by taking an image list as input and randomly splits it according to the provided percentages which are 80% (0.8) training, 10% (0.1) validation and 10% (0.1) test. In data splitting, the ratio value of 1 for train, valid and test was verified before splitting the dataset. This separation verified and compared the performance of YOLOv3, YOLOv4, YOLOv5, and their sub-models that were structured and designed using the CNN multilayer. This performance was indicative of the behavior of YOLO models and tuning the optimizer's hyper-parameters improved the selected model.

TABLE 1. The configuration of the image dataset for non-underwater and underwater environments in the YOLO learning.

Dataset	No of Classes	Object Detection	Total images
Open Image Dataset	6	Car, motorcycle, building, traffic sign, traffic light, and streetlight	3415
Aquarium Dataset	7	Fish, jellyfish, penguin, shark, puffin, stingray, and starfish	638
The Brackish Dataset	6	Big fish, small fish, shrimps, crabs, jellyfish, and starfish	14,518

The YOLO algorithm was fed with 416 × 416 input images, running them through backbone blocks and layers that learned to extract statistical features for locating objects along with their labels. During the training, the batch size was set at 64, representing the number of samples/images propagated through the YOLO network before updating the model's internal parameters. Additionally, all three models used the stochastic gradient descent (SGD) as the default optimizer. During the training, the model tuned the weight of the YOLO model since this parameter would decide the amount of output that could be affected by the input while minimising the loss function through the optimizer. Meanwhile, all YOLO models in this study used a similar head structure, i.e., the head of YOLOv3. In a single-stage detector, the head section performed dense prediction composing the coordinates of the predicted bounding box, the prediction's confidence score, and the class label. In this study, the confidence score was set to 0.25 and Intersection over Union (IoU) threshold 0.5 for the detection. The training was run with the same 65 epochs at an image input size of 416 × 416, a batch size of 64, and the default SGD optimizer. This forward- and backward-pass cycle represented the number of updating the algorithm's parameters. Table 2 summarizes the configuration of all training parameters.

During the training, validation, and testing, the performance of each YOLO model was tested and evaluated through evaluation metrics. Detections were evaluated via ground-truthing. Each frame in the tested images was calculated based on the value of true positive (TP), false positive (FP), and false negative (FN), generating four standard evaluation metrics to determine two performance parameters, i.e., precision and recall, as denoted in Equations 1 and 2, respectively [25].

TABLE 2. The default configuration of training parameters for the YOLO model.

YOLO Model	Learning Rate	Momentum	Activation Function
YOLOv3	0.001	0.900	Leaky ReLU
Tiny-YOLOv3	0.001	0.900	Leaky ReLU
YOLOv4	0.001	0.949	Mish
Tiny-YOLOv4	0.001	0.900	Leaky ReLU
YOLOv5s	0.010	0.937	Hidden layers – Leaky ReLU Final detection layer - Sigmoid
YOLOv5m	0.010	0.937	Hidden layers – Leaky ReLU Final detection layer - Sigmoid

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

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

The precision represents the usefulness of the detection; a high precision indicates that the trained model returns a truly-detected object rather than a falsely detected one, while the recall defines the truly-detected object that the model returns [25]. A high precision shows a low value of FP, while the recall is usually related to a small number of FPs. Therefore, a higher percentage of precision and recall indicates that a model performs better [22]. The model's performance was evaluated using the F1-score given by Equation 3 below [45].

$$\text{F1-score} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

This F1-score represents the harmonic mean of precision and recall, and a higher F1-score shows a better performance. All models were then evaluated with the mean average precision (mAP) using Equation 4 below [25], [45].

$$\text{mAP} = \frac{\sum_{i=1}^{k} \text{AP}_i}{k} \tag{4}$$

Averaging the average precision (AP) for all classes involved in the trained model yields mAP. A sufficiently well-performed model tends to produce a higher accuracy. Finally, the performance of processing rate was calculated using frame per second (FPS) with Equation 5 below [45] to evaluate the model speed in processing the input in real-time applications.

$$\text{FPS} = \frac{\text{Number of Frames}}{\text{Total Detection Time(s)}} \tag{5}$$

In general, the higher the FPS, the faster the model in detecting an object. Upon all training, a robust model was selected via the comparative performance. Hence, the model would be improved based on the tuning of the optimizer to increase the model's precision.

B. TUNING THE HYPER-PARAMETERS OF LEARNING RATE AND MOMENTUM

Fig.3 shows the proposed work for improving the selected model. In this proposed method, the YOLO model was optimized using hyper-parameter tuning.

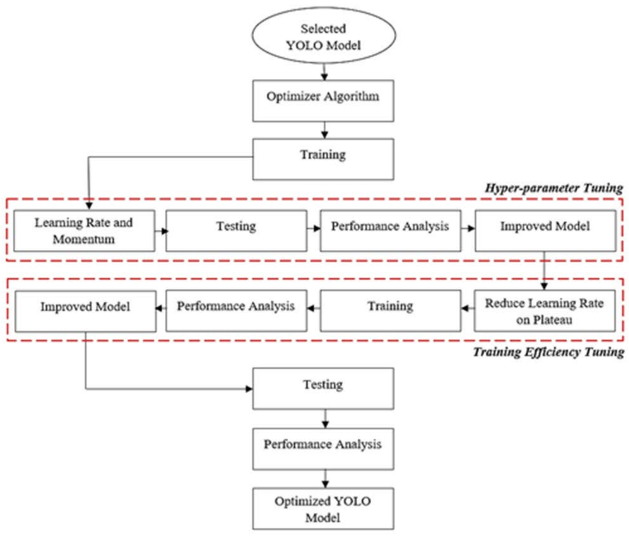


FIGURE 3. The proposed method for optimizing the YOLO model by tuning the hyper-parameter in the optimizer and the learning rate on plateau.

The selected YOLO model was improved based on the optimizer algorithm, focusing on the learning rate and momentum since both features contributed to the performance accuracy and the speed of processing rate. The training hyper-parameter, i.e., the optimizer algorithm, was used to reduce the losses of regression problem while providing estimation as accurately as possible. Since YOLO is a CNN-based neural network that transmits output error in a hidden layer to each neuron through backward propagation, it updates the connection weights of each neuron iteratively [24]. Updating the weight to reduce erroneous values was performed through the gradient descent type of optimiser algorithm. By default, the optimiser in the YOLO model used SGD with momentum. SGD computes the gradient of the cost function for the parameters θ for the entire training dataset, as given in Equation 6 below [46]. SGD minimised the objective function, $J(\theta)$ parameterised by a model's parameters $\theta \in R^d$, and updated the parameters in the opposite direction of the gradient of the objective function, $\nabla\theta J(\theta)$, for the parameters [46]. The learning rate, η, determined the size of the steps to reach a local minimum. Equation 7 below gives the update of a parameter by SGD for each training example, x^i and label y^i [46].

$$\theta = \theta - \eta.\nabla_\theta J(\theta) \tag{6}$$

$$\theta = \theta - \eta.\nabla_\theta J(\theta; x^i; y^i) \tag{7}$$

Equation 8 shows that adding a fraction, γ of the update vector, v_t of the past time step to the current update vector accelerates SGD in relevant directions while dampening oscillations [46]. This modification gives the updates of Equation 9 [46].

$$v_t = \gamma v_{t-1} + \eta\nabla_\theta J(\theta) \tag{8}$$

$$\theta = \theta - v_t \tag{9}$$

The extension of SGD is the adaptive moment estimation (ADAM) [6] that computes the decaying averages of past, m_t, and the past squared gradients of v_t, as shown in Equations 10 and11, respectively [46].

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{10}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{11}$$

Estimates of m_t and v_t would yield the first moment (the mean) and the second moment (the uncentered variance) of the gradients, respectively. These estimates would update the parameter based on the rule of ADAM in Equation 12 [46]. In general, β_1 represented the exponential decay rate for the first moment estimate, β_2 was the exponential decay rate for the second moment estimate, and g was the gradient on the current mini-batch [46]. The ADAM optimiser proposed a default value of 0.9 for β_1, 0.999 for β_2, and is 10^{-8} for ϵ.

