Fish Classification System Using YOLOv3-ResNet18 Model for Mobile Phones

Suryadiputra Liawatimena[1*], Edi Abdurachman[2], Agung Trisetyarso[3], Antoni Wibowo[4], Muhamad Keenan Ario[5], and Ivan Sebastian Edbert[6]
[1−4]Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University
Jakarta 11480, Indonesia
[1,5,6]Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University
Jakarta 11480, Indonesia
[1]Computer Engineering Department, Faculty of Engineering, Bina Nusantara University
Jakarta 11480, Indonesia
Email: [1]suryadi@binus.edu, [2]edia@binus.ac.id, [3]trisetyarso@binus.ac.id, [4]anwibowo@binus.edu, [5]muhamad.ario@binus.ac.id, [6]ivan.edbert@binus.edu

***Abstract*—Every country in the world needs to report its fish production to the Food and Agriculture Organization of the United Nations (FAO) every year. In 2018, Indonesia ranked top five countries in fish production, with 8 million tons globally. Although it ranks top five, the fisheries in Indonesia are mostly dominated by traditional and small industries. Hence, a solution based on computer vision is needed to help detect and classify the fish caught every year. The research presents a method to detect and classify fish on mobile devices using the YOLOv3 model combined with ResNet18 as a backbone. For the experiment, the dataset used is four types of fish gathered from scraping across the Internet and taken from local markets and harbors with a total of 4,000 images. In comparison, two models are used: SSD-VGG and autogenerated model Huawei ExeML. The results show that the YOLOv3-ResNet18 model produces 98.45% accuracy in training and 98.15% in evaluation. The model is also tested on mobile devices and produces a speed of 2,115 ms on Huawei P40 and 3,571 ms on Realme 7. It can be concluded that the research presents a smaller-size model which is suitable for mobile devices while maintaining good accuracy and precision.**

***Index Terms*—Fish Classification System, YOLOv3-ResNet18 Model, Mobile Phone**

I. Introduction

ACCORDING to the Food and Agriculture Organization of the United Nations (FAO), Indonesia ranked top five countries in fish production with 8 million tons in 2018 [1]. All countries must monitor and report fish caught every year to FAO to avoid overfishing. Although Indonesia is a maritime country with 62% of the area covered by the sea, it is still dominated by small-scale industries that use traditional ships and equipment [2]. Hence, manual calculation is still used to determine the total and type of caught fish. It makes the report inaccurate.

To make the data and report more reliable and accurate, computer vision, one of the famous research topics, can help sailors and the Indonesian government to calculate and classify the fish produced by small industries. Computer vision is already used in other fields, such as facial expression [3], healthcare [4, 5] and agriculture [6, 7]. Then, many researchers propose their methods. The examples are Region-Based Convolutional Neural Network (R-CNN) [8], Fast R-CNN [9], Faster R-CNN [10] that use the Region Proposal Network method, Single Shot Detector (SSD) [11], and You Only Look Once (YOLO) [12]. YOLO is one of favorite topics because it has fast detection speed and retains good accuracy. It predicts objects using a single neural network that divides into multiple regions and predicts a boundary box for an object in an image.

Many studies have also analyzed fish. For example, YOLOv3 is used to detect underwater object recognition [13]. Another research applies MobileNetv1 as a backbone to detect and classify fish to produce a more accurate and lightweight model [14]. Then, there is an improved model of YOLOv3 by using four detection scales, k-means ++ clustering, and novel transfer learning [15]. Other improvements for

Received: Feb. 21, 2022; received in revised form: May 27, 2022; accepted: July 27, 2022; available online: March 17, 2023.
*Corresponding Author

YOLO, like YOLO Nano [16], YOLO-LITE [17], YOLOv3-Lite [18], and YOLOv3 Tiny [19], make it more suitable for low computational devices. However, these methods are never really tested on low computational devices. Another method, ResNet, uses skip connections to solve the vanishing gradient problem in Deep Neural Networks [20]. With this method, ResNet produces a significant accuracy. The inception of ResNet V2 produces an accuracy of 96.12% in detecting expiry date on food label [21]. Another research also shows that ResNet-101 has 82.17% accuracy and ResNet-152 has 82.12% accuracy on the PASCAL VOC dataset [22].

