



# Yolov4-tiny with wing convolution layer for detecting fish body part

Eko Prasetyo<sup>a, b</sup>, Nanik Suciati<sup>a, \*</sup>, Chastine Fatichah<sup>a</sup>

<sup>a</sup> Department of Informatics, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember, Jl. Raya ITS, Surabaya 60111, Indonesia

<sup>b</sup> Department of Informatics, Faculty of Engineering, University of Bhayangkara Surabaya, Jl. Ahmad Yani 114, Surabaya 60231, Indonesia

## ARTICLE INFO

### Keywords:

Yolov4-tiny  
Wing convolutional layer  
Head  
Tail and fish detection  
Spatial pyramid pooling  
Scale detector

## ABSTRACT

Detection of a fish's eye, tail and body is the initial process in the vision system for determining the freshness and species of fish, as well as calculating the number of fish automatically in the fishing industry. Classification performance of a system is affected by its ability to detect the intact body of a fish or its body parts. The You Only Look Once version 4 tiny (Yolov4-tiny) is a lightweight object detector that can detect body parts of a fish with fairly good detection accuracy. However, massive siltation of convolution layer in the Yolov4-tiny backbone leads to low feature diversity. This research proposes a modification of the Yolov4-tiny architecture to improve detection accuracy by enhancing and balancing feature diversity and attaching an extra-branch detector to detect small-sized objects. In addition, we propose the use of bottleneck and expansion convolution to reduce computational resources usage. Our contributions are enhancing feature diversity using a wing convolution layer (WCL), balancing feature diversity using tiny spatial pyramid pooling (Tiny-SPP), reducing computational resources of feature pyramid network (FPN) connections using bottleneck and expansion convolution (BEC), and detecting small objects using an extra-branch as a third-scale detector. Our experimental results show that the proposed model outperforms the original model and other modified Yolov4-tiny models with Precision, Recall, AP, and mAP of 97.48%, 93.3%, 94.07%, and 92.38% respectively. The proposed model is smaller in size and more efficient in the use of computing resources.

## 1. Introduction

Fish is a highly nutritious dish (Jose et al., 2021), and contains proteins, vitamins, and minerals required for human health (Erasmus et al., 2021; Prabhakar et al., 2020). The affordable price and freshness of fish motivate people to choose fish as a dish (Mitra et al., 2021). The fresher the fish, the better the nutritional content. However, recognizing the freshness of a fish in a market is a quite difficult for most people. The easiest way to check the freshness of the fish is by touching and pressing the body to measure its elasticity. Fresh fish is generally more elastic. However, this technique causes bacterial contamination that can damage the fish and occur food-borne diseases (Hashanuzzaman et al., 2020). A computer vision application that can classify fish freshness by analyzing the visual appearance of fish body parts is a touchless alternative solution to avoid bacterial contamination (Lalabadi et al., 2020).

In the commercial fishing industry, estimation of the species and quality of fish are used to determine the appropriate breeding system. In addition, the number of fish is also an important issue in maintaining the condition of fish and environmental quality in the fish breeding

industry. Counting the number of fish manually can harm fish growth, time-consuming, labor-intensive and has a high error probability. The adoption of an automated computer vision system can be of great help to the commercial fishing industry. In such vision system, the detection of fish body parts is an important initial process before further analysis such as classification of fish species and quality and the calculation of the number of fish can be carried out (Cai et al., 2020; N.S. et al., 2021).

The detection of fish body parts has been conducted in several studies, for example, segmentation of fish gill and eye based on color change in several color spaces (Kunjulakshmi et al., 2020), segmenting eye and gill based on color degradation of fish eye and gill in several color spaces (Lalabadi et al., 2020; Sengar et al., 2018), segmenting fish gill using clustering (Dutta et al., 2016), and segmenting fish gill using several strategies of image processing (Issac et al., 2017). (Issac et al., 2017). Furthermore, several segmentation methods to separate intact fish from complex backgrounds have been proposed, such as using blob analysis (Prados et al., 2017), combination of multiscale deformable (Nian et al., 2017), and Hough circle detection (C. Yu et al., 2020). These approach are commonly performed by analyzing the intensity, color,

\* Corresponding author.

E-mail address: [nanik@if.its.ac.id](mailto:nanik@if.its.ac.id) (N. Suciati).

<https://doi.org/10.1016/j.compag.2022.107023>

Received 25 October 2021; Received in revised form 28 April 2022; Accepted 30 April 2022

Available online 12 May 2022

0168-1699/© 2022 Elsevier B.V. All rights reserved.

and shape of the fish image; which should take into consideration the variations in background, color, and lighting.

Meanwhile, several studies have explored deep learning approaches to detect fish and reported better performance than conventional approaches in environments with complex backgrounds (Sung et al., 2017), cloudy water conditions (Christensen et al., 2018), and in the case of detecting dead fish on the water surface (G. Yu et al., 2020). Performance improvements in deep learning approaches are generally achieved by modifying the architecture, combining the architecture with conventional methods and improving data quality. Such improvements include replacing the backbone of Single Shot MultiBox Detector (SSD) using MobileNet instead of VGG-16 (G. Yu et al., 2020), replacing the backbone of You Only Look Once version 3 (Yolov3) using MobileNet (Cai et al., 2020), enhancing the deep learning framework (Alshdaifat et al., 2020), combining Yolo and Gaussian mixture models (Jalal et al., 2020), adding a negative class (Christensen et al., 2018), and increasing the variety of feature maps generated by the Yolov4-tiny backbone using spatial pyramid pooling (SPP) (Prasetyo et al., 2021). Yolov4-tiny can trim 91% of the standard Yolov4 model size by compressing 23 CSPDarknet layers backbone into three CSPDarknet layers and using only two scale detectors (Bochkovskiy et al., 2020) with fairly good detection accuracy. In addition, this model has a 2.54% increase in mAP in the problem of face-mask detection by adding six convolution layers in the backbone (Kumar et al., 2021). Modifying the loss function and layer model can also increase recall by 98.8% in badminton shuttlecock detection (Cao et al., 2021). However, massive siltation of the convolution layer in the Yolov4-tiny backbone leads to low feature diversity. Consequently, the model is poor in detecting small objects such as fishtails as can be seen in the low recall. In this research, we propose a modification of Yolov4-tiny to improve detection accuracy of fish body part and to reduce its computational resource usage. Our contributions are as follows:

- Enhancing feature diversity using Wing Convolutional Layer (WCL)

Yolov4-tiny is a fast object detector with small-sized model and limited performance. It is caused by the lack of feature diversity by shallow CSPDarknet. We address such a problem by enhancing the variety of the features using WCL, while by separating and paralleling convolutional layers to the main backbone. The end feature map of WCL is combined with the feature map generated by the main backbones. This way improves the detection of intact fish, fish heads, and fishtails.

- Balancing the diversity of features using Tiny-SPP

The increase in feature diversity by SPP induces exaggerated features as well. We address this problem by reducing the kernel pooling to balance the diversity of features.

- Attaching an extra branch detector to detect fishtails

The first- and second-scale detectors of Yolov4-tiny are sufficient to detect intact fish and fish heads. However, it is insufficient to detect fishtails because of the small size. We attach a third-scale detector using a feature pyramid network (FPN) by upsampling the second scale's feature map which is then combined with the feature map from the previous layer.

- Reducing computational resources of feature pyramid network (FPN)

Since the third version, Yolo has used FPN to facilitate multi-scale detectors by concatenating current and previous feature maps and standard convolution. This method causes excessive computational resources. We propose a bottleneck and expansion convolution (BEC) using a bottleneck and expansion convolution sequentially. BEC convolutes the feature maps concatenation using a lower cost than the

standard convolution.

- Modification of Yolov4-tiny using WCL, Tiny-SPP, BEC, and an additional third-scale detector

We propose enhancing the Yolov4-tiny model by modifying the backbone using WCL, adding Tiny-SPP, replacing the FPN connection convolution method using BEC, and attaching a third-scale detector. WCL enhances the diversity of features from backbones. Tiny-SPP intention balances and prevents the expansion and excessive diversity of features. BEC collects relevant features with a low computational cost, and the third-scale detector increases the ability to detect small objects, particularly fishtails. These enhancements can improve and optimize the performance of the Yolov4-tiny model in detecting intact fish, fish heads, and fishtails.

The dataset used in this study is called Fish and Fish Part Detection (FFPD). It was developed from the dataset (200 images and two object classes) used in the research conducted by Prasetyo et al. (2021) and added with 400 images of three object classes, namely heads, tails, and intact fish body. Furthermore, the dataset was used to evaluate the performance of every module in improving the detection performance. The performance of our proposed model was compared to the performance of other Yolo-based models, namely the original Yolov4-tiny, Yolov4-tiny + SPP, Yolov3-tiny, Yolov3, and Yolov4. The experimental results show that our proposed model outperformed all the other models and achieved the best Precision and Recall.

The remainder of this paper is arranged as follows. The related work utilized in the research is discussed in Section 2. Section 3 describes the details of our proposed model. In Section 4, we show and discuss the experimental results. Finally, Section 5 presents the conclusion of this study and possible future works.