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{\upsilon_t}} + \epsilon} + \widehat{m_t} \tag{12}$$

Focusing on optimising the model, this study compared two optimisers for training the selected YOLO model (YOLOv5), focusing on optimising the model. The default SGD optimiser was compared with the implemented ADAM optimiser. The first execution step of ADAM was used to tune the best-fit parameters for model training. During the training, the learning rate and momentum affected the behaviour of ADAM. The tuning involved training for different values for both parameters. A range of learning rates and momentums were experimented with ADAM to determine the best combination value for better training performance for the selected YOLO. The learning rate ranged between 10^{-6} and 1.0 [37], while the momentum and common values used in practice were 0.5, 0.9, and 0.99 [38]. Based on these references, this study employed a combination of the learning rate and momentum, and parameters were named in alphabetical order of upper and lowercase letters (Table 3). The YOLOv5s with ADAM and all combinations of hyper-parameters were trained using the same configuration except for the optimiser, learning rate, and momentum. Besides, the ADAM configuration of zZ was compared with the SGD default value to study the effect of the same learning rate and momentum on different optimisers. Each experimented model was trained using the Brackish dataset. After training with all varying hyper-parameters, training performances were compared. This step was essential for choosing the best-fit combination for the optimizer for the selected YOLO model and later in the subsequent improvement of the model.

TABLE 3. The configuration of the ADAM optimizer for improving the YOLO model.

Optimizer	Learning Rate	Momentum
zZ	0.01*	0.937*
aA	0.001	0.9
aB		0.99
aC		0.999
aD		0.9999
bA	0.0001	0.9
bB		0.99
bC		0.999
bD		0.9999
cA	0.00001	0.9
cB		0.99
cC		0.999
cD		0.9999
dA	0.000001	0.9
dB		0.99
dC		0.999
dD		0.9999

*Default: the SGD optimizer

C. HYPER-PARAMETER TUNING OF THE LEARNING RATE ON A PLATEAU

Besides using ADAM as an optimizer to adapt the learning rate for each weight, this study improved the behaviour of the learning rate on each epoch during the training using the technique of reducing-learning-rate-on-plateau. This technique improved the model accuracy, descending into areas of lower loss by monitoring the loss during the training. The learning rate would be reduced if the loss was stagnant for several epochs. The term plateau was indicative of the point when the change in the loss for training iterations was below the threshold, θ. In short, the curve of epoch against loss became flat and did not improve Since a specific parameter setting was yet available for the YOLOv5 model, therefore, in this study, the implementation was executed using the module of ReduceLROnPlateau in PyTorch. By default, the first training parameter mode was set as a minimum (min) to reduce the initial learning rate (LR_1) once the loss stopped decreasing. Secondly, the factor (by which learning rate will be reduced) was used to reduce the new learning rate (LR_{new}) by this ratio, or given by Equation 13 below [47].

$$\mathrm{LR}_{\mathrm{new}} = \mathrm{LR}_1 \times \mathrm{factor} \tag{13}$$

Thirdly, the patience parameter reduced the learning rate when the model showed no improvement after the 8th epoch. Finally, the threshold measured the new optimum, focusing only on remarkable changes. Table 4 summarises the value for each parameter. In general, a precision-recall (PR) curve interprets the performance metrics for object detection that symbolizes the trade-off between two metrics (precision and recall) through different confidence thresholds [42]. From the trade-off, an object detector model is robust in locating correct bounding boxes if its precision stays high as its recall increases, as shown by the area under the curve. Since detecting underwater animals requires high precision and recall, the area under the curve in a PR curve will need to be as big as possible. In this study, this PR curve was plotted using the validation dataset.

TABLE 4. The tuning of learning parameters.

Parameter	Setting
Mode	Min
Factor	0.1
Patience	10
Threshold	1e-4

IV. RESULT AND DISCUSSION

A. PERFORMANCE OF YOLO MODELS

Fig. 4 shows the test images detected using YOLO models performed on the Open Image V6 dataset. These images consisted of three objects of multiscale ground truth to evaluate the learning capability of each YOLO model for detecting far and a small-scale traffic sign. YOLOv3, Tiny-YOLOv3, and YOLOv5m detected two bounding boxes, while YOLOv4 and Tiny-YOLOv4 could not detect the far-end traffic signs. Thus, YOLOv5s was the only model that could detect all three bounding boxes, contributing to TP value for improving the model precision. Fig. 5 shows the detection output using the testing dataset for the Aquarium Dataset. YOLOv3, YOLOv4, and YOLOv5s detected all fishes and stingrays with multiscale that varying of sizes in the image, and essentially able to detect the targets at different scales. In addition, all the three models are also able to deal with the complexity of differentiating between the stingray and the background. Among all YOLO models that detected the underwater animals (stingray), YOLOv4 showed the highest confidence value of detection (0.99), followed by YOLOv5 (0.90) and YOLOv3 (0.81).

Fig. 6 shows the detection outputs of YOLO models on the Brackish Dataset. In general, tiny models were inefficient in detecting two crabs; Tiny-YOLOv3 located three bounding boxes, and Tiny-YOLOv4 uncovered just one. Tiny-YOLOv3 gave FP to the model, and Tiny-YOLOv4 yielded FN, thus reducing the precision of Tiny-YOLOv3 and the recall of Tiny-YOLOv4.

Table 5 shows the test performance of each YOLO model and its subsets. Compared to other models, YOLOv3 gave high values of precision, while YOLOv5s attained the highest mAP for Open Image Dataset V6 at 56.4%, followed by YOLOv4 at 55.9%.

Meanwhile, YOLOv4 achieved the highest mAP on the Aquarium Dataset, i.e., 78.5%, outperforming YOLOv5s at 75.7%. YOLOv3 yielded the highest precision (82.0%) and Tiny-VOLOv3 the lowest (64.0%). YOLOv4 achieved the highest recall, i.e., 76.0%, followed by YOLOv5s at 75.8%, YOLOv5m at 68.5%, YOLOv3 at 63.0%, Tiny-YOLOv4 at 57.0%, and Tiny-YOLOv3 at 47.0%. In the most challenging underwater dataset, i.e., the Brackish Dataset, YOLOv5s outperformed other models with the highest mAP at 97.7%, and followed by YOLOv5m at 97.5%, YOLOv4 at 97.3%,

TABLE 5. The performance of YOLO models trained on different images of datasets.

Dataset	Model	Precision	Recall	F1-Score	mAP@0.5	FPS	Weight/MB
The Aquarium Dataset	YOLOv3	0.820	0.630	0.710	0.654	53.3	239.0
	Tiny-YOLOv3	0.640	0.470	0.540	0.538	87.0	33.1
	YOLOv4	0.760	0.760	0.760	0.785	54.0	244.3
	Tiny-YOLOv4	0.750	0.570	0.650	0.614	84.7	22.5
	YOLOv5s	0.662	0.758	0.707	0.757	125.0	14.1
	YOLOv5m	0.710	0.685	0.697	0.711	76.9	41.3
The Open Image Dataset	YOLOv3	0.620	0.410	0.490	0.493	53.5	239.0
	Tiny-YOLOv3	0.560	0.310	0.400	0.388	77.7	33.2
	YOLOv4	0.510	0.540	0.520	0.559	57.7	244.3
	Tiny-YOLOv4	0.460	0.410	0.430	0.438	88.0	22.5
	YOLOv5s	0.436	**0.655**	**0.524**	**0.564**	90.9	14.1
	YOLOv5m	0.353	0.598	0.444	0.475	66.7	41.3
The Brackish Dataset	YOLOv3	**0.970**	0.960	**0.965**	0.964	56.3	235.0
	Tiny-YOLOv3	0.810	0.800	0.810	0.872	87.4	33.1
	YOLOv4	0.940	0.970	0.955	0.973	46.6	244.3
	Tiny-YOLOv4	0.940	0.820	0.876	0.886	76.0	22.5
	YOLOv5s	0.904	**0.979**	0.940	**0.977**	106.4	14.1
	YOLOv5m	0.874	**0.976**	0.922	**0.975**	82.0	41.3

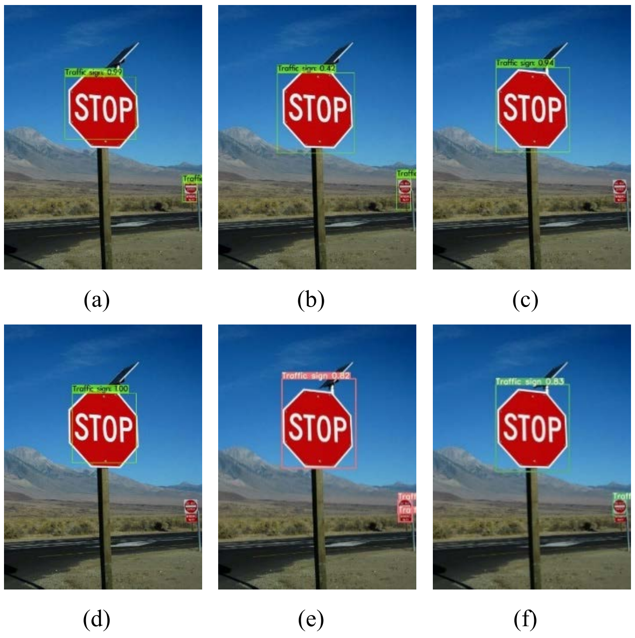


FIGURE 4. Detection outputs tested on the Open Image V6 Dataset: (a) YOLOv3, (b) Tiny-YOLOv3, (c) YOLOv4, (d) Tiny-YOLOv4, (e) YOLOv5s, and (f) YOLOv5m.