The model that has been discussed before is mainly optimized to be used on high computational devices. Usually, when it comes to low computational devices, such as mobile phones, it will significantly affect performance. However, mobile phones have also become more potent for computational tasks as technology grows. The multi-core processor and dedicated GPUs help mobile phones to become more capable of handling the process of machine learning than before. Currently, many high-end mobile devices have serious computational overhead and battery drain when running a machine learning task.

There are several studies focused on low computational devices. The Tiny-YOLOv3 model is compared on several platforms, such as TensorFlow Lite, OpenCV, and SNPE on Android, and has an accuracy of 33.1% on the COCO dataset and 33.8 MB file size [23]. Another experiment tests YOLOv3 and Tiny-YOLOv3 SSD models on DJI Drone and Android with 55.3% and 33.1% accuracy with 248 MB and 35.4 MB file sizes [24]. Then, Alexnet and Googlenet are compared on the smartphone with 57.4% and 59.3% accuracy with the Art Sculpture dataset [25]. Another approach using SSD running on the smartphone to detect 3D assets is conducted by achieving 75% accuracy with a 22 MB file size [26]. Although it is possible to achieve a model for low computational devices, the research conducted by the previous researchers shows that the small file size model that is suitable for the smartphone but still has low accuracy. Hence, the accuracy needs to be improved while retaining the file size.

The research contributes to producing an accurate and compatible model for mobile devices. The current YOLOv3 model is modified to achieve the solution by implementing the ResNet18 backbone to replace the Darknet53. The model is also evaluated on mobile devices to compare the inference time and memory usage.

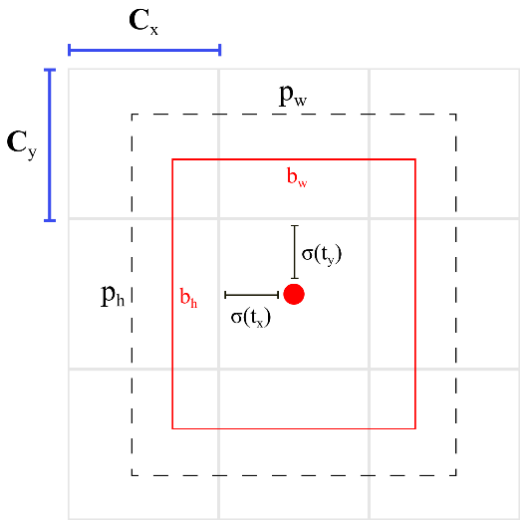

Fig. 1. Using YOLO to predict the bounding box with the anchor box by predicting the four coordinates of the bounding box (t_x, t_y, t_w, t_h). The width and height of the box are offsets of the cluster centroid.

II. Research Method

The research method carried out in the experiment is presented. The YOLOv3 model with the Darknet53 backbone network is changed into the ResNet18 backbone to perform fish detection in the image dataset. The YOLOv3 model is famous for its detection speed, and the ResNet18 model has significant accuracy while having a smaller backbone than the YOLOv3 model [20, 27].

A. You Only Look Once (YOLO)

The convolution enables to compute prediction of an object in an image in an optimized way. This solution avoids using a sliding window to compute and predict the object. On the other hand, YOLO detects the object in the image by creating a bounding box for each detected object using a single neural network instead of predicting the box of the entire image [12, 27, 28]. YOLO uses Eqs. 1–4 to predict the bounding box. Those are also shown in Fig. 1. The b_x and b_y represent the center coordinate. Then, b_w is the width, and b_h is the height of the predicted bounding box. The c_x and c_y are the top-left coordinates of the grid cell, and p_w and p_h are the anchor dimensions. The equations can be seen as follows.