## 2. Related work

The detection and localization of fish or its body parts are necessary to support the judgment of fish freshness. Several studies addressed this issue using a clustering approach (Dutta et al., 2016), color features (Lalabadi et al., 2020), multiscale deformable of the object (Nian et al., 2017), and Gaussian mixture modeling (Salman et al., 2019). These approaches are conventional, and highly dependent on several constraints, such as color, lighting, and background; thus, the effort is more profound than using a deep learning approach. In contrast, deep learning is becoming a popular approach for detecting objects due to not being influenced by the constraints. Region-based models such as Faster-RCNN (Ren et al., 2017), Mask-RCNN (He et al., 2017), and some improvements using network distillation (Peng et al., 2020), saliency guiding (Sharma and Mir, 2019), and shape regression (Haas et al., 2020) are a few examples of recently proposed deep learning approaches. A common vulnerability faced by RCNN-based models is low-speed detection due to a two-stage detection. The other approach uses one-stage detection with only one regression process, such as Single Shot MultiBox Detector (Liu et al., 2016) and You Only Look Once (Yolo) (Redmon et al., 2016). With only one regression process, this approach obtains fast speed in detection.

Yolo is a model used to detect the object in various image conditions such as color, lighting, complicated backgrounds, and multiple objects. Yolo reaches a more improved model on per updated version, such as Yolo version 2 (Yolov2) (Redmon and Farhadi, 2017), Yolov3 (Redmon and Farhadi, 2018), and Yolov4 (Bochkovskiy et al., 2020). Several researchers carried out further enhancements on Yolo to improve its performance, such as strengthening the feature maps using dense connections (Huang et al., 2020), using random kernel and double hidden layer as the backbone (Yin et al., 2020), replacing the backbone using Resnet (Loey et al., 2021), using the third-scale and activating the fifth-scale detector to detect extremely small objects (Hu et al., 2021), and replacing the backbone with the architecture of MobileNet (G. Yu et al.,

2020). The combination of Yolo with other models can also solve special problems, such as using negative classes to prevent detecting non-fish objects (Sung et al., 2017), using gaussian mixture models to detect and count fish objects (Jalal et al., 2020), and using a classifier to detect and monitor the vehicle in traffic flow (Azimjonov and Özmen, 2021).

From the explanation above, it can be concluded that detecting fish or its body parts using a deep learning approach promises a fast detection model, robust background variations, and small size. The detection of intact fish, fish heads, and fishtails promotes a problem since few tail targets are detected due to their small size. Therefore, in this study, we modified several parts of Yolov4-tiny, such as the backbone, the part after the backbone, FPN connections, and the detector.

### 3. Proposed methods

#### 3.1. Wing convolution layer

Yolov4-tiny model has a small size and fast speed in detection. Therefore, it is highly applicable to mobile systems with limited storage and computational resources. This small model is represented by a shallow convolution layer backbone, in which it utilizes only three CSPDarknet and a few convolution layers. In addition, Yolov4-tiny uses two-scale detectors for detecting large and medium-sized objects. Due to this simple architecture, Yolov4-tiny is extremely fast in detecting objects. However, it has a low detection performance. The principal problem is the lack of feature diversity generated by the backbone, where the shallow convolution layer is insufficient to generate high-level features. SPP expands the diversity of features (Prasetyo et al., 2021) but may also cause excessive diversity of features that detect non-heads and non-tails.

Increasing the diversity of features is not continually accomplished in layer after the backbone. Instead, it can be carried out by creating a separating and paralleling branch block to the main backbone, called Wing Convolution Layer (WCL). WCL can be designed with arbitrarily deeper layers or other deeper architectures as a convolution branch. We call it wing because WCL generates other features that the main convolutional layer backbone may not have. The feature map generated by the final layer of WCL is combined with the feature map generated by the main backbone. However, adding WCL increases the model size. WCL supports the expansion of required features but increases the usage of computational resources. This is the trade-off that needs to be taken into consideration.

Suppose WCL performs convolution on several layers, the formulas are as follows:

$$F_i = K_i \times I \quad (1)$$

$$F_{wcl} = \sum_{i=1}^N K_i \times I \quad (2)$$

where  $I$  is the convolved image,  $K_i$  is the convolution kernel, and  $F_i$  is feature map at the  $i$ -th layer. WCL uses the feature map of the first layer, then substitutes  $I$  with the feature map of the first layer.  $F_{wcl}$  is the WCL feature map obtained after  $N$  convolution layers.

WCL reuses Darknet53 using convolution repetitions of 1, 2, 8, and 8 times for each block. Regarding the trade-off, the model size and computational resources required need to be taken into consideration. Therefore, we use a depthwise separable convolution (Howard et al., 2017). A depthwise separable convolution (DSC) is a convolution method that consists of depthwise convolution (DC) and pointwise convolution (PC). DC convolves the feature map using a single filter on each channel of the feature map, while PC is the same as the standard convolution where it convolves the feature map using a  $1 \times 1$  filter. The DSC concept achieves similar performances to standard convolution but with a significantly lower number of parameters and model size. Therefore, we designed WCL using DSC and Darknet53 as the main

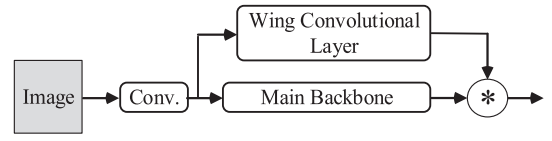


Fig. 1. Wing Convolution Layer.

module and architecture. We also use max-pooling to down-sample from one block to another. We concatenate both the resulting feature maps at the end of the WCL layer and the main backbone then forward it to the next stage. The formula to combine the WCL feature map ( $F_{wcl}$ ) and the main backbone feature map ( $F_{mb}$ ) is as follows: Fig. 1.

$$F_{bb} = F_{wcl} * F_{mb} \quad (3)$$

WCL is designed to generate features that the main backbone may not provide. This method is essential in detecting intact fish, fish heads, and fishtail, in which the WCL generates complementary features for generated features by the main backbone. We employed Darknet53 using convolution repetitions and DSC to reduce the model size and computational cost, as shown in Fig. 2. The smallest module in WCL is CBL, consisting of a standard convolution, batch normalization, and leaky ReLU. Meanwhile, DCBL is a DSC module that consists of a depthwise convolution, a batch normalization, a leaky ReLU, and a CBL with a kernel size of  $1 \times 1$ . The WCL architecture consists of a CBL, several max-pooling, and several DCBL. DCBL is repeated 1, 2, 8, and 8 times, similar to Darknet53 except for the utilization of DSC as the convolution approach. There are 39 convolution layers in WCL called Mobile-Darknet-39.

Yolov4-tiny highlights the advantages of tiny model size. Instead, the insertion of WCL into the architecture enriches computational resources' utilization. We use Floating Point Operations Per Second (FLOPS) to measure the computational resources as follows:

$$FLOPS = 2 \times K^2 \times F_{in} \times F_{out} \times W \times H \quad (4)$$

where  $K$  is the kernel size,  $F_{in}$  is the feature map of the input layer,  $F_{out}$  is the feature maps of the output layer, and  $W$  and  $H$  are the feature map output resolutions. Yolov4-tiny requires 6.79 billion FLOPS (BFLOPS) of computational resources, while the WCL architecture only requires 0.2646 BFLOPS. As a result, WCL only accounts for roughly 3.9% of the total computational resources of Yolov4-tiny, which is a significantly lesser amount.

#### 3.2. Tiny spatial pyramid pooling (tiny-SPP)

The limited detection performance of Yolo is addressed using SPP by pooling with several kernel sizes and concatenating several multi-scale local region features (Huang et al., 2020). This technique improves the detection performance indicated by an increase in mAP. However, the matter of non-target objects being detected has not been evaluated. Researches on the detection of fish head and fishtail show that SPP increases the detection of target and non-target objects (Prasetyo et al., 2021). Therefore, SPP increases the diversity of features with an excessive diversity of features instead, indicated by more non-target objects being detected. Generally, kernel pooling is used as follows:

$$size_K = \lceil size_{fmap} / i \rceil \quad (4)$$

where  $i = 1, 2, 3$ ,  $size_{fmap}$  is the size of the feature map, and  $size_K$  is the kernel size. The results of pooling using the three kernels are combined with the original feature map. We maintain the ability of SPP to increase the diversity of features through pooling with several kernel sizes while also avoiding excessive feature diversity. To achieve this goal, we propose to reduce the over-pooling of SPP by pooling using only two kernels plus a non-pooling feature map. When  $i = 1, 2$  and  $size_{fmap} = 13$ , we obtain kernel sizes of  $13 \times 13$  and  $7 \times 7$  called Tiny-SPP. Reducing the

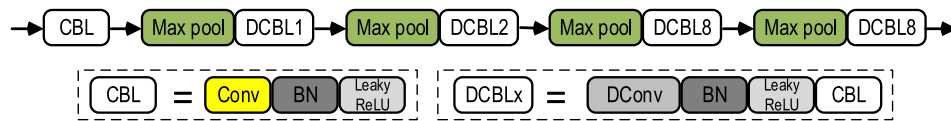


Fig. 2. WCL for the detection of intact fish, fish head, and fishtail.

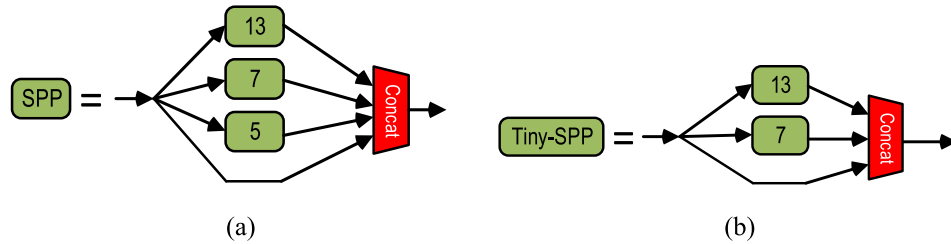


Fig. 3. Spatial Pyramid Pooling; (a) Original; (b) Tiny.