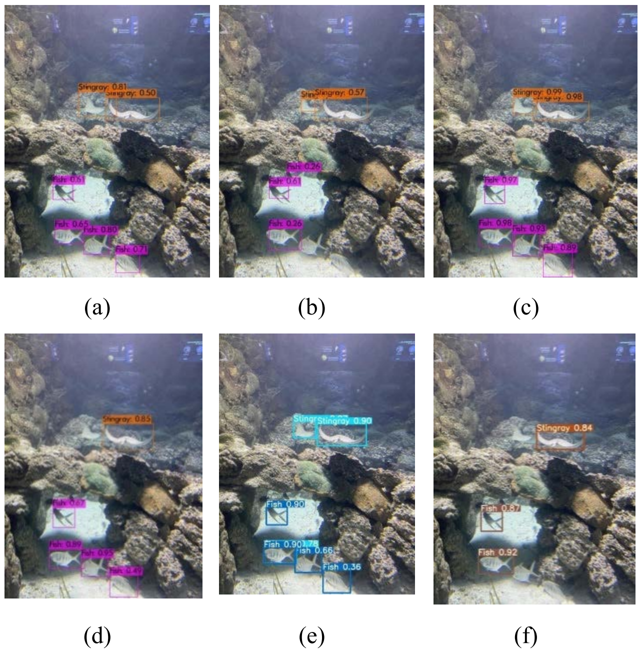


FIGURE 5. Detection outputs tested on the Aquarium Dataset: (a) YOLOv3, (b) Tiny-YOLOv3, (c) YOLOv4, (d) Tiny-YOLOv4, (e) YOLOv5s, and (f) YOLOv5m.

YOLOv3 at 96.4%, Tiny-YOLOv4 at 88.6%, and Tiny-YOLOv3 at 87.2%. Even shallow network models, such as Tiny-YOLOv3 and Tiny-YOLOv4 mAP values above 87%. Despite the blur image of the Brackish Dataset, all models correctly understood the semantic representation pixel by pixel for detecting the object.

In the Open Image Dataset V6, the YOLOv5s model recorded the highest FPS, i.e., 125.0, through Tesla T4 GPU, while YOLOv4 was the lowest at 46.6 in The Brackish dataset. YOLOv5s also outperformed other YOLO models in all datasets with an excellent execution speed. Other primary models, i.e., YOLOv3 and YOLOv4, performed at half of the YOLOv5s speed. Besides, the weight size affected the speed substantially on some models. For example, YOLOv5s, with the smallest weight size of 14.1 MB only, allowed it to execute the deep network rapidly in Open Image dataset (FSP: 125.0). By contrast, all YOLOv3 and YOLOv4, with a weight size of 235.0 MB, 239.0 MB, and 244.3 MB, respectively, yielded a slower FPS, i.e., 56.3, (53.3 and 53.5), and (54, 57.7, and 46.6), respectively. In general, tiny models had a smaller weight size than their primary models. On average, the weight size of Tiny-YOLOv3 models was 86% than YOLOv3, while

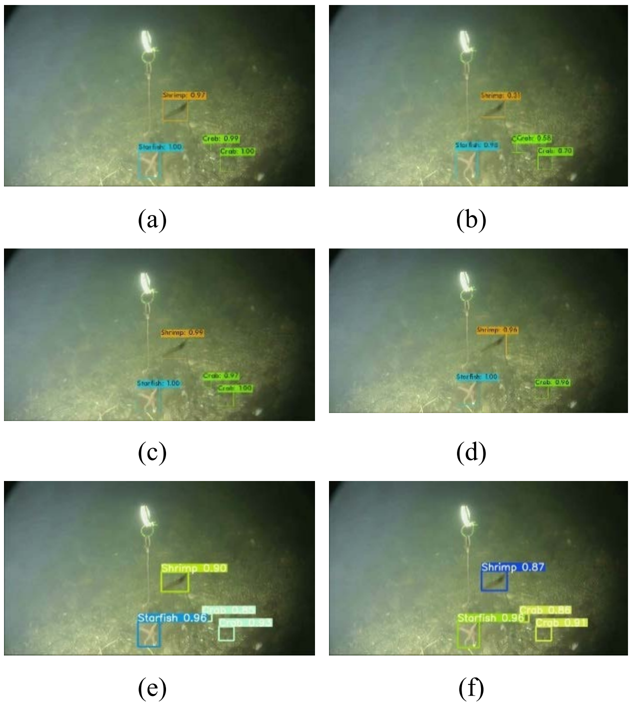


FIGURE 6. Detection outputs tested on the Brackish Dataset: (a) YOLOv3, (b) Tiny-YOLOv3, (c) YOLOv4, (d) Tiny-YOLOv4, (e) YOLOv5s, and (f) YOLOv5m.

Tiny-YOLOv4 models were 90.8% smaller than YOLOv4 models. Tiny models performed better in FPS, i.e., more than 20% faster than primary models. Thus, FPS and weight seemed to be correlated, i.e., the smaller the weight size, the faster the execution speed of a model.

B. TRAINING EFFICIENCY BASED ON THE TUNING OF HYPER-PARAMETERS ON THE LEARNING RATE AND MOMENTUM

YOLOv5s was selected as the model for further development based on its performance. Model optimization was based on the Brackish Dataset only since this study aimed to improve underwater animal detection in low light and murky environments.

Fig. 7 compares the training progress between ADAM and SGD optimizers. The SGD-based model outperformed YOLOv5s-zZ with a faster convergence speed in YOLOv5s-SGD. Besides, at the final epoch, YOLOv5s-zZ did not yield a consistent performance with mAP and barely exceeding 0.9. Such a fluctuating performance was due to inadequate generalization in YOLOv5s-zZ with an increment in the training time. Thus, a good fit in the learning rate and momentum values for one model did not apply to another model of a different optimizer. This behavior indicated that the two SGD and ADAM optimizers performed differently, even though they were set at the same learning rate and momentum value.

Fig. 8 shows the tuning of the learning rate and momentum of each YOLOv5s model. Among all four trained models, YOLOv5s-aA achieved nearly similar performance

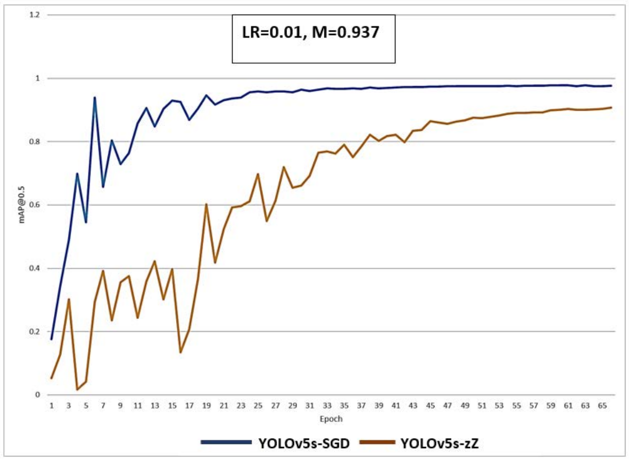


FIGURE 7. Comparison of SGD and ADAM optimizers in the YOLOv5s model (learning rate = 0.01 and momentum = 0.937).

as YOLOv5s-SGD towards the end with less fluctuation starting at the 20th epoch. Meanwhile, YOLOv5s-aC and YOLOv5s-aD converged at a lower speed in execution, indicating that the learning rate and momentum values would need additional epochs to produce better performance for generalization. Meanwhile, Fig. 9 shows that YOLOv5s-bA trained with a learning rate of 0.0001 and a momentum of 0.9 performed excellently. Its performance was comparable to YOLOv5s-SGD, starting from the 34th till the last epoch. This finding became the focal point for tuning the ADAM optimizer in this study because it represented a workable combination of learning rate and momentum. Throughout the training, this combination consistently gave a smoother convergence than YOLOv5s-SGD. By contrast, YOLOv5s-SGD fluctuated at smaller epochs. Also, YOLOv5s-bB with a momentum of 0.99 yielded a well-trained model towards the end of the epoch, despite a slower convergence speed at the 50th epoch.