$$b_x = \sigma(t_x) + c_x, \tag{1}$$

$$b_y = \sigma(t_y) + c_y, \tag{2}$$

$$b_w = p_w e^{t_w}, \tag{3}$$

$$b_h = p_h e^{t_h}. \tag{4}$$

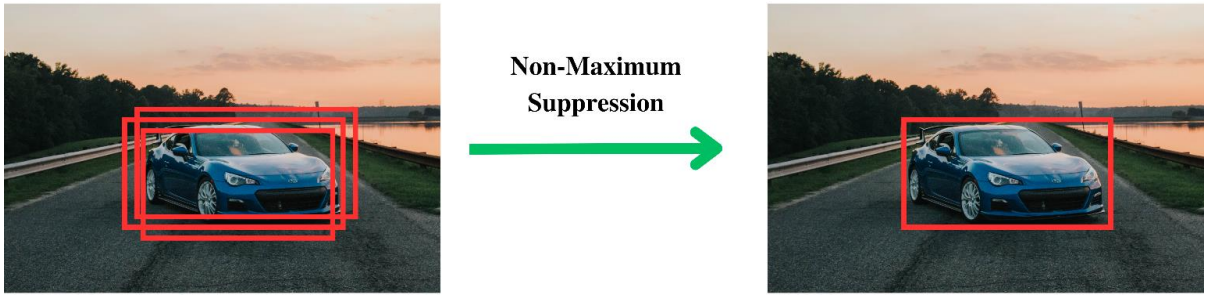

Fig. 2. Removing the bounding boxes under a certain IoU threshold and the remaining bounding boxes under a certain confidence threshold by Non-Maximum Suppression (NMS) to produce the final bounding box.

After the bounding box is predicted, the model also predicts the confidence score on the bounding box. The bounding box is then determined by assigning one predictor called ‘responsible’ to predict the object based on the highest Intersection over Union (IoU) with ground truth, as shown in Eq. 5. The value of IoU is generated by dividing the area overlap between area A, the predicted bounding box, and area B, the ground truth bounding box (A ∩ B), with the union of area A and area B (A ∪ B). However, some large objects sometimes create multiple bounding boxes. The multiple bounding boxes detected in the same object are reduced to one by using Non-Maximum Suppression (NMS) [22, 29], as shown in Fig. 2. The NMS reduces overlapping bounding boxes by comparing the IoU score to the threshold score (usually more than 0.6). If the IoU of the predicted boxes is lower than the threshold, the predicted box will be removed. After that, all the remaining boxes will be checked for the confidence score and compared to the confidence threshold score to reduce the boxes into one final predicting box.

$$\text{IoU} = \frac{A \cap B}{A \cup B} \tag{5}$$

Redmon then introduces YOLOv2 with changes of a fully connected layer to the anchor box and adds batch normalization to tackle the overfitting problem in YOLO [29]. The resolution used to detect also changes from 224×224 to 448×448. Next, the Darknet-19 frame is introduced as the neural network framework of YOLOv2, consisting of 19 convolutional layers and 5 max-pooling layers. The accuracy of YOLOv2 increases from 63.4 to 78.6 mAP. In addition, the YOLOv3 model predicts 4 bounding box coordinates by using logistic regression to predict the score for each bounding box [30]. For class prediction, YOLOv3 uses an independent logistic classifier. Finally, the Darknet53 framework is introduced to replace YOLOv2 Darknet19 [28]. Figure 3 shows Darknet53 backbone architecture as a feature extractor, mainly composed of 3×3 and 1×1 filters with the residual network to skip connections like in ResNet.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3. Darknet53 backbone architecture as a feature extractor, mainly composed of 3×3 and 1×1 filters with the residual network to skip connections like in ResNet.

B. Residual Network (ResNet)

ResNet was introduced in 2015 as a specific type of neural network [20]. The additional layers are stacked in the Deep Neural Networks to improve accuracy and performance, mostly to solve a complex problem. However, there is a maximum threshold for depth with the traditional convolutional neural network model, resulting in more errors. ResNet uses skip connections to solve the vanishing gradient problem in Deep Neural

TABLE I
RESNET ARCHITECTURE.

Layer Name	Output Size	18-Layer	50-Layer	152-Layer
Convolution 1	112×112	7×7, 64, Stride 2		
Convolution 2.x	56×56	3×3 Max Pool, Stride 2		
		$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times3$
Convolution 3.x	28×28	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times8$
Convolution 4.x	14×14	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 1024 \end{bmatrix}\times36$
Convolution 5.x	7×7	$\begin{bmatrix} 3\times3,\ 512 \\ 3\times3,\ 512 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 512 \\ 3\times3,\ 512 \\ 1\times1,\ 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1,\ 512 \\ 3\times3,\ 512 \\ 1\times1,\ 2048 \end{bmatrix}\times3$
	1×1	Average Pool, 1000-d fc, Softmax		

TABLE II
RESNET PERFORMANCE ON MOBILE DEVICES BY PREVIOUS RESEARCH

Network	Layers	Speed (ms)
ResNet18	18	31.54
ResNet34	34	51.59
ResNet50	50	103.58
ResNet101	101	156.44
ResNet152	152	217.91

Networks by allowing alternate shortcut paths for the gradient to flow through.