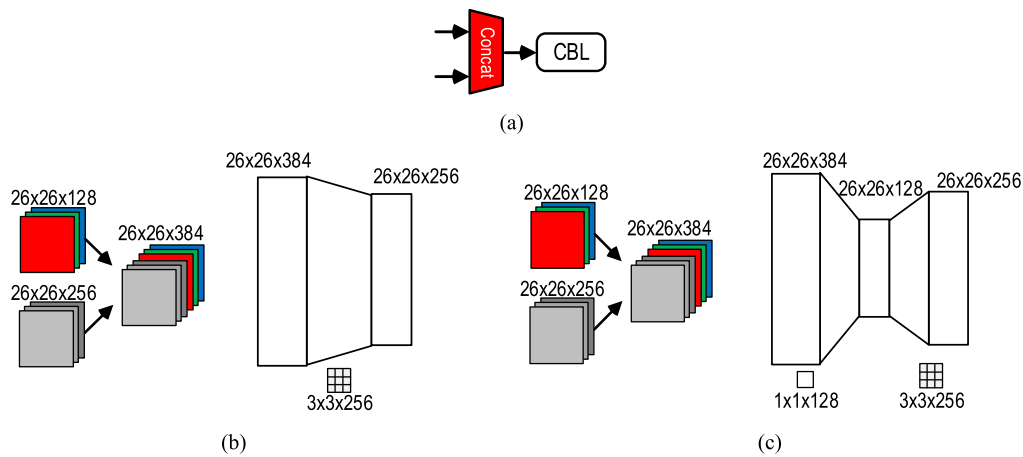


Fig. 4. FPN Connection; (a) Connecting two feature maps; (b) Standard convolution; (c) Bottleneck and Expansion Convolution.

number of pools does not hinder the expansion of feature diversity and avoids excessive diversity of features.

The difference between Tiny-SPP and SPP is the absence of kernel pooling 5, as shown in Fig. 3. In Fig. 3 (a), SPP uses a three-kernel pooling and a non-pooling feature map, where both are concatenated into a new feature map. On the other hand, Tiny-SPP uses only a two-kernel pooling and a non-pooling feature map combined to assemble a new feature map, as shown in Fig. 3 (b).

### 3.3. Feature map concatenation of FPN using bottleneck and expansion convolution

BEC is a convolution method that linearly squeezes the feature map, then re-expands it using linear convolution (Sandler et al., 2018). According to empirical investigations, bottleneck convolution squeezes the feature map to leave significant features on the low-dimensional feature map. Then, the feature map is restored to its intended size by using linear expansion convolution.

Since the third version, called Yolov3 (Redmon and Farhadi, 2018), Yolo detects objects using three detectors with different scales. Each detects objects in different scale sizes, namely large, medium, and small. The large, medium and small sizes are the first, second, and third scales. Features of each scale are furthermore connected to other ones using an FPN (Lin et al., 2017), as shown in Fig. 4(a). As an example, the second-scale detector is employed by combining up-sampled first-scale features ( $26 \times 26 \times 128$ ) and second-scale features within skip connection ( $26 \times 26 \times 256$ ). The result is a ( $26 \times 26 \times 384$ ) feature maps. Unfortunately,

Yolov4 utilizes standard convolution to combine the features from both sources. As a result, the concatenation of the features inflates the computational resources. Convoluting into  $26 \times 26 \times 256$  requires a total of 1.196 BFLOPS. We propose a BEC that consists of two-step convolution by convoluting to a smaller size of the feature map, then re-convoluting to the intended size, as shown in Fig. 4(b). The standard convolution utilizes a kernel of size  $3 \times 3$ , while BEC utilizes kernels of sizes  $1 \times 1$  and  $3 \times 3$  for bottleneck and expansion convolution, respectively. At the same time, the number of channels created by bottleneck convolution is 128.

Convoluting the  $26 \times 26 \times 384$  feature map into  $26 \times 26 \times 256$  using BEC requires computational volumes of 0.067 and 0.399 BFLOPS for bottleneck and expansion, respectively. The total computational volume required is 0.466 BFLOPS, only approximately 39% of the computational volume required by standard convolution.

### 3.4. Attaching an extra branch used as a third-scale detector

Yolov4-tiny compresses the model with shallow convolutions in the backbone and employs only large- and medium-sized object detectors (no small-sized object detectors). As a result, Yolov4-tiny is extremely fast in detection speed and remarkably reliable in detecting large- and medium-sized objects. Yolov4-tiny is reliable in detecting intact fish and fish heads. However, it is counterproductive in detecting fishtail. The principal issue is relying only on large- and medium-sized object detectors; hence, it is ineffective in detecting small-sized objects, such as fishtails. Therefore, we propose attaching an extra-branch as a third-



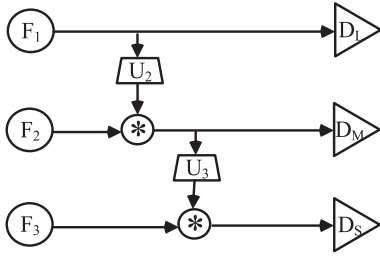


Fig. 5. The three scale detectors of the proposed model.

scale detector using FPN to solve this problem.

The third scale is restored using FPN by combining the up-sampled second-scale features with the third-scale features from the skip connection of the backbone. Subsequently, the integrated feature maps are convolved and utilized by the third-scale detector. The third-scale detector helps detect fishtails that are small objects and failed to be detected using the original YOLOv4-tiny model.

Fig. 5 illustrates the three-scale detectors of YOLO, in which we restored the third-scale detector.  $F_1$ ,  $F_2$ , and  $F_3$  are the feature maps of the backbone with resolution sizes according to their defined scale,  $D_L$  and  $D_M$  are the first scale and second scale detectors, respectively. We added the  $D_S$ , the third-scale detector that combines the up-sampled second scale features ( $U_3$ ) and the feature map of the skip connection of the backbone ( $F_3$ ).

### 3.5. Enhancing YOLOv4-tiny using WCL, Tiny-SPP, BEC, and a third scale detector

This study proposes an enhancement of the YOLOv4-tiny model, called WCL-YOLOv4-tiny, by modifying the backbone of the model, adding Tiny-SPP, and replacing the convolution in the FPN connection using BEC, and attaching a third-scale detector. We modified the backbone by adding WCL as a separate and parallel convolution block, then fusing the main backbone with the WCL feature map at the end of the backbone. WCL utilizes the feature map from the first convolution of the backbone, then convolves it using the several layers of WCL as previously explained. WCL is employed in parallel to the main backbone and has a deeper layer than the main backbone. Tiny-SPP supports the expansion of feature diversity without causing excessive diversity of features and is placed after combining the main backbone with the feature map of WCL.

WCL-YOLOv4-tiny utilizes three detectors of different scales. Therefore, our proposed model requires two FPNs to connect the second-scale and third-scale detectors. The problem faced by FPN is the usage of standard convolution to link between two feature maps. This technique

causes high computational resources requirement. We solve such a problem using BEC to replace the standard convolution and, as a result, combining two features map utilizes a low computational resources. WCL-YOLOv4-tiny further adds a third-scale detector, with a size of  $52 \times 52$ , to enhance the model in detecting small fish body parts, particularly the tail. The WCL-YOLOv4-tiny architecture is presented in Fig. 6.

With the support of WCL, Tiny-SPP, BEC, and a third-scale detector, the WCL-YOLOv4-tiny model generates high with non-excessive diversity of features, has a low computational resources usage, and exhibits high performance in the detection of intact fish, fish head, and fishtail, as proven in the experimental results of this study.

### 3.6. The dataset

We used the FFPD dataset, which contains 600 images added with 4486 annotations. The annotation consists of 1880 fish heads, 1602 fishtails, and 1004 intact fish. It can be downloaded from <https://data.mendeley.com/datasets/pz9jptf7f>. During the experiment, we divided the data into 480 training images (80%) and 120 testing images (20%). The training data contains 1492, 1276, and 810 annotations of fish heads, fishtails, and intact fish. The testing data includes 388, 326, and 192 annotations of fish heads, fishtails, and intact fish. The fish head and fishtail annotations were created on all visible fish heads and fishtails, including those are partially visible in the image due to the overlapping with other objects. On the other hand, intact fish annotations were performed on fish bodies that were fully visible in the image.

Examples of images within the FFPD dataset are presented in Fig. 7, in which there are three types of problems: single fish, multi fish, and complicated fish. The single fish problem, shown in Fig. 7(a), is the task of detecting one fish head, one fishtail, and one intact fish in images that contains a single fish. The multi fish problem, shown in Fig. 7(b), is the task of detecting fish heads, fishtails, and intact fish in images that contain several overlapping fishes, in which each fish was fully visible in the image. Fig. 7(b) shows three heads and three tails are detected. The complicated fish problem, shown in Fig. 7(c-d), is the task of detecting fish heads, fish tails, and intact fish in images that contain several overlapping fishes, in which some of the fish bodies were not fully visible in the image and even some fish heads and fishtails were also not fully visible in the image. Fig. 7(c) shows there are six heads, four tails, and two intact fish to detect. Meanwhile, Fig. 7(d) shows there are eight heads, four tails, and three intact fish to detect. Those detection problems require the development of reliable detection models that are robust towards issues such as object variations, background variations, lighting, and clipped objects.