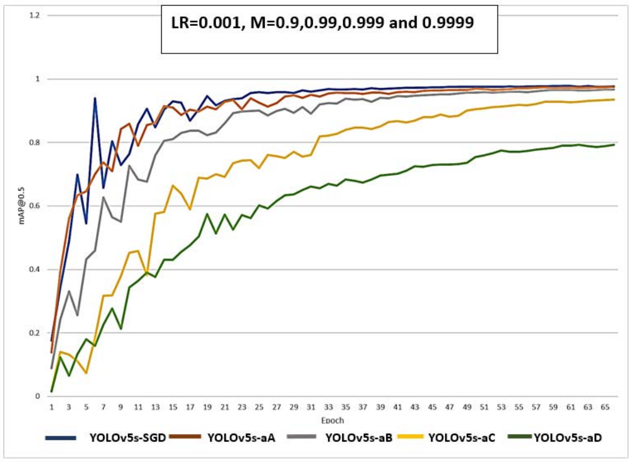


FIGURE 8. Comparison of SGD and ADAM optimizers in the YOLOv5s model (learning rate = 0.001 and momentums = 0.9, 0.99, 0.999, and 0.9999).

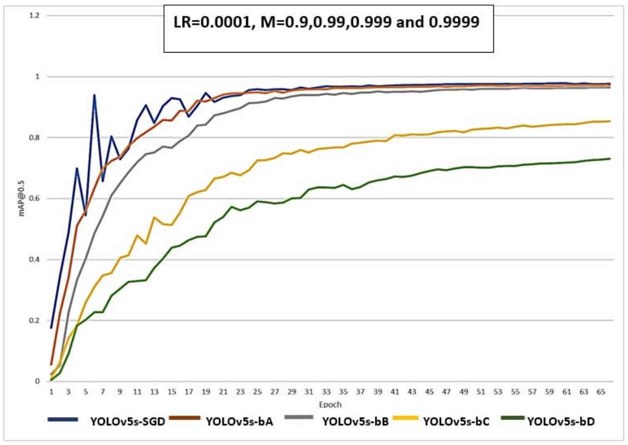


FIGURE 9. Comparison of SGD and ADAM optimizers in the YOLOv5s model (learning rate = 0.0001 and momentums = 0.9, 0.99, 0.999, and 0.9999).

Fig.10 shows that all the four YOLOv5s models with a learning rate of 0.00001 struggled to converge, and their mAP values were below 80%. For example, YOLOv5s-cA yielded a 75% mAP only during the final epoch. Besides, all the models experienced under-fitting with lower mAP value when the training epoch increased. This result indicated that all the models would require more learning iteration with additional training times to give optimal performance. Also, Fig. 11 shows that a learning rate of 0.000001 and momentum variations of 0.9, 0.99, 0.999, and 0.9999 for the ADAM optimizer yielded poor convergence for all the models, i.e., these values were incompatible.

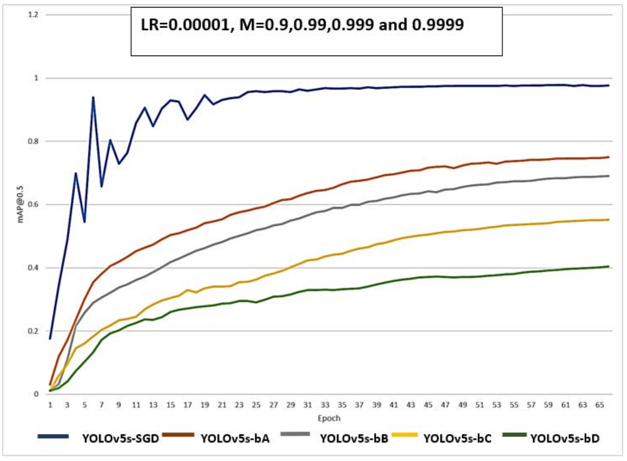


FIGURE 10. Comparison of SGD and ADAM optimizers in the YOLOv5s model (learning rate = 0.0001 and momentums = 0.9, 0.99, 0.999, and 0.9999).

Table 6 tabulates the testing performance using different combinations of learning rates and momentums on all the trained YOLOv5s models. YOLOv5s-zZ yielded a slightly lower mAP value (91.5%) than YOLOv5s-SGD (97.7%) despite these two models having a similar learning rate and momentum. YOLOv5s-zZ struggled to generate a fast and

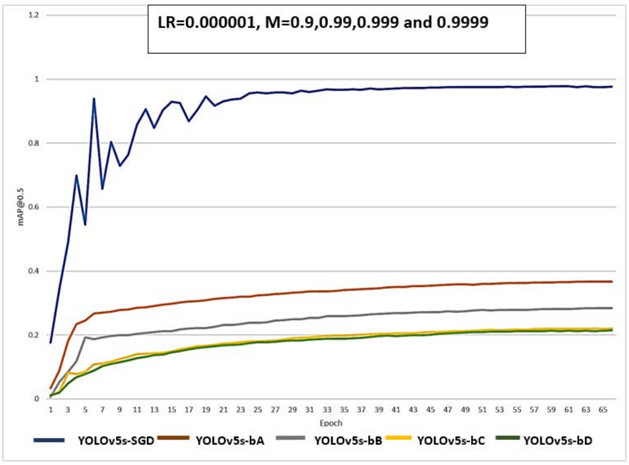


FIGURE 11. Comparison of SGD and ADAM optimizers in the YOLOv5s model (learning rate = 0.000001 and momentums = 0.9, 0.99, 0.999, and 0.9999).

smooth convergence speed, leading to a lower performance during the testing. Hence, it was a poor generalization model. Meanwhile, YOLOv5s-aA and YOLOv5s-bA, with a mAP value of 97.7% and 97.6%, respectively, had nearly similar performance as YOLOv5s-SGD. These two models were replicated with the best performance during the training at an optimum learning rate of 0.001 and 0.0001, respectively, and a momentum of 0.9. In comparison, when trained with a learning rate of 0.000001 and momentum of 0.9999, YOLOv5s-dD yielded a 21.5% mAP only, with the worst performance through the never-seen dataset.

TABLE 6. The performance of testing using different combinations of learning rates and momentums on YOLOv5s models.

YOLOv5s Model	Optimizer	Learning Rate	Momentum	mAP@0.5%
YOLOv5s-SGD	SGD	0.01	0.937	97.7
YOLOv5s-zZ	ADAM	0.01	0.937	91.5
YOLOv5s-aA	ADAM	0.001	0.9	97.7
YOLOv5s-aB	ADAM	0.001	0.99	96.9
YOLOv5s-aC	ADAM	0.001	0.999	92.5
YOLOv5s-aD	ADAM	0.001	0.9999	80.4
YOLOv5s-bA	ADAM	0.0001	0.9	97.6
YOLOv5s-bB	ADAM	0.0001	0.99	96.9
YOLOv5s-bC	ADAM	0.0001	0.999	85.5
YOLOv5s-bD	ADAM	0.0001	0.9999	74.1
YOLOv5s-cA	ADAM	0.00001	0.9	75.6
YOLOv5s-cB	ADAM	0.00001	0.99	69.0
YOLOv5s-cC	ADAM	0.00001	0.999	54.0
YOLOv5s-cD	ADAM	0.00001	0.9999	39.7
YOLOv5s-dA	ADAM	0.000001	0.9	35.7
YOLOv5s-dB	ADAM	0.000001	0.99	28.4
YOLOv5s-dC	ADAM	0.000001	0.999	21.6
YOLOv5s-dD	ADAM	0.000001	0.9999	21.5

Overall, YOLOv5s-bA showed an outstanding performance in the training of mAP together with a smoother training curve and faster convergence than other ADAM-based models or even the default YOLOv5s. A smoother training result was probably due to the ADAM algorithm's capability in adapting the gradient descent after each iteration, allowing it to remain in control and unbiased throughout the training. Consequently, it could efficiently process huge input data, such as detecting underwater animals that required a large image sample for model development.