Inspired by VGG-19, ResNet architecture shown in Table I uses a layer of plain network architecture to which the shortcut connection is added. The architecture is mainly composed of 3×3 convolution with a fixed feature of map of 64, 128, 256, and 512, respectively and bypassing the input every two convolutions. The difference between each ResNet type (ResNet-18, ResNet-50, ResNet-152) is in convolution 2.x layer. There is an additional 1×1 convolution on each step with a different feature map (256, 512, 1024, 2048). Additionally, in ResNet-152, the number of convolutions happens in convolution 3.x and convolution 4.x doubles up to 8 layers and 36 layers.

Another previous research, as shown in Table II, compares the performance of several ResNet model families [31]. The previous research is conducted using a benchmark with a minibatch size of 16 and image size of 224×224 and running on GTX 1080 with 8 GB of memory. The speed shown in Table II is the total time for a forward and backward pass. Since ResNet18 has a smaller layer than others, the speed is relatively faster. Therefore, ResNet18 is used as the based backbone for the solution model.

C. The Model

The standard YOLOv3 model is not entirely compatible with deployed on mobile devices [27]. As a result, previous research decreases the depth of the convolutional layer called YOLOv3-Tiny [19]. The running speed significantly increases, but the detection accuracy is reduced. YOLOv3-Tiny reduces the number of convolutional layers and uses a pooling layer. Then, it divides the picture into S×S grid cells and ignores the bounding box with not the best objectness score. However, in the other method, the proposed model tested on mobile devices still achieves low accuracy [23–26]. To solve this problem, ResNet18 is used since it produces the fastest speed in previous research [31].

The YOLOv3-ResNet18 model loads the ResNet18 pre-trained model on ImageNet-1k. This model acts as a feature extraction to replace the YOLO Darknet53 backbone, as seen in Fig. 4. The method uses pre-trained model networks to initialize all layers, except the top fully connected layer whose weights are randomly initialized. As a result, the convolutional layer for feature extraction is reduced to 18 layers deep. The YOLOv3 classifier is used for the classifier layer instead of the regular ResNet18 classifier layer. Then, the result continues with detection layers with scale 1, scale 2, and the same scale with the standard YOLOv3 model. The network is trained on the fish dataset to detect and classify the fish on the images.

D. Single Shot Detector (SSD)

Proposed in 2016, SSD was a model based on a feed-forward convolutional network that produced a fix-sized bounding box, scored the class of each bounding

Cite this article as: S. Liawatimena, E. Abdurachman, A. Trisetyarso, A. Wibowo, M. K. Ario, and I. S. Edbert, “Fish Classification System Using YOLOv3-ResNet18 Model for Mobile Phones”, CommIT Journal 17(1), 71–79, 2023.

	Type	Size	Filters	Output
	Convolutional	3 x 3	32	256 x 256
	Convolutional	3 x 3 / 2	64	128 x 128
1x	Convolutional	1 x 1	32	
	Convolutional	3 x 3	64	
	Residual			128 x 128
	Convolutional	3 x 3 / 2	128	
2x	Convolutional	1 x 1	64	
	Convolutional	3 x 3	128	
	Residual			64 x 64
	Convolutional	3 x 3 / 2	256	32 x 32
8x	Convolutional	1 x 1	128	
	Convolutional	3 x 3	256	
	Residual			32 x 32
	Convolutional	3 x 3 / 2	512	16 x 16
8x	Convolutional	1 x 1	256	
	Convolutional	3 x 3	512	
	Residual			16 x 16
	Convolutional	3 x 3 / 2	1024	8 x 8
4x	Convolutional	1 x 1	512	
	Convolutional	3 x 3	1024	
	Residual			8 x 8
	AvgPool		Global	
	Connected		1000	
	Softmax			