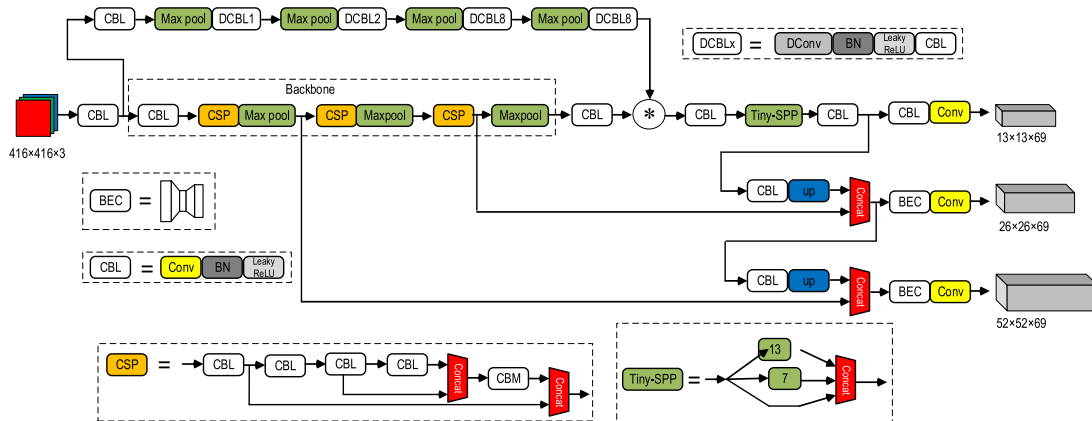


Fig. 6. The WCL-YOLOv4-tiny architecture.

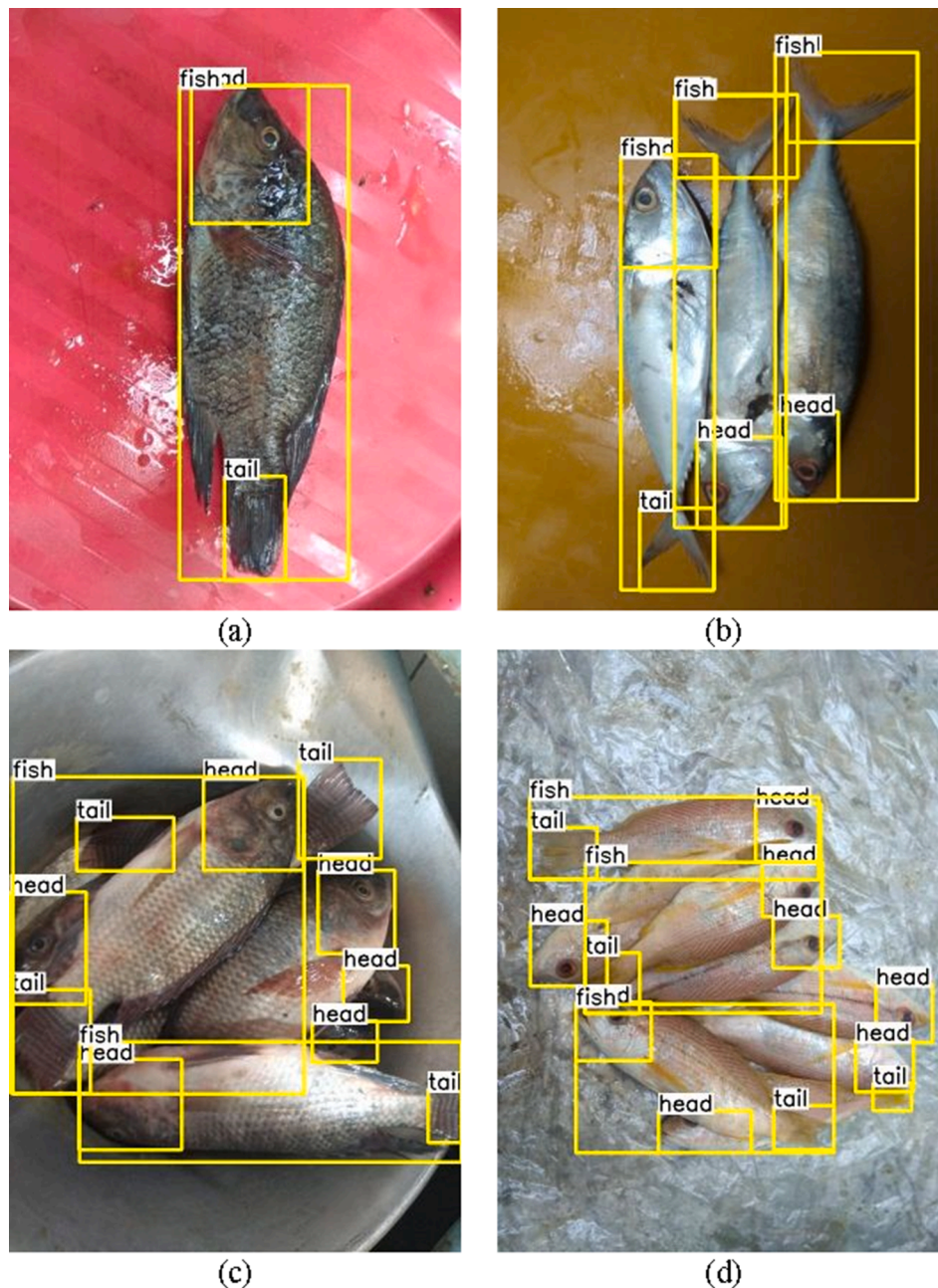


Fig. 7. Sample images of the FFPD dataset. (a). Single fish, (b) Multi fish, (c)-(d) Complicated fish.

### 3.7. Experimental scenarios

We evaluated our proposed model using 480 images as training data and 120 images as test data. Each of our proposed components for enhancing the YOLOv4-tiny model was tested to measure the improvement obtained. The performance of the WCL-YOLOv4-tiny model was compared to YOLOv3, YOLOv4, and YOLOv4-tiny. In this study, we focus on the performance of a model in detecting the target objects, measured using Recall and Precision. In addition, Average Precision (AP) was used as a general metric in object detection and BFLOPS and megabytes (MB) to measure computational volume and model size, respectively.

## 4. Results and discussions

During experiments, we utilized the Darknet framework using the configurations shown in Table 1. We utilized tools provided by Google

Colab, and YOLO parameters used in the experiment, such as max batch, steps, input image size, momentum, and learning rate, are shown in the table. We recalculated the anchors from the annotated dataset, both head, tail, and intact fish objects, as presented in Table 1 for the two- and three-scale models. The detection stage uses a confidence threshold of 0.25 and an NMS threshold of 0.5.

This section analyzes Recall and Precision to evaluate each model in detecting the target and correct objects, respectively. In addition, Average Precision (AP) is used as a curve between Precision and Recall to analyze the performance of each model.

### 4.1. The effect on the performance of each module on fish head detection

The fish head is the front part of the body, which contains several essential organs used to observe the freshness, such as the eyes, gills, and even the appearance of the head itself. The Recall results of fish head

**Table 1**

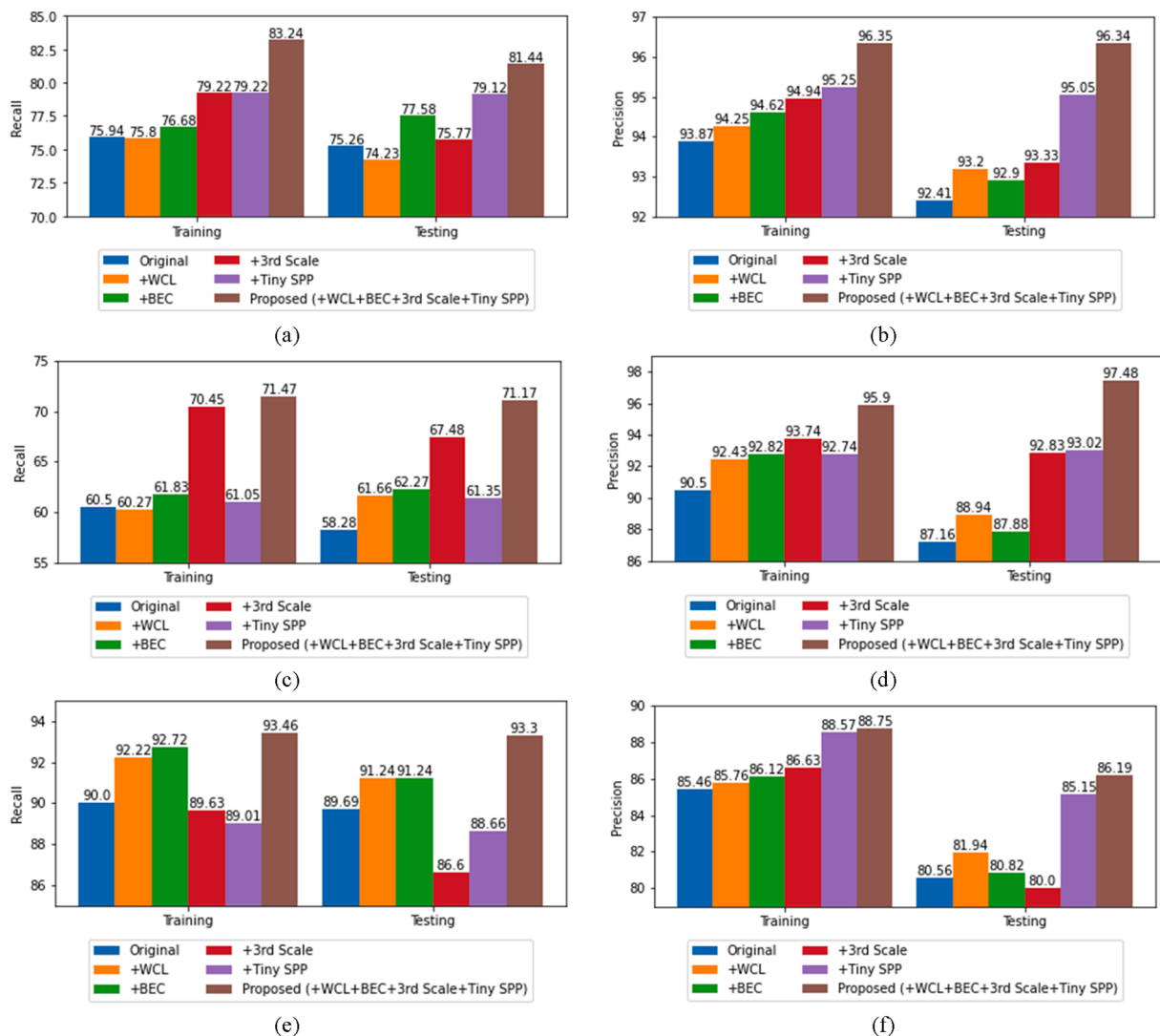
Experiment configuration.