Fig. 12 shows the effect of the learning rate on the testing of mAP with a fixed momentum value. In general, the learning rate increased/reduced the mAP performances in YOLOv5s. Meanwhile, Fig.13 shows that when the momentum was set as 0.9, the testing yielded the best and optimum value for the ADAM-based YOLOv5s model. Also, at the same learning rate, an increased in momentum reduced the mAP performance. Each momentum value would change the steps taken to the minimum by enforcing previous updates into a current one. Such a change in the size of steps depended on how momentum worked in ADAM by adjusting the direction from the updates made for fast descent towards the minimum point [28], [47].

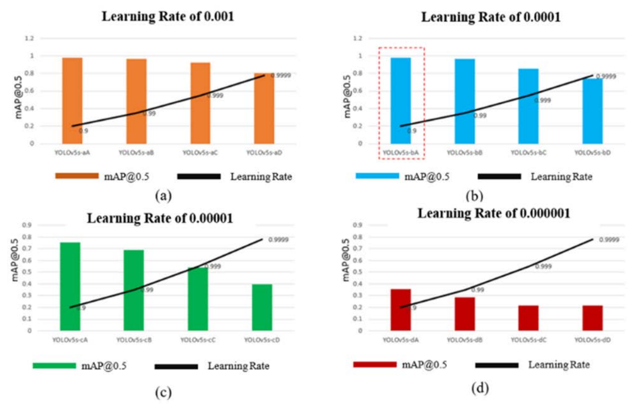


FIGURE 12. The effect of learning rate effect on the testing of mAP with the best model: (a) YOLOv5s-aA, YOLOv5s-bA, (b) YOLOv5s-aB, YOLOv5s-bB, (c) YOLOv5s-aC, YOLOv5s-bC, and (d) YOLOv5s-aD, YOLOv5s-bD.

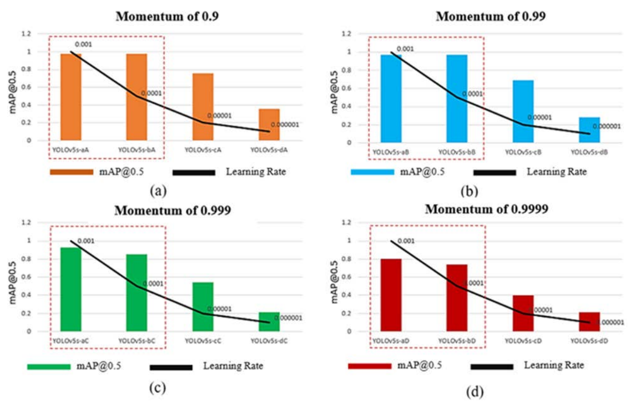


FIGURE 13. The effect of momentum on the testing of mAP with the best model: (a) YOLOv5s-aA, YOLOv5s-bA, (b) YOLOv5s-aB, YOLOv5s-bB, (c) YOLOv5s-aC, YOLOv5s-bC, and (d) YOLOv5s-dA, YOLOv5s-dB.

C. TRAINING EFFICIENCY BASED ON THE REDUCTION OF LEARNING RATE ON A PLATEAU

The YOLOv5s-bA model with a reduced learning rate on a plateau was designated as YOLOv5s-bA-LRP. Fig. 14 compares the training curves of YOLOv5s-SGD, YOLOv5s-bA, and YOLOv5s-bA-LRP. In early epochs of eight to 21, the YOLOv5s-bA-LRP curve showed better feature extraction performance than YOLOv5s-SGD and YOLOv5s-bA. For example, the performance of YOLOv5s-bA was 3.9% better than the two other models at the 9th epoch and 3.6% higher at the 17th epoch. Since better training performance was produced at earlier epochs, the speed of YOLOv5s-bA-LRP converged faster than YOLOv5s-bA. Then, from the beginning of the 18th epoch to the 26th epoch, the curve showed a plateauing behavior, triggering the function reduce-learning-rate-on-plateau to work on the YOLOv5s-bA-LRP model. Consequently, at the 26th epoch, the learning rate was reduced by a 0.1 ratio. Therefore, YOLOv5s-bA-LRP yielded the best performance with a 0.41% increment upon reaching the 27th epoch and increased further by 0.51% at the 28th epoch. The improvement was then set below the threshold for eight consecutive epochs, activating the function again at the 36th epoch 36. Hence, mAP at the 38th epoch increased by 0.35%. Subsequently, the auto-tuning of the function reducing-learning-rate-on-plateau was stimulated at the 53rd epoch as the YOLOv5s-bA-LRP model reached the 8th patience set due to its below-threshold improvement for mAP. Finally, Fig. 15 shows the performance of YOLOv5-SGD comprising curves of all classes, and the overall classes curve was calculated from the average of all mAP classes. YOLOv5s-bA-LRP improved from 0.9779 to 0.9844 at the 55th epoch, with a 0.69% improvement in mAP, indicating that by reducing the learning rate, the network took smaller steps to continue developing the learning progress.

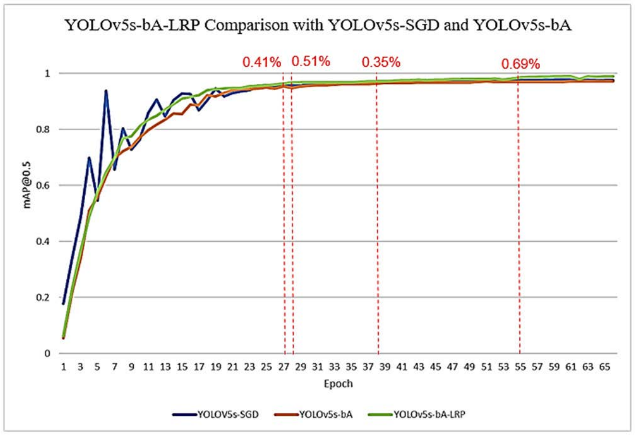


FIGURE 14. Comparison of YOLOv5s-bA-LRP, YOLOv5s-SGD, and YOLOv5s-bA.

Fig. 15 shows the performance of YOLOv5-SGD comprising curves of all classes in the PC curve, and the overall class curve was calculated from the average of all mAP classes.

Also, Fig. 16 shows a similar pattern in the PR performance of YOLOv5s-bA with a larger area under the curve, thus denoting an excellent detection by YOLOv5s-bA. Besides, this model yielded a mAP value similar to YOLOv5s-SGD

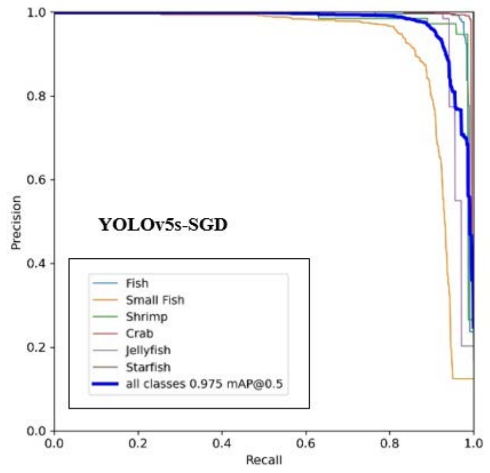


FIGURE 15. The precision-recall curve of YOLOv5s-SGD.

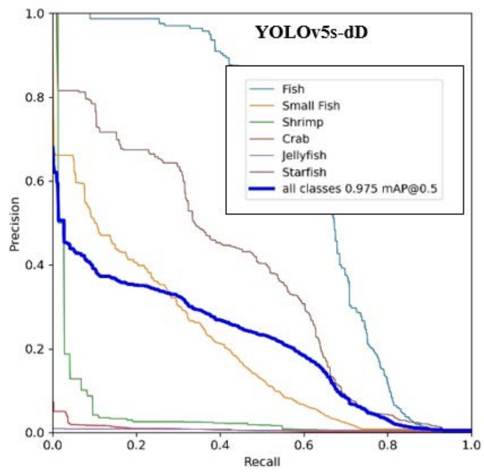


FIGURE 17. The precision-recall curve of YOLOv5s-dD.

with a slight increment of 0.001. This finding showed that YOLOv5s-bA could reach high precision and recall as YOLOv5s-SGD. However, Fig. 17 shows that among all tested models, YOLOv5s-dD had the lowest area under the curve in the PR curve, i.e., YOLOv5s-dD could achieve a satisfying recall value exceeding 0.8 but hardly achieved 0.7 in precision. In other words, although the YOLOv5s-dD model detected most of the positive samples correctly, it generated many FPs. Therefore, a low area under the curve led to a low mAP performance of 0.214 only for all classes.