Name		Type	ResNet-18		Output
conv1		Convolutional	7 x 7	64, stride 2	112 x 112 x 64
		MaxPool	3 x 3	stride 2	
conv2_x	2x	Convolutional	3 x 3	64	56 x 56 x 64
		Convolutional	3 x 3	64	
conv3_x	2x	Convolutional	3 x 3	128	28 x 28 x 128
		Convolutional	3 x 3	128	
conv4_x	2x	Convolutional	3 x 3	256	14 x 14 x 256
		Convolutional	3 x 3	256	
conv5_x	2x	Convolutional	3 x 3	512	7 x 7 x 512
		Convolutional	3 x 3	512	
		AvgPool	7 x 7 average pool		1 x 1 x 512
		Connected	1000		
		Softmax			

Fig. 4. Darknet53 backbone changed into the ResNet18 backbone to reduce the convolution while retaining its detection speed.

box, and used non-max suppression to produce the final bounding box [11]. SSD uses multi-scale feature maps to detect multiple scales, and the convolutional layers are added to the end truncated base network. SSD uses VGG-16 to extract feature maps from the image, followed by six convolutional layers. Conv4_3 layers are used to detect objects. SSD improvement is proposed by adding SSD to the MobileNet base network [32].

E. Huawei ExeML

ExeML automates the model design, parameter tuning, training, compression, and deployment with already labeled data [33]. The process does not require the developer to code anything. It optimizes the model to use in the mobile device to produce the model with low inference time but still has decent accuracy. The developer only needs to upload the dataset and set the label for training. Then, Huawei processes the dataset with the auto-training feature and generates the model. Then, the model can be deployed to be used as API or downloaded to be embedded on mobile devices.

F. Dataset

The dataset used in the experiment consists of 4,000 images with JPEG image encoding. The dataset is classified into four classes: Katsuwonus Pelamis, Euthynnus Affinis, Coryphaena Hippurus, and Loligo Chinensis. The examples are in Fig. 5. The resolution for the image varies from low-resolution images (303×166) in the Katsuwonus Pelamis class to high-resolution images (1300×1011). Image dataset is gathered from scraping across the Internet and taken from local market and harbor. The image is then split into

Fig. 5. The dataset collected from scraping on the Internet and taken from local markets and harbors. There are four classes: Katsuwonus Pelamis, Euthynuss Affinis, Coryphaena Hippurus, and Loligo Chinensis. Left image is Coryphaena Hippurus, and right image is Katsuwonus Pelamis.

80% for training, 10% for validation, and 10% for evaluation.

III. Results and Discussion

The dataset is prepared by giving the label to all images. In the research, all the labeling data and training processes are conducted on the Huawei cloud using respective labeling and training features. The specification used for training is NVIDIA-V100 32GB GPU, 8vCPUs, and 64GB RAM. Then, the model is trained with 2,000 epochs, 32 batch sizes, and 0.0001 learning rates. Other models used to compare YOLOv3-ResNet18 model performance are SSD-VGG and Huawei ExeML. All models are also trained in the same hyperparameter. In Table III, the YOLOv3-ResNet18 model has a smaller file size than other models. This small size is suitable for mobile devices.

In Table IV, the YOLOv3-ResNet18 model shows better accuracy at 98.447%. In precision, Huawei ExeML performs 0.198% better than the YOLOv3-ResNet18 model. However, the YOLOv3-ResNet18 model performs better on recall with 96.52% and an

Cite this article as: S. Liawatimena, E. Abdurachman, A. Trisetyarso, A. Wibowo, M. K. Ario, and I. S. Edbert, "Fish Classification System Using YOLOv3-ResNet18 Model for Mobile Phones", CommIT Journal 17(1), 71–79, 2023.

TABLE III
TRAINING RESULTS.

Model	Training Duration	File Size
YOLOv3-ResNet18	11 hours, 21 minutes	63.10 Mb
SSD-VGG	12 hours, 3 minutes	234.97 Mb
Huawei ExeML	3 hours, 34 minutes	123.70 Mb

TABLE IV
COMPARISON OF TRAINING PERFORMANCE.