Configuration	Parameter
Environment	Google Colab
GPU	Nvidia
fAcceleration	CUDA 11.0, cuDNN 7.6.5
Library	OpenCV 3.2.0
Max batch	6000
Steps	4800, 5400
Image size	416 × 416
Momentum	0.9
Learning rate	0.00261
Model's anchor with two scale	41, 57, 69, 50, 65, 78, 93, 94, 129, 301, 305, 137
Model's anchor with three scale	37, 49, 65, 36, 46, 75, 63, 58, 68, 85, 95, 73, 96, 105, 127, 303, 294, 137
Confidence threshold	0.25
NMS threshold	0.5

detection are shown in Fig. 8(a). It can be seen in Fig. 8 (a) that the original Yolov4-tiny model exhibited good performance in detecting fish heads, achieving a Recall of 75.94% and 75.26% for the training and testing sessions, respectively. Adding the BEC and Tiny-SPP modules to the Yolov4-tiny model impacts a significant Recall increment in the testing session. BEC increased the Recall by 2.32% to 77.58%. Meanwhile, Tiny-SPP increased the Recall by 3.86% to 79.12%. WCL

decreased the Recall by 1.03% in the testing session, which is not a significant decrease. Yolov4-tiny with third-scale detector attachment obtains a significant increase in Recall, in which the Recall was increased by 3.28% to 79.22% in the training session. However, in the testing session, the addition of a third-scale detector increased the Recall by only 0.51% to 75.77%. It can be concluded that the addition of WCL only or a third-scale detector only to the Yolov4-tiny model did not significantly increase the performance of Yolov4-tiny in detecting fish heads. This result is reasonable because the fish head is not a small object. Therefore, attaching a third-scale detector was ineffective in improving the performance. The proposed WCL-Yolov4-tiny model achieved a significantly higher Recall than the Yolov4-tiny model. The WCL-Yolov4-tiny model achieved a Recall of 83.24% and 81.44% in the training and testing sessions, respectively. The Recall was increased by 7.3% and 6.18% in the training session and testing session, respectively, compared to the Yolov4-tiny model. This result shows that the addition of all the modules to the Yolov4-tiny model significantly improved the model's performance in detecting fish heads based on Recall.

The Precision results of fish head detection are shown in Fig. 8(b). It shows that the original Yolov4-tiny model achieved a Precision of 93.87% and 92.41% for the training session and testing session, respectively. In the training session, four modules positively impacted the performance of the Yolov4-tiny model, in which all four modules increased the Precision. The Tiny-SPP module increased the Precision by



**Fig. 8.** The performance model. (a), (c), (e) Recall of head, tail, and intact fish, respectively. (b), (d), (f) Precision of head, tail, and intact fish, respectively.



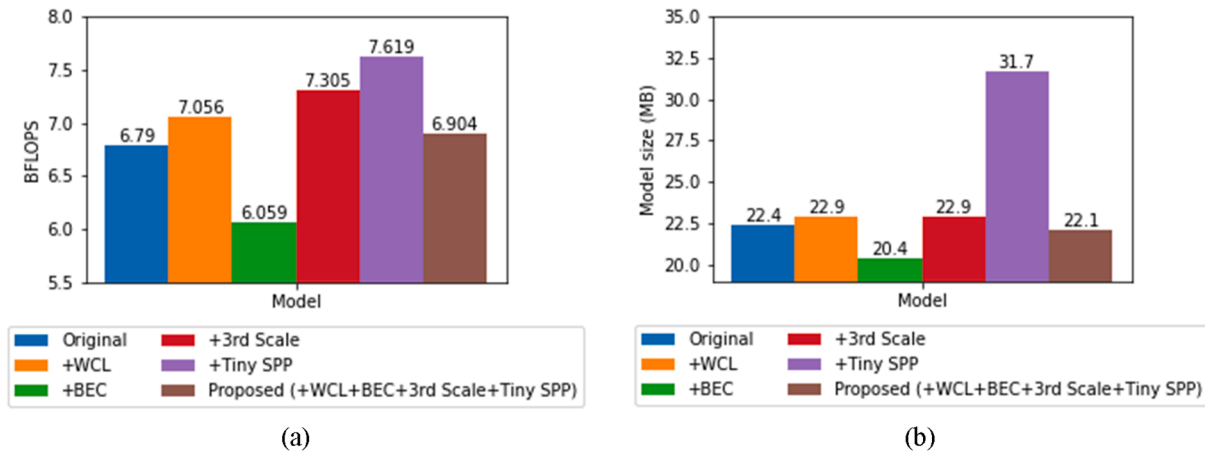


Fig. 9. Resource requirements; (a) Computation volume; (b) Model size.

the most significant margin; the Precision of the YOLOv4-tiny model in the training session obtained 95.25%. In the testing session, similar results were exhibited, in which four modules increased the Precision of the YOLOv4-tiny model. The Tiny-SPP module increased the Precision by the most significant margin, increasing the Precision of the YOLOv4-tiny model in the testing session to 95.05%. The proposed WCL-YOLOv4-tiny model achieved a significantly higher Precision than the YOLOv4-tiny model. The WCL-YOLOv4-tiny model achieved a Precision of 96.35% and 96.34% in the training and testing sessions, respectively. It was increased by 2.48% and 3.93%, respectively, compared to the YOLOv4-tiny model. Therefore, it can be concluded that adding all modules impacts the performance of the YOLOv4-tiny model in detecting fish heads. The model also prevents excessive diversity of features indicated by the decrease in false positives.

#### 4.2. The effect on the performance of each module on tail detection

The Recall results of fishtail detection are shown in Fig. 8(c). The original YOLOv4-tiny model exhibited low performance in detecting fish heads, with 60.5% and 58.28% for the training and testing sessions, respectively. The YOLOv4-tiny model with one of the WCL, BEC, and Tiny-SPP modules did not significantly increase the Recall in the training session, while in the testing session, the most significant improvement in Recall was achieved by the BEC addition with a Recall of 62.27%. Attaching the third-scale detector to the YOLOv4-tiny model significantly increased Recall. The Recall increased by 9.95% to 70.45% and 9.2% to 67.48% in training and testing sessions, respectively. These results indicate that the third-scale detector effectively detects small objects such as tails. The proposed WCL-YOLOv4-tiny model achieved a significantly higher Recall than the YOLOv4-tiny model, with 71.47% and 71.17% in the training and testing sessions, respectively. The Recall was increased by 10.97% and 12.89% in the training session and testing session, respectively, compared to the YOLOv4-tiny model. This result shows that adding all modules to the YOLOv4-tiny model significantly improved the model's performance in detecting fishtails.

The Precision results of fishtail detection are shown in Fig. 8(d). The original YOLOv4-tiny model achieved a Precision of 90.5% and 87.16% for the training session and testing session, respectively. In the training session, four modules impacted the performance of the YOLOv4-tiny model, in which four modules increased the Precision. The third-scale detector module increased the Precision by the most significant margin, increasing the Precision of the YOLOv4-tiny model in the training session to 93.74%. In the testing session, four modules also impacted the performance of the YOLOv4-tiny model, in which all modules increased the Precision. The Tiny-SPP module increased the Precision by the most significant margin, increasing the Precision of the YOLOv4-tiny model in the testing session to 93.02%. The proposed WCL-YOLOv4-tiny model

achieved a significantly higher Precision than the YOLOv4-tiny model. It achieved 95.9% and 97.48% in the training and testing sessions, respectively. The Precision was increased by 5.4% and 10.32% in the training session and testing session, respectively, compared to the YOLOv4-tiny model. Therefore, it can be concluded that adding all modules impacts the performance of the YOLOv4-tiny model in detecting fishtails. The model also prevents excessive diversity of features indicated by the decrease in false positives. The main concern of this study, which was the low detection of fishtails, was overcome by the proposed WCL-YOLOv4-tiny model.