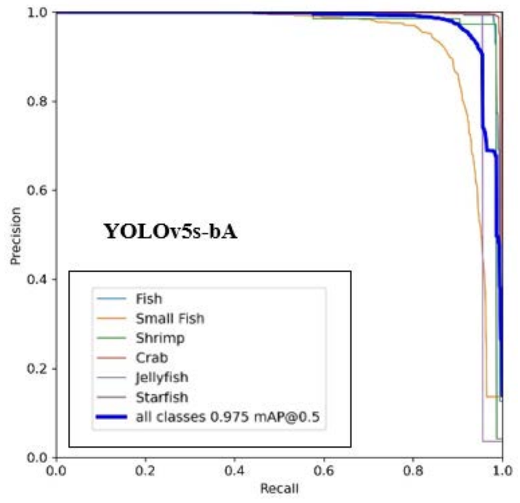


FIGURE 16. The precision-recall curve of YOLOv5s-bA.

D. PERFORMANCE OF THE OPTIMISED YOLOv5s-bA-LRP

Figure 18 compares the testing performance of mAP on the improved YOLOv5s-bA-LRP model, YOLOv5s-SGD, and YOLOv5s-bA. Overall, YOLOv5s-bA-LRP outperformed the other two models, attaining the highest mAP of 98.6%. Although the mAP value of YOLOv5s-bA-LRP was just 1% higher than YOLOv5s-bA and YOLOv5s-SGD, implementing the ADAM optimizer was suffix to stop the learning rate from reducing on a plateau, and hence, improved the model performance.

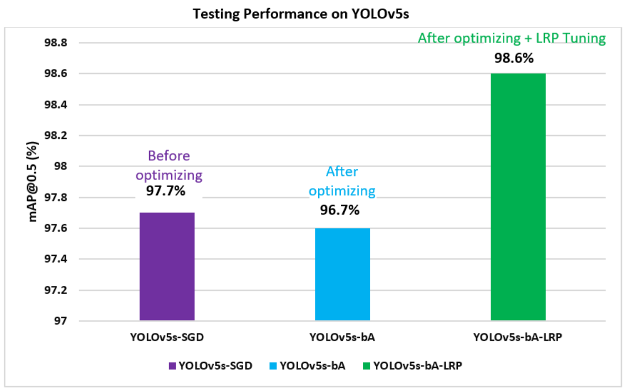


FIGURE 18. Comparison of testing performances among YOLOv5s-bA-LRP, YOLOv5s-SGD, and YOLOv5s-bA.

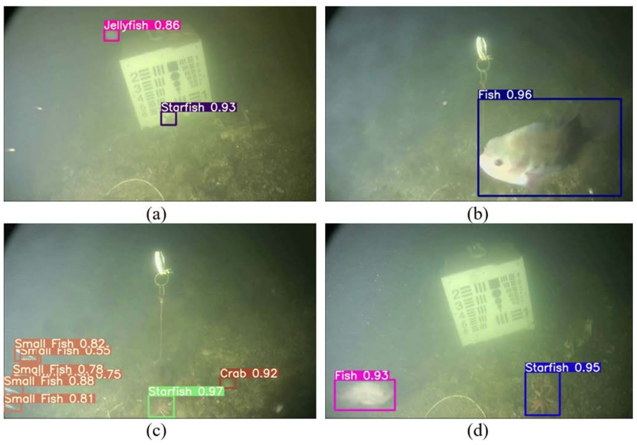


FIGURE 19. Output detection of the optimized YOLOv5s-bA-LRP model that could differentiate non-animal objects from (a) small animals, (b) large animals, (c) animals at the seafloor, and (d) blur animals.

Figure 19 shows the final validation of the optimised model via output detections from the test dataset, confirming that YOLOv5s-bA-LRP located bounding boxes in the challenging underwater environment. Specifically, the multi-scale detection further reinforced the efficiency of its head section and backbone competency in learning the feature complexity, i.e., differentiating the underwater animals and the background.

V. CONCLUSION

This study improved the performance of YOLO models by optimizing the tuning of learning rate and momentum in optimizer algorithm. Compared to all models, YOLOv5s produced the highest mAP at 97.7% with respective FPS of 106.4 that remarks outstanding model for detection underwater object in blur image. Furthermore, the improved model presented as YOLOv5s-bA consistently yielded a smooth training curve and faster convergence throughout the training phase with optimized parameters of learning rate 0.0001 and momentum at 0.9 producing the highest mAP at 97.6%. However, the superiority of the improved model based on ADAM is inadequate since the default SGD optimizer (learning rate of 0.01 and momentum at 0.937) is closely produced mAP at 97.7%. Thus, implementing the reduce-learning-rate-on-plateau function into the improved YOLOv5s (namely as YOLOv5s-bA-LRP) facilitated the tuning of the learning rate for the model to descend into areas of lower losses. YOLOv5s-bA-LRP improved to 98.6% mAP at the 55th epoch, indicating that by scaling down the learning rate over time, YOLOv5s yielded better convergence, producing a high-performance rate for underwater detection. Since the learning rate setting is not a one-size-fits-all parameter, the practicality of reducing the learning rate scheme by cutting the step size by a constant ratio in the absence of progress would become difficult in building a deep learning model. Overall, implementing hyper-parameter tuning into the YOLOv5s optimizer and reducing the learning rate on a plateau enhanced the training for optimizing the underwater detection model more effectively.

ACKNOWLEDGMENT

Members of the Advanced Control System and Computing Research Group, Advanced Rehabilitation Engineering in Diagnostic and Monitoring Research Group, and Center of Electrical Engineering, College of Engineering of Universiti Teknologi MARA (UiTM), Cawangan P. Pinang provided assistance and guidance for the field works. The authors are grateful to the Research Management Institute and UiTM, Cawangan Pulau Pinang for administrative support.

REFERENCES

[1] L. Kechiche, L. Touil, and B. Ouni, "Real-time image and video processing: Method and architecture," in *Proc. 2nd Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, Mar. 2016, pp. 194–199, doi: 10.1109/ATSIP.2016.7523067.

[2] A. Salman, A. Jalal, F. Shafait, A. Mian, M. Shortis, J. Seager, and E. Harvey, "Fish species classification in unconstrained underwater environments based on deep learning," *Limnol. Oceanogr., Methods*, vol. 14, no. 9, pp. 570–585, Sep. 2016, doi: 10.1002/lom3.10113.

[3] S. A. Siddiqui, A. Salman, M. I. Malik, F. Shafait, A. Mian, M. R. Shortis, and E. S. Harvey, "Automatic fish species classification in underwater videos: Exploiting pre-trained deep neural network models to compensate for limited labelled data," *ICES J. Mar. Sci.*, vol. 75, no. 1, pp. 374–389, Jan./Feb. 2017, doi: 10.1093/icesjms/fsx109.

[4] A. Salman, S. A. Siddiqui, F. Shafait, A. Mian, M. R. Shortis, K. Khurshid, A. Ulges, and U. Schwanecke, "Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system," *ICES J. Mar. Sci.*, vol. 77, no. 4, pp. 1295–1307, Jul. 2020, doi: 10.1093/icesjms/fsz025.

[5] B. W. McLaren, T. J. Langlois, E. S. Harvey, H. Shortland-Jones, and R. Stevens, "A small no-take marine sanctuary provides consistent protection for small-bodied by-catch species, but not for large-bodied, high-risk species," *J. Exp. Mar. Biol. Ecol.*, vol. 471, pp. 153–163, Oct. 2015, doi: 10.1016/j.jembe.2015.06.002.

[6] S. Kong, X. Fang, X. Chen, Z. Wu, and J. Yu, "A real-time underwater robotic visual tracking strategy based on image restoration and kernelized correlation filters," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2018, pp. 6436–6441, doi: 10.1109/CCDC.2018.8408261.

[7] W. Xu and S. Matzner, "Underwater fish detection using deep learning for water power applications," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2018, pp. 313–318, doi: 10.1109/CSCI46756.2018.00067.

[8] J. Marshall, "Vision and lack of vision in the ocean," *Current Biol.*, vol. 27, no. 11, pp. R494–R502, Jun. 2017, doi: 10.1016/j.cub.2017.03.012.

[9] D. L. Rizzini, F. Kallasi, F. Oleari, and S. Caselli, "Investigation of vision-based underwater object detection with multiple datasets," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 6, p. 77, Jun. 2015, doi: 10.5772/60526.