Model	Accuracy	Precision	Recall	F1 Score
YOLOv3-ResNet18	98.45%	98.45%	96.52%	97.48%
SSD-VGG	78.55%	77.37%	70.42%	77.39%
Huawei ExeML	93.99%	98.65%	89.11%	93.63%

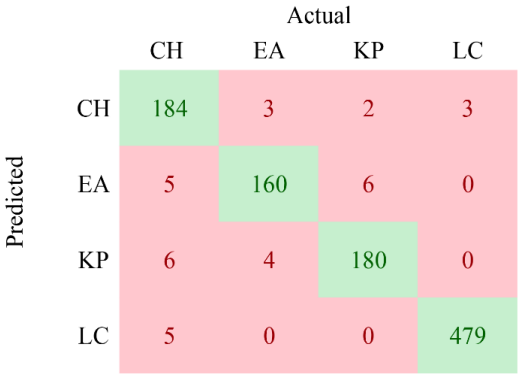

Fig. 6. Confusion matrix result of YOLOv3-Resnet18 model.

Fig. 7. Confusion matrix result of Huawei ExeML model.

TABLE V
THE EVALUATION OF YOLOV3-RESNET18 MODEL.

Class	Accuracy	Precision	Recall	F1 Score
Coryphaena Hippurus	0.97	0.98	0.92	0.94
Katsuwonus Pelamis	0.98	0.94	0.96	0.95
Euthynnus Affinis	0.98	0.95	0.96	0.95
Loligo Chinensis	0.99	0.99	0.99	0.99

TABLE VI
THE EVALUATION OF HUAWEI EXEML MODEL.

Class	Accuracy	Precision	Recall	F1 Score
Coryphaena Hippurus	0.98	0.99	0.93	0.96
Katsuwonus Pelamis	0.98	0.95	0.91	0.93
Euthynnus Affinis	0.99	0.96	0.98	0.97
Loligo Chinensis	0.96	0.96	0.99	0.98

F1 score of 97.48%. On the other hand, the SSD-VGG model has the largest file size and performs poorly with only 78.553% accuracy.

Next, the multiclass confusion matrix is used to evaluate the models. The confusion matrix is represented in 4×4 matrix form since the dataset consists of four classes. Then, the confusion matrix is calculated to find the evaluated models' accuracy, precision, recall, and F1 score. First, the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are calculated from the matrix results. Then, the accuracy, precision, recall, and F1 score are also calculated.

Figures 6 and 7 show the confusion matrix result of YOLOv3-Resnet18 and Huawei ExeML models. Both models have similar results in detecting and predicting the four datasets. Both models perform the same correct prediction on Loligo Chinensis datasets with 479 correct detections. Interestingly, Huawei ExeML model have tendencies to detect the fish as Loligo Chinensis.

Both results are calculated to get the accuracy, precision, recall, and F1 score to understand the matrix more deeply. The result shown in Table V is the evaluation of the YOLOv3-ResNet18 model. The model falls on recall of Coryphaena Hippurus with 0.92. For comparison, based on Table VI, the Huawei ExeML model has an overall accuracy of 0.97. Since the model will be used mainly to detect Katsuwonus Pelamis and Euthynnus Affinis, the YOLOv3-ResNet18 model performs better in accuracy, precision, and recall with an average score of 0.98, 0.97, and 0.96, respectively. Meanwhile, the Huawei ExeML model only has 0.97 accuracy, 0.96 precision, and 0.95 recall.

The test continues by deploying the model on the Huawei Cloud server as Application Programming Interface (API). This step ensures the model can be run and deployed on mobile devices. The mobile device sends the image to the cloud server to be processed by the model. Then, the model produces a result (Fig. 8) with returning predicted bounding boxes. These bounding boxes are then displayed on mobile devices.

For the last evaluation process, the models that have already been downloaded are implemented into mobile devices. For this experiment, the operating system is Android with two devices for comparison: Huawei P40 and Realme 7. To achieve the implementation, the models are converted into TensorFlow Lite (tflite) extension before being loaded by the application. The