#### 4.3. The effect on the performance of each module on intact fish detection

The Recall results of intact fish detection are shown in Fig. 8(e). The original YOLOv4-tiny model exhibited good performance in detecting intact fish, achieving a Recall of 90% and 89.69% for the training and testing sessions, respectively. The addition of the WCL and BEC modules to the YOLOv4-tiny model similarly improved the performance. In the training session, the WCL and BEC module increased the Recall by 2.22% and 2.72%, respectively, while in the testing session, they increased the Recall by 1.55%. The addition of the third-scale detector to the YOLOv4-tiny model slightly decreases performance, whereas it decreased the Recall by 0.37% and 3.09% in the training session and testing session, respectively. This is because the third-scale detector is only effective in detecting small-sized objects such as fishtails and is quite ineffective for large-sized objects, such as intact fish. The use of the Tiny-SPP module slightly decreased the Recall by 0.99% and 1.03% in the training session and testing session, respectively. The WCL-YOLOv4-tiny model achieved Recall of 93.46% and 93.3%, with an improvement of 3.46% and 3.61% in the training and testing sessions, respectively, compared to the YOLOv4-tiny model. This result shows that adding all modules to the YOLOv4-tiny model significantly improved the model's performance in detecting intact fish.

The Precision results of intact fish detection are shown in Fig. 8(f). It shows that the original YOLOv4-tiny model achieved a Precision of 85.46% and 80.56% for the training session and testing session, respectively. The YOLOv4-tiny model with the addition of a single module among one of the WCL, BEC, and third-scale detector modules did not significantly increase the Precision in the training session. In the testing session, the addition of the WCL and BEC module to the YOLOv4-tiny model slightly increased the Precision by 1.38% and 0.26%, respectively, while the addition of the third-scale detector module to the YOLOv4-tiny model decreased the Precision by 0.56%. Meanwhile, adding the Tiny-SPP module to the YOLOv4-tiny model significantly increased the Precision by 3.11% and 4.59% in the training session and testing session, respectively. The WCL-YOLOv4-tiny model achieved Precision of 88.75% and 86.19%, with an improvement of 3.29% and



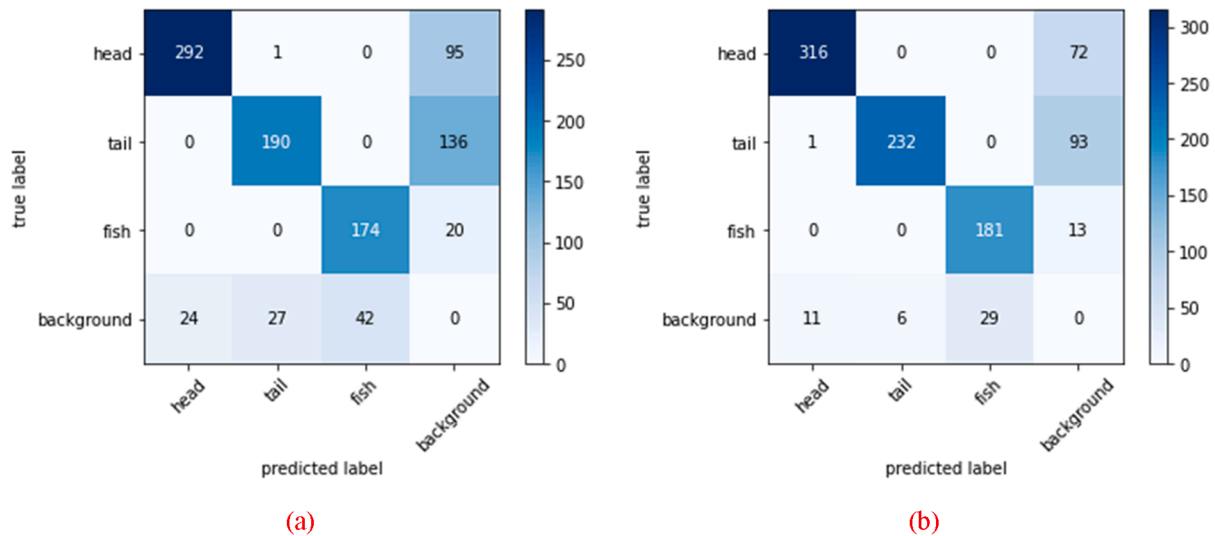


Fig. 10. Confusion matrix of testing data; (a) Yolov4-tiny original; (b) WCL-Yolov4-tiny.

5.63% in the training and testing sessions, respectively. Therefore, it can be concluded that adding all modules impacts the performance of the Yolov4-tiny model in detecting intact fish. The main motivation for this improvement was the addition of the Tiny-SPP module; it added diversity to the fish head features.

#### 4.4. Computational volume and model size evaluation

We use BFLOPS to quantify the computational volume and evaluate each module in terms of resource utilization. BFLOPS is calculated from feature maps' combined size and resolution and kernel size. The computational resource of each model is shown in Fig. 9(a). The addition of the WCL, third-scale detector and Tiny-SPP modules increases the computational resource of the Yolov4-tiny model. On the other hand, BEC achieves its principal goal of decreasing the computational resource of the Yolov4-tiny model while also maintaining its performance with a decrement of 0.73%. The proposed WCL-Yolov4-tiny and Yolov4-tiny model has a computational volume of 6.904 BFLOPS and 6.79 BFLOPS, respectively. By considering the trade-off between performance and computational resources, the WCL-Yolov4-tiny model has a slightly larger computational resource. Furthermore, it is significantly better performance than the Yolov4-tiny model. Hence, it can be concluded that the slight increase in computational resources is worth the trade-off.

We also use model size to evaluate the feasibility of using our proposed model in devices with limited storage and computation resources. We measured the storage requirement of each model with an input size of  $416 \times 416$ . It can be seen in Fig. 9(b) that the addition of WCL and the third scale detector does not require high additional storage as in BEC. The original Yolov4-tiny model only requires 22.4 MB of storage, while the addition of Tiny-SPP to the Yolov4-tiny model increases its storage requirement to 31.7 MB. Our proposed WCL-Yolov4-tiny model has a lower storage requirement of 22.4 MB than the Yolov4-tiny model. This shows that our proposed model applies to devices with limited storage resources.

#### 4.5. Performance comparison with the original version

We compared detection result of the WCL-Yolov4-tiny and the original version model on the same testing data. The confusion matrix shown in Fig. 10 present four classes, namely head, tail, fish, and background. The background class is used to label other objects that are mistaken detected. The main diagonal presents the true positive result. The detection results in all body parts show that WCL-Yolov4-tiny is

superior to both head, tail, and fish, with the number of detected objects 316, 232, and 181, respectively, while the original versions are 292, 190, 174, respectively. WCL-Yolov4-tiny also reduces the number of ground truths that fail to detect. The fourth column shows there are 72 heads, 93 tails, and 13 fish that fail to detect. The result of tail detection shows a significant improvement, from 136 undetected tails decrease to 93. The number of false-positive objects was also reduced by WCL-Yolov4-tiny, where the head, tail, and fish detected are 11, 6, and 29 objects, respectively, while the original version reached 24, 27, and 42 objects, respectively. All false positive objects are background or other parts that should not be detected. There is almost no misdetection between head, tail, and fish between the two models except for one object. WCL-Yolov4-tiny detects one tail object like a head, while the original Yolov4-tiny detects one head as a tail. We show the failure case detection of WCL-Yolov4-tiny in the following subsection.

#### 4.6. Performance evaluation of different models

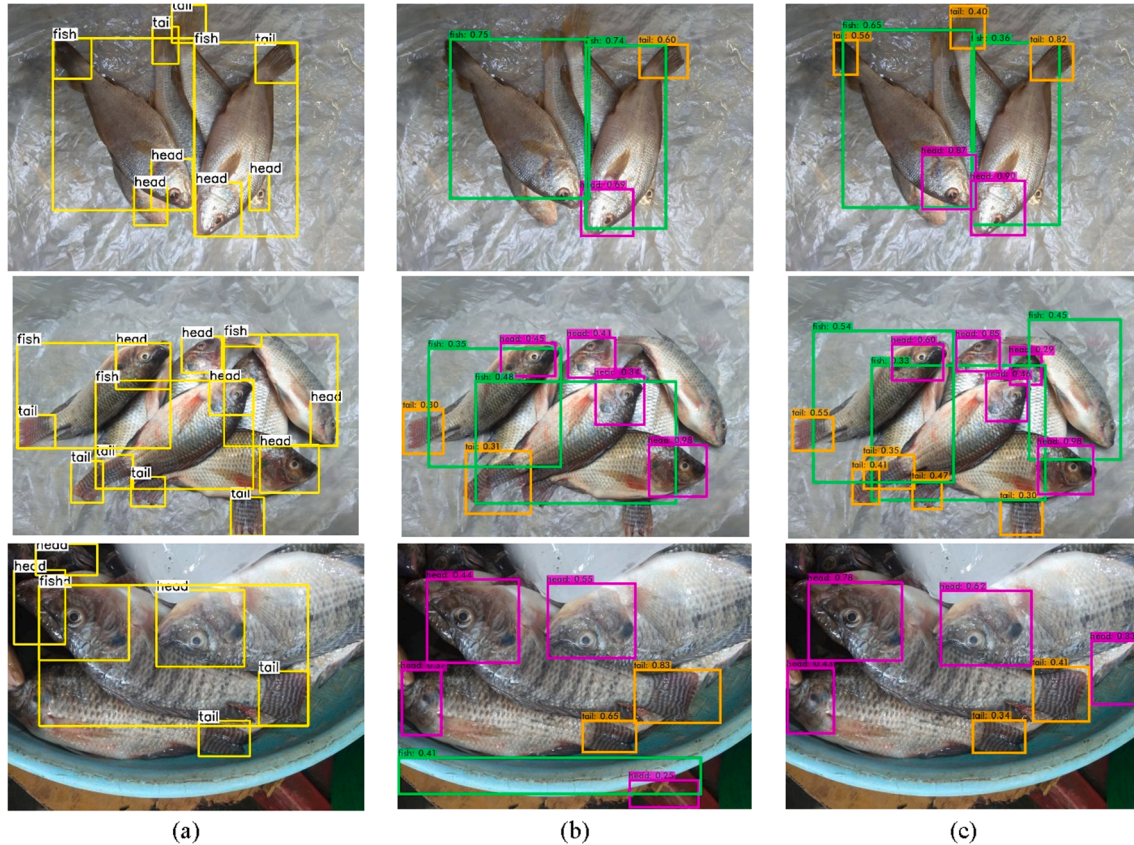
The performance of the proposed WCL-Yolov4-tiny model was compared to the original Yolov4-tiny model and the Yolov4-tiny model with the original SPP (Yolov4-tiny + SPP). The Yolov4-tiny + SPP model used more computational resources than the proposed WCL-Yolov4-tiny model and the original Yolov4-tiny model, with BFLOPS of 8.03, 6.91, and 6.79, respectively. The proposed WCL-Yolov4-tiny model uses slightly more computational resources than the Yolov4-tiny model but achieves better detection performance. The number of fish head, fish tails, and intact fish detected by our proposed model was higher than the other two models, with fewer false positives. The mAP with IOU = 0.5 is used to evaluate the overall performance of the models. In our problem (detecting fish head, fishtail, and intact fish), our proposed model performed better than the other two models, where the AP of the proposed model for the fish head, fishtail, and intact fish was 94.07%, 90.3%, and 92.78%, respectively.