[10] F. Han, J. Yao, H. Zhu, and C. Wang, "Marine organism detection and classification from underwater vision based on the deep CNN method," *Math. Problems Eng.*, vol. 2020, pp. 1–11, Feb. 2020, doi: 10.1155/2020/3937580.

[11] M. Moniruzzaman, S. M. S. Islam, P. Lavery, and M. Bennamoun, "Faster R-CNN based deep learning for seagrass detection from underwater digital images," in *Proc. Digit. Image Comput., Techn. Appl. (DICTA)*, Dec. 2019, pp. 1–7, doi: 10.1109/DICTA47822.2019.8946048.

[12] M. Zhao, C. Hu, F. Wei, K. Wang, C. Wang, and Y. Jiang, "Real-time underwater image recognition with FPGA embedded system for convolutional neural network," *Sensors*, vol. 19, no. 2, p. 350, Jan. 2019, doi: 10.3390/s19020350.

[13] H. Zhou, H. Huang, X. Yang, L. Zhang, and L. Qi, "Faster R-CNN for marine organism detection and recognition using data augmentation," in *Proc. Int. Conf. Video Image Process.*, Dec. 2017, pp. 56–62.

[14] A. Jalal, A. Salman, A. Mian, M. Shortis, and F. Shafait, "Fish detection and species classification in underwater environments using deep learning with temporal information," *Ecol. Informat.*, vol. 57, May 2020, Art. no. 101088, doi: 10.1016/j.ecoinf.2020.101088.

[15] F. Ahmad, L. Ning, and M. Tahir, "An improved D-CNN based on YOLOv3 for pedestrian detection," in *Proc. IEEE 4th Int. Conf. Signal Image Process. (ICSIP)*, Jul. 2019, pp. 405–409, doi: 10.1109/SIPROCESS.2019.8868839.

[16] A. M. Algorry, A. G. Garcia, and A. G. Wofmann, "Real-time object detection and classification of small and similar figures in image processing," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2017, pp. 516–519, doi: 10.1109/CSCI.2017.87.

[17] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster R-CNN and YOLOv3," in *Proc. 1st Int. Conf. Unmanned Vehicle Syst.-Oman (UVS)*, Feb. 2019, pp. 1–6, doi: 10.1109/UVS.2019.8658300.

[18] S. Liu, X. Li, M. Gao, Y. Cai, R. Nian, P. Li, T. Yan, and A. Lendasse, "Embedded online fish detection and tracking system via YOLOv3 and parallel correlation filter," in *Proc. OCEANS MTS/IEEE Charleston*, Oct. 2018, pp. 1–6, doi: 10.1109/OCEANS.2018.8604658.

[19] R. Huang, J. Gu, X. Sun, Y. Hou, and S. Uddin, "A rapid recognition method for electronic components based on the improved YOLO-V3 network," *Electronics*, vol. 8, no. 8, p. 825, Jul. 2019, doi: 10.3390/electronics8080825.

[20] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, S. Nazir, and A. U. Haq, "Object detection through modified YOLO neural network," *Sci. Program.*, vol. 2020, pp. 1–10, Jun. 2020, doi: 10.1155/2020/8403262.

[21] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, ''DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection,'' *Inf. Sci.*, vol. 522, pp. 241–258, Jun. 2020, doi: 10.1016/j.ins.2020.02.067.
[22] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, ''A target detection model based on improved tiny-YOLOv3 under the environment of mining truck,'' *IEEE Access*, vol. 7, pp. 123757–123764, 2019, doi: 10.1109/ACCESS.2019.2928603.
[23] X. Zhang, W. Yang, X. Tang, and J. Liu, ''A fast learning method for accurate and robust lane detection using two-stage feature extraction with YOLOv3,'' *Sensors*, vol. 18, no. 12, p. 4308, Dec. 2018, doi: 10.3390/s18124308.
[24] S. Zhang, Y. Wu, C. Men, and X. Li, ''Tiny YOLO optimization oriented bus passenger object detection,'' *Chin. J. Electron.*, vol. 29, no. 1, pp. 132–138, Jan. 2020, doi: 10.1049/cje.2019.11.002.
[25] X. Zhang, W. Wang, Y. Zhao, and H. Xie, ''An improved YOLOv3 model based on skipping connections and spatial pyramid pooling,'' *Syst. Sci. Control Eng.*, vol. 9, pp. 142–149, Apr. 2021, doi: 10.1080/21642583.2020.1824132.
[26] T. Zeng, J. Wang, B. Cui, X. Wang, D. Wang, and Y. Zhang, ''The equipment detection and localization of large-scale construction jobsite by far-field construction surveillance video based on improving YOLOv3 and grey wolf optimizer improving extreme learning machine,'' *Construct. Building Mater.*, vol. 291, Jul. 2021, Art. no. 123268, doi: 10.1016/j.conbuildmat.2021.123268.
[27] Z. Xu, H. Shi, N. Li, C. Xiang, and H. Zhou, ''Vehicle detection under UAV based on optimal dense Yolo method,'' in *Proc. 5th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2018, pp. 407–411, doi: 10.1109/ICSAI.2018.8599403.
[28] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, ''Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection,'' *Sustain. Cities Soc.*, vol. 65, Feb. 2021, Art. no. 102600, doi: 10.1016/j.scs.2020.102600.
[29] D. P. Kingma and J. L. Ba, ''Adam: A method for stochastic optimization,'' in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[30] C. Nunez. (2019). *Our Oceans are Under Attack by Climate Change, Overfishing, National Geographic*. Accessed: Dec. 2, 2020. [Online]. Available: https://www.nationalgeographic.com/environment/article/ocean
[31] J. Tang, S. Liu, B. Zheng, J. Zhang, B. Wang, and M. Yang, ''Smoking behavior detection based on improved YOLOv5s algorithm,'' in *Proc. 9th Int. Symp. Next Gener. Electron. (ISNE)*, Jul. 2021, pp. 1–4, doi: 10.1109/ISNE48910.2021.9493637.
[32] S. Li, Y. Li, Y. Li, M. Li, and X. Xu, ''YOLO-FIRI: Improved YOLOv5 for infrared image object detection,'' *IEEE Access*, vol. 9, pp. 141861–141875, 2021, doi: 10.1109/access.2021.3120870.
[33] J. Wang, T. Xiao, Q. Gu, and Q. Chen, ''YOLOv5_CSL_F: YOLOv5's loss improvement and attention mechanism application for remote sensing image object detection,'' in *Proc. Int. Conf. Wireless Commun. Smart Grid (ICWCSG)*, Aug. 2021, pp. 197–203, doi: 10.1109/ICWCSG53609.2021.00045.
[34] G. Jocher *et al.*, ''Ultralytics/YOLOv5: V3.1—Bug fixes and performance improvements,'' *Zenodo*, Oct. 2020, doi: 10.5281/zenodo.4154370.
[35] C.-Y. Wang, H.-Y. M. 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,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1571–1580.
[36] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, ''Path aggregation network for instance segmentation,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.
[37] Y. Bengio, ''Practical recommendations for gradient-based training of deep architectures,'' in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 437–478.
[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 2016.
[39] M. Zaheer, S. J. Reddi, D. Sachan, S. Kale, and S. Kumar, ''Adaptive methods for nonconvex optimization,'' in *Proc. Neural Inf. Process. Syst. (NIPS)*, no. 32, 2018, pp. 9793–9803.
[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, no. 1, pp. 1–9.
[41] Jahandad, S. M. Sam, K. Kamardin, N. N. A. Sjarif, and N. Mohamed, ''Offline signature verification using deep learning convolutional neural network (CNN) architectures GoogLeNet inception-v1 and inception-v3,'' *Proc. Comput. Sci.*, vol. 161, pp. 475–483, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919318587#!
[42] R. Padilla, S. L. Netto, and E. A. B. da Silva, ''A survey on performance metrics for object-detection algorithms,'' in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 237–242.
[43] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, S. Nazir, and A. U. Haq, ''Object detection through modified YOLO neural network,'' *Sci. Program.*, vol. 2020, pp. 1–10, Jun. 2020, doi: 10.1155/2020/8403262.
[44] L. N. Smith, ''A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay,'' 2018, *arXiv:1803.09820*.
[45] M. S. Asyraf, I. S. Isa, M. I. F. Marzuki, S. N. Sulaiman, and C. C. Hung, ''CNN-based YOLOv3 comparison for underwater object detection,'' *J. Electr. Electron. Syst. Res.*, vol. 18, pp. 30–37, Apr. 2021, doi: 10.24191/jeesr.v18i1.005.
[46] D. P. Kingma and J. L. Ba, ''Adam: A method for stochastic optimization,'' in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[47] K. Mukherjee, A. Khare, and A. Verma, ''A simple dynamic learning rate tuning algorithm for automated training of DNNs,'' 2019, *arXiv:1910.11605*.
[48] O. Almutiry, K. Iqbal, S. Hussain, A. Mahmood, and H. Dhahri, ''Underwater images contrast enhancement and its challenges: A survey,'' *Multimedia Tools Appl.*, pp. 1–26, Feb. 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11042-021-10626-4#citeas
[49] C. Akila and R. Varatharajan, ''Color fidelity and visibility enhancement of underwater image de-hazing by enhanced fuzzy intensification operator,'' *Multimedia Tools Appl.*, vol. 77, no. 4, pp. 4309–4322, Feb. 2018.
[50] S. Anwar and C. Li, ''Diving deeper into underwater image enhancement: A survey,'' *Signal Process., Image Commun.*, vol. 89, Nov. 2020, Art. no. 115978.
[51] J. Banerjee, R. Ray, S. R. K. Vadali, S. N. Shome, and S. Nandy, ''Real-time underwater image enhancement: An improved approach for imaging with AUV-150,'' *Sadhana*, vol. 41, no. 2, pp. 225–238, Feb. 2016.
[52] F. Lei, F. Tang, and S. Li, ''Underwater target detection algorithm based on improved YOLOv5,'' *J. Mar. Sci. Eng.*, vol. 10, no. 3, p. 310, Feb. 2022.
[53] H. Zhang, M. Tian, G. Shao, J. Cheng, and J. Liu, ''Target detection of forward-looking sonar image based on improved YOLOv5,'' *IEEE Access*, vol. 10, pp. 18023–18034, 2022.
[54] G. Huang, ''A comparative study of underwater marine products detection based on YOLOv5 and underwater image enhancement,'' *Int. Core J. Eng.*, vol. 7, no. 5, pp. 213–221, 2021.
[55] K. M. Awan, P. A. Shah, K. Iqbal, S. Gillani, W. Ahmad, and Y. Nam, ''Underwater wireless sensor networks: A review of recent issues and challenges,'' *Wireless Commun. Mobile Comput.*, vol. 2019, Jan. 2019, Art. no. 6470359, doi: 10.1155/2019/6470359.
[56] Y. Shen, C. Zhao, Y. Liu, S. Wang, and F. Huang, ''Underwater optical imaging: Key technologies and applications review,'' *IEEE Access*, vol. 9, pp. 85500–85514, 2021, doi: 10.1109/ACCESS.2021.3086820.
[57] R. S. Nair, R. Agrawal, S. Domnic, and A. Kumar, ''Image mining applications for underwater environment management—A review and research agenda,'' *Int. J. Inf. Manage. Data Insights*, vol. 1, no. 2, Nov. 2021, Art. no. 100023, doi: 10.1016/j.jjimei.2021.100023.
[58] R. S. Nair and S. Domnic, ''A combination of learning and non-learning based method for enhancement, compression and reconstruction of underwater images,'' *Aquaculture Fisheries*, vol. 7, no. 2, pp. 201–210, Mar. 2022, doi: 10.1016/j.aaf.2021.10.006.
[59] S. Xu, J. Zhang, X. Qin, Y. Xiao, J. Qian, L. Bo, H. Zhang, H. Li, and Z. Zhong, ''Deep retinex decomposition network for underwater image enhancement,'' *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107822, doi: 10.1016/j.compeleceng.2022.107822.
[60] H. Wang, S. Zhang, S. Zhao, Q. Wang, D. Li, and R. Zhao, ''Real-time detection and tracking of fish abnormal behavior based on improved YOLOV5 and SiamRPN++,'' *Comput. Electron. Agricult.*, vol. 192, Jan. 2022, Art. no. 106512, doi: 10.1016/j.compag.2021.106512.
[61] Z. Wang, P. Xu, B. Liu, Y. Cao, Z. Liu, and Z. Liu, ''Hyperspectral imaging for underwater object detection,'' *Sensor Rev.*, vol. 41, no. 2, pp. 176–191, May 2021, doi: 10.1108/SR-07-2020-0165.
[62] H. Huang, Z. Sun, S. Liu, Y. Di, J. Xu, C. Liu, R. Xu, H. Song, S. Zhan, and J. Wu, ''Underwater hyperspectral imaging for *in situ* underwater microplastic detection,'' *Sci. Total Environ.*, vol. 776, Jul. 2021, Art. no. 145960, doi: 10.1016/j.scitotenv.2021.145960.
[63] H. Chen, J. Lin, L. Zhuge, and X. Xia, ''Underwater image restoration and target detection based on monocular depth estimation,'' in *Proc. China Autom. Congr. (CAC)*, Oct. 2021, pp. 5597–5601.
[64] X. Li, X. Xia, Z. Hu, B. Han, and Y. Zhao, ''Intelligent detection of underwater fish speed characteristics based on deep learning,'' in *Proc. 5th Asian Conf. Artif. Intell. Technol. (ACAIT)*, Oct. 2021, pp. 182–189, doi: 10.1109/acait53529.2021.9731159.