Furthermore, an mAP of 92.38%, achieved by the proposed model, was significantly better than the mAP of the others, which was 84.76% and 85.47% for the Yolov4-tiny model and the Yolov4-tiny + SPP model, respectively. We also compared the performance of the proposed model with the Yolov3 and Yolov4 models, which have deeper convolution layers in the backbone and several other layers within the model. The Yolov3 and Yolov4 models' performance was better than the WCL-Yolov4-tiny model in which the Yolov3, Yolov4, and WCL-Yolov4-tiny model obtained a mAP of 96%, 97.43%, and 92.38%, respectively. However, the Yolov3 and Yolov4 models uses significantly more computational resources, in which the Yolov3, Yolov4, and WCL-

**Table 2**

Performance evaluation of different models.

Model	Input Size	BFLOPS	TP			FP			AP			mAP
			Head	Tail	Fish	Head	Tail	Fish	Head	Tail	Fish	
Ground Truth	–	–	388	326	194	–	–	–	–	–	–	–
Yolov4-tiny original	416 × 416	6.79	292	190	174	24	28	42	87.09	77.23	89.96	84.76
Yolov4-tiny + SPP	416 × 416	8.03	295	206	172	17	29	31	87.87	78.23	90.31	85.47
WCL-Yolov4-tiny (Proposed)	416 × 416	6.91	316	232	181	12	6	29	94.07	90.3	92.78	92.38
Yolov3	416x416	65.32	377	299	190	26	20	30	98.01	92.88	97.11	96
Yolov4	416 × 416	59.58	385	310	188	14	28	24	98.91	96.63	96.74	97.43
WCL-Yolov4-tiny (Proposed)	512 × 512	10.46	324	235	175	18	19	22	92.8	88.08	94.56	91.82
WCL-Yolov4-tiny (Proposed)	608 × 608	14.75	325	252	174	15	7	36	94.91	91.72	92.7	93.11



**Fig. 11.** Visual comparison between (a) Ground truth and fish detection results using: (b) Yolov4-tiny model; (c) WCL-Yolov4-tiny model.

Yolov4-tiny have computational volumes of 65.32 BFLOPS, 59.58 BFLOPS, 6.91 BFLOPS, respectively. This shows that Yolov3 and Yolov4 are not applicable in devices with limited storage and computational resources.

We also increased the input size of the proposed model to  $512 \times 512$  and  $608 \times 608$  during the experiment to determine the impact of increasing the input size on the detection performance. Table 2 shows that an increase in input size increases the computational resource usage. A WCL-Yolov4-tiny model with an input size of  $416 \times 416$ ,  $512 \times 512$ , and  $608 \times 608$  has a computational volume of 6.91 BFLOPS, 10.46 BFLOPS, and 14.75 BFLOPS, respectively. The proposed model with an input size of  $608 \times 608$  achieved the highest AP of 94.91%, 91.72%, and 92.7% for the fish head, fishtail, and intact fish, respectively. This is slightly better than the AP of the proposed model with an input size of  $416 \times 416$ , which were 94.07%, 90.3%, and 92.78% for fish head, fishtail, and intact fish, respectively. The trade-off between computational volume and detection performance when the input size of the model is increased is not worth it. Therefore, it can be concluded that the

proposed WCL-Yolov4-tiny model with an input size of  $416 \times 416$  is the most appropriate model to be employed in devices with limited storage and computational resource. It is reasonable due to the WCL-Yolov4-tiny using a low computational resource with high detection performance.

#### 4.7. Detection results

The head, tail and intact fish detection results are presented in Fig. 11. WCL-Yolov4-tiny showed satisfying results in detecting objects. The first image contains four heads, four tails, and two intact fish; Yolov4-tiny detects only one head, one tail, and two intact fish, while WCL-Yolov4-tiny detects two heads, three tails, and two intact fishes. These results indicate that our proposal has sufficient features diversity to detect more tails and heads where more heads and tails are detected than the original Yolov4-tiny. Yolov4-tiny detected 8 and 13 objects which were true positive. These results also show that WCL-Yolov4-tiny can detect more objects with modifications addressed to the original Yolov4-tiny. The third image consists of four heads, two tails, and one





Fig. 12. Failure cased in detection by WCL-Yolov4-tiny model.

intact fish; Yolov4-tiny and WCL-Yolov4-tiny detect two heads and two tails, respectively. In addition, Yolov4-tiny and WCL-Yolov4-tiny also detected three non-head and tails; where each detected three and two false-positive objects. The results show that WCL-Yolov4-tiny outperformed the original Yolov4-tiny where more and fewer true positives and false positives were achieved. With better detection performance, where many true objects were detected and fewer false ones were avoided, WCL-Yolov4-tiny is appropriate for multifarious organizations. Commercial fish packaging companies require a lightweight, fast, and high-performance processing machine in sorting the freshness of fish; this model supports the requirement in determining the position of fish and body parts. The personal users of mobile applications with limited storage and computation resources also require a simple and low computational resource system to localize mixed fish in one bucket and their freshness level.

The proposed WCL-Yolov4-tiny achieved a significant improvement in object detection. The tail detection performance as a small object increased true positives while decreasing false positives. False positives detected in this case are background or intact fish whose parts are not observable, for example, intact fish with head covered by other objects; and the model detects them as intact fish, as shown by Fig. 12 (a). One fish was detected with 0.56 confidence. Notwithstanding achieving fairly good confidence, the object should not be detected due to an

incomplete body part. Another failed case is Fig. 12 (b), the model detected the background as fish with 0.27 confidence. This background should not be detected, and we can ignore it by increasing the confidence threshold. However, increasing the confidence threshold also reduced the true positives achieved. For example, tail and fish with 0.26 and 0.28 confidence, respectively (Fig. 12(a)), are dismissed if the confidence threshold increases to 0.3. Hence, by holding a confidence threshold of 0.25, we achieved more true positive objects with fewer false-positive ones. Another case of misdetection is Fig. 12(c), where a tail with 0.54 confidence should not be detected as a tail. This body part is a fin with an appearance looking like a tail. The similarity of the visual appearance caused the failure of detection by our proposed model. Some detection failures in this model are nevertheless an understandable problem due to natural causes and parameter trade-offs used during the experiment. The number of failures is slight because only four of the 120 test images were detected incorrectly.

## 5. Conclusion

This research proposes an object detector model that modifies the Yolov4-tiny model, called the WCL-Yolov4-tiny model. The model performs better in detecting intact fish, fish eyes, and fishtails. The object detected supports the fish freshness classification and is worth applying

to devices with limited storage and computational resources. The WCL-Yolov4-tiny model uses WCL to expand feature diversity, Tiny-SPP to balance feature diversity and avoid excessive diversity of features, BEC to reduce computational resource usage, and a third-scale detector to detect small-sized objects such as fishtails. The experimental results show that the proposed WCL-Yolov4-tiny model achieved a Precision, Recall, AP, and mAP of 97.48%, 93.3%, 94.07%, and 92.38%, respectively, which were higher than the original Yolov4-tiny model. This result indicates that the proposed model can expand and avoid excessive diversity of features and better performance in detecting objects with different sizes. The proposed model is lightweight, high-performance, and requires a low computational resource. Hence, it is applicable in mobile devices with limited storage and computational resource in detecting fish and body part localization. Therefore, it can be applied to personal users and the commercial fishing industry.

This study found a new model that increases Recall on tail detection from the original version by + 12.91% to 71.17%; indicating a significant improvement. However, this performance was below 80%; the subsequent research recommendation such as data augmentation and post-processing could be explored to improve the detection performance.

#### CRedit authorship contribution statement

**Eko Prasetyo:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft. **Nanik Suciati:** Funding acquisition, Validation, Resources, Writing – review & editing, Visualization, Supervision. **Chastine Fatichah:** Validation, Resources, Writing – review & editing, Visualization, Supervision.

#### Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

#### Acknowledgment

This research is funded by Ministry of Education, Culture, Research, and Technology, Indonesia under grant of doctoral dissertation research year 2021-2022.

#### References

- Alshdaifat, N.F.F., Talib, A.Z., Osman, M.A., 2020. Improved deep learning framework for fish segmentation in underwater videos. *Ecol. Inform.* 59, 101121 <https://doi.org/10.1016/j.ecoinf.2020.101121>.
- Azimjonov, J., Özmen, A., 2021. A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways. *Adv. Eng. Informatics* 50, 101393. <https://doi.org/10.1016/j.aei.2021.101393>.
- Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. <https://doi.org/10.48550/arxiv.2004.10934>.
- Cai, K., Miao, X., Wang, W., Pang, H., Liu, Y., Song, J., 2020. A modified YOLOv3 model for fish detection based on MobileNetV1 as backbone. *Aquac. Eng.* 91, 102117 <https://doi.org/10.1016/j.aquaeng.2020.102117>.
- Cao, Z., Liao, T., Song, W., Chen, Z., Li, C., 2021. Detecting the shuttlecock for a badminton robot: A YOLO based approach. *Expert Syst. Appl.* 164, 113833 <https://doi.org/10.1016/j.eswa.2020.113833>.
- Christensen, J.H., Mogensen, L.V., Galeazzi, R., Andersen, J.C., 2018. Detection, Localization and Classification of Fish and Fish Species in Poor Conditions using Convolutional Neural Networks, in: AUV 2018–2018 IEEE/OES Autonomous Underwater Vehicle Workshop, Proceedings. In: Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/AUV.2018.8729798>.
- Dutta, M.K., Issac, A., Minhas, N., Sarkar, B., 2016. Image processing based method to assess fish quality and freshness. *J. Food Eng.* 177, 50–58.
- Erasmus, V.N., Kadhila, T., Thyberg, K., Kamara, E.N., Bauleth-D'Almeida, G., 2021. Public perceptions and factors affecting domestic marine fish consumption in Namibia, southwestern Africa. *Reg. Stud. Mar. Sci.* 47, 101921.
- Haas, T., Schubert, C., Eickhoff, M., Pfeifer, H., 2020. BubCNN: Bubble detection using Faster RCNN and shape regression network. *Chem. Eng. Sci.* 216, 115467 <https://doi.org/10.1016/j.ces.2019.115467>.
- Hashanuzzaman, M., Bhowmik, S., Rahman, M.S., Zakaria, M.U.M.A., Voumik, L.C., Mamun, A. Al, 2020. Assessment of food safety knowledge, attitudes and practices of fish farmers and restaurants food handlers in Bangladesh. *Heliyon* 6, e05485. <https://doi.org/10.1016/j.heliyon.2020.e05485>.
- He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision. <https://doi.org/10.1109/ICCV.2017.322>.
- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.
- Hu, X., Liu, Y., Zhao, Z., Liu, J., Yang, X., Sun, C., Chen, S., Li, B., Zhou, C., 2021. Real-time detection of uneaten feed pellets in underwater images for aquaculture using an improved YOLO-V4 network. *Comput. Electron. Agric.* 185, 106135 <https://doi.org/10.1016/j.compag.2021.106135>.
- Huang, Z., Wang, J., Fu, X., Yu, T., Guo, Y., Wang, R., 2020. DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection. *Inf. Sci. (N.Y.)* 522, 241–258.
- Issac, A., Dutta, M.K., Sarkar, B., 2017. Computer vision based method for quality and freshness check for fish from segmented gills. *Comput. Electron. Agric.* 139, 10–21.
- Jalal, A., Salman, A., Mian, A., Shortis, M., Shafait, F., 2020. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecol. Inform.* 57, 101088 <https://doi.org/10.1016/j.ecoinf.2020.101088>.
- Jose, J.A., Kumar, C.S., Sureshkumar, S., 2022. Tuna classification using super learner ensemble of region-based CNN-grouped 2D-LBP models. *Information Processing in Agriculture* 9 (1), 68–79.
- Kumar, A., Kalia, A., Verma, K., Sharma, A., Kaushal, M., 2021. Scaling up face masks detection with YOLO on a novel dataset. *Optik (Stuttg.)* 239, 166744 <https://doi.org/10.1016/j.ijleo.2021.166744>.
- Kunjulakshmi, S., Harikrishnan, S., Murali, S., D'Silva, J.M., Binsi, P.K., Murugadas, V., Alfiya, P.V., Delfiya, D.S.A., Samuel, M.P., 2020. Development of portable, non-destructive freshness indicative sensor for Indian Mackerel (Rastrelliger kanagurta) stored under ice. *J. Food Eng.* 287, 110132 <https://doi.org/10.1016/j.jfoodeng.2020.110132>.
- Lalabadi, H.M., Sadeghi, M., Mireei, S.A., 2020. Fish freshness categorization from eyes and gills color features using multi-class artificial neural network and support vector machines. *Aquac. Eng.* 90, 102076 <https://doi.org/10.1016/j.aquaeng.2020.102076>.
- Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. Institute of Electrical and Electronics Engineers Inc., pp. 936–944. <https://doi.org/10.1109/CVPR.2017.106>.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single shot multibox detector, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2).
- Loey, M., Manogaran, G., Taha, M.H.N., Khalifa, N.E.M., 2021. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustain. Cities Soc.* 65, 102600 <https://doi.org/10.1016/j.scs.2020.102600>.
- Mitra, S., Khatun, M.N., Prodhan, M.M.H., Khan, M.A., 2021. Consumer preference, willingness to pay, and market price of capture and culture fish: Do their attributes matter? *Aquaculture* 544, 737139.
- N.S., A., D., S., R.K., 2021. Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries. *Ecol. Inform.* 61, 101248. <https://doi.org/10.1016/J.ECOINF.2021.101248>.
- Nian, R., Wang, X., Che, R., He, B., Xu, X., Li, P., Lendasse, A., 2017. Online fish tracking with portable smart device for ocean observatory network, in: OCEANS 2017 - Anchorage. pp. 1–7.
- Peng, C., Zhao, K., Lovell, B.C., 2020. Faster ILOD: Incremental learning for object detectors based on faster RCNN. *Pattern Recognit. Lett.* 140, 109–115. <https://doi.org/10.1016/j.patrec.2020.09.030>.
- Prabhakar, P.K., Vatsa, S., Srivastav, P.P., Pathak, S.S., 2020. A comprehensive review on freshness of fish and assessment: Analytical methods and recent innovations. *Food Res. Int.* 133, 109157.
- Prados, R., Garcia, R., Gracias, N., Neumann, L., Vagstol, H., 2017. Real-time fish detection in trawl nets, in: OCEANS 2017 - Aberdeen. In: Institute of Electrical and Electronics Engineers Inc, pp. 1–5. <https://doi.org/10.1109/OCEANSE.2017.8084760>.
- Prasetyo, E., Suciati, N., Fatichah, C., 2021. Yolov4-tiny and Spatial Pyramid Pooling for Detecting Head and Tail of Fish. *International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*. 157–161.
- Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. In: You only look once: Unified, real-time object detection, in: IEEE Computer Society, pp. 779–788. <https://doi.org/10.1109/CVPR.2016.91>.
- Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement.
- Redmon, J., Farhadi, A., 2017. YOLO9000: Better, faster, stronger, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. <https://doi.org/10.1109/CVPR.2017.690>.
- Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (6), 1137–1149.
- Salman, A., Maqbool, S., Khan, A.H., Jalal, A., Shafait, F., 2019. Real-time fish detection in complex backgrounds using probabilistic background modelling. *Ecol. Inform.* 51, 44–51. <https://doi.org/10.1016/j.ecoinf.2019.02.011>.



- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation Mark. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. <https://doi.org/10.1109/CVPR.2018.00474>.
- Sengar, N., Gupta, V., Dutta, M.K., Travieso, C.M., 2018. Image Processing Based Method For Identification Of Fish Freshness Using Skin Tissue, in: International Conference on &quot;Computational Intelligence and Communication Technology&quot;, CICT 2018. <https://doi.org/10.1109/CICT.2018.8480265>.
- Sharma, V., Mir, R.N., 2022. Saliency guided faster-RCNN (SGFr-RCNN) model for object detection and recognition. J. King Saud Univ. - Comput. Inf. Sci. 34 (5), 1687–1699.
- Sung, M., Yu, S.C., Girdhar, Y., 2017. Vision based real-time fish detection using convolutional neural network, in: OCEANS 2017 - Aberdeen. In: Institute of Electrical and Electronics Engineers Inc, pp. 1–6. <https://doi.org/10.1109/OCEANSE.2017.8084889>.
- Yin, Y., Li, H., Fu, W., 2020. Faster-YOLO: An accurate and faster object detection method. Digit. Signal Process. A Rev. J. 102, 102756 <https://doi.org/10.1016/j.dsp.2020.102756>.
- Yu, C., Fan, X., Hu, Z., Xia, X., Zhao, Y., Li, R., Bai, Y., 2020a. Segmentation and measurement scheme for fish morphological features based on Mask R-CNN. Information Processing in Agriculture 7 (4), 523–534.
- Yu, G., Wang, L., Hou, M., Liang, Y., He, T., 2020. An adaptive dead fish detection approach using SSD-MobileNet, in: Proceedings - 2020 Chinese Automation Congress, CAC 2020. Institute of Electrical and Electronics Engineers Inc., pp. 1973–1979. <https://doi.org/10.1109/CAC51589.2020.9326648>.