IZA SAZANITA ISA (Member, IEEE) received the bachelor's degree in electrical engineering from Universiti Teknologi MARA (UiTM), Penang Campus, Malaysia, in 2004, the M.Sc. degree from Universiti Sains Malaysia (USM), Malaysia, in 2008, and the Ph.D. degree in electrical engineering, in 2018, under the SLAB/SLAI Scholarship.

Since 2009, she has been an Young Lecturer with UiTM and promoted as a Senior Lecturer with the School of Electrical Engineering, College of Engineering, UiTM, in 2013. She is currently a Postdoctoral Fellow with the School of Computer Sciences, USM. She is attached to the Department of Control System Engineering at the faculty. She is also a member of AREDiM Research Group, the Head of the Research Group RIDyLT, and actively involved in teaching and learning research. Her research interests include the model development using image processing and artificial intelligence.

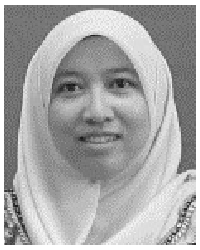
MOHAMED SYAZWAN ASYRAF ROSLI was born in Kelantan, Malaysia, in 1996. He received the B.Sc. degree in electrical engineering from Universiti Teknologi MARA (UiTM), Penang Branch, Malaysia, in 2019, where he is currently pursuing the master's degree in science by fully research mode with the School of Electrical Engineering, College of Engineering, under the Graduate Research Assistant (GRA) FRGS Grant Scheme 291/2019.

He was awarded with the Vice Chancellor Award from UiTM.

UMI KALSOM YUSOF received the B.Sc. degree from Western Illinois University, Macomb, IL, USA, the M.Sc. degree from Universiti Sains Malaysia (USM), Penang, and the Ph.D. degree in computer science from Universiti Teknologi Malaysia (UTM), Skudai, Johor.

She is currently an Associate Professor and a Lecturer with the School of Computer Sciences, USM. She has published research articles at national and international journals, conference proceedings, as well as chapters of books. Her research interests include data mining, web engineering, computational intelligence, artificial intelligence, multiobjective optimization, evolutionary computing, computer security, and grid computing.

MOHD IKMAL FITRI MARUZUKI (Member, IEEE) received the B.Eng. degree in computer engineering from Ehime University, Japan, in 2004, and the master's degree in communication and computer engineering from Universiti Kebangsaan Malaysia (UKM), in 2010. He is currently employed as a Lecturer at the School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA (UiTM), Penang Branch, Malaysia. His research interests include machine learning, image processing, and embedded systems.



SITI NORAINI SULAIMAN (Senior Member, IEEE) received the B.Eng. degree (Hons.) in electrical and electronics and the M.Sc. degree in electrical and electronics engineering (medical imaging) from Universiti Sains Malaysia, in 2000 and 2003, respectively, and the Ph.D. degree in imaging from the School of Electrical and Electronics Engineering, Universiti Sains Malaysia, in 2012.

She joined the Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM), Penang Branch, Penang, Malaysia, as a Contract Lecturer, in 2002. She is currently an Associate Professor heading the Advanced Rehabilitation Engineering in Diagnostic and Monitoring Research Group (AREDiM), School of Electrical Engineering, College of Engineering, UiTM. She has published numerous research articles in international journals and conference proceedings. Her research interests include intelligent systems, image processing, neural networks for medical applications, and algorithms.