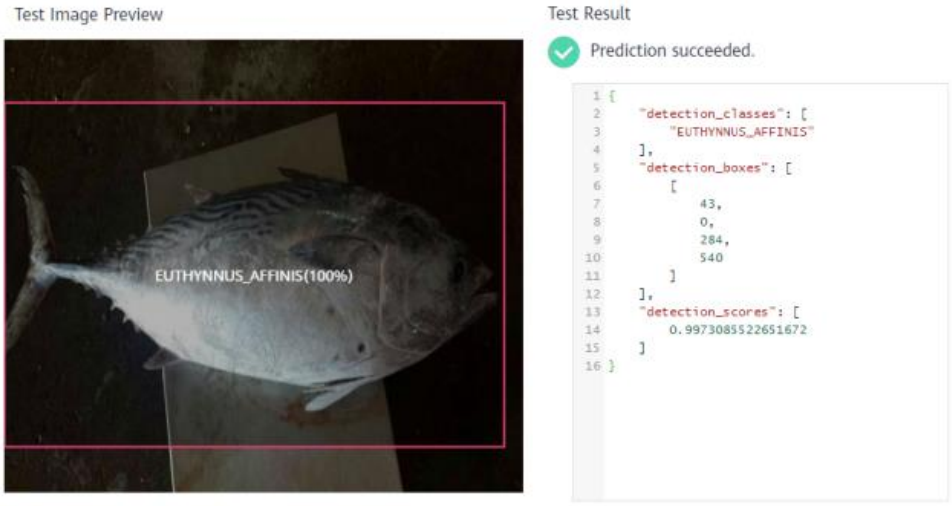

Fig. 8. The result of the test in the cloud and running the model via API.

TABLE VII
CHIPSET COMPARISON USED IN EVALUATION.

Specification	Kirin 990	Helio G95
Core	8	8
CPU Frequency (MHz)	2,860	2,050
GPU Frequency (MHz)	900	700
FLOPS (GFLOPS)	896	195.8
GPU Execution Unit	16	4
NPU	Da Vinci	Da Vinci
TDP	6 W	6 W
AnTuTu	488,442	294,929

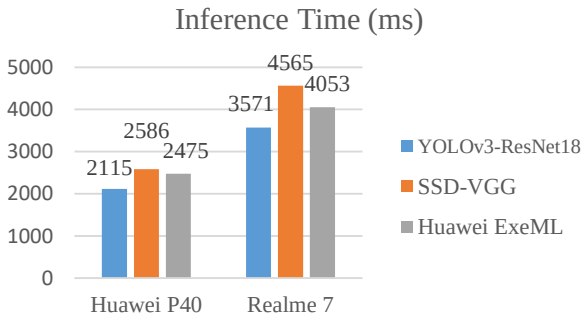

Fig. 9. Inference time comparison of three models in mobile devices.

complete specification of both processors is shown in Table VII. Huawei P40 uses Kirin 990 chipset, and Realme 7 has Helio G95 chipset. In terms of specification, Kirin 990 has significantly better specification than Helio G95. The GPU execution unit of Kirin 990 with 16 units makes this chipset have more processing units leading to better performance. It is also supported by the FLOPS of Kirin 990 with 896 Giga FLOPS compared to Helio G95 with only 195.8 Giga FLOPS.

After the model is implemented in mobile devices using the Huawei ML Kit library. The benchmark process is used to determine the inference time of the models. For this process, the benchmark uses TensorFlow Lite performance measurement tools. This test produces the inference time and overall memory usage of the models. The result of the test is shown in Figs. 9 and 10.

In Fig. 9, the YOLOv3-ResNet18 model has the fastest inference time on both tested devices. Meanwhile, in Fig. 10, Huawei ExeML has the lowest memory usage on Huawei P40 devices but increases to 44 Mb in Realme 7. Overall, YOLOv3-ResNet18 has better inference time and memory usage on both tested devices. Figures 9 and 10 also show that all models perform better on Huawei P40. The chipset used by both devices causes the result. Huawei P40 has a Kirin 990 chipset, while Realme 7 has an Helio G95. Both chipsets have Da Vinci NPU. However, Kirin 990 has better Floating Point Operations Per Second (FLOPS) with 896 GFLOPS. FLOPS is used to measure computer performance. It leads to Huawei P40 having a more significant inference time than Realme 7 in all models.

IV. CONCLUSION

Mobile device technology grows rapidly and has better chipsets to compute more powerful tasks. As a maritime country dominated by small-scale industries with traditional boats, Indonesia needs technology implementation to determine the type of fish they capture and the total weight of captured fish. The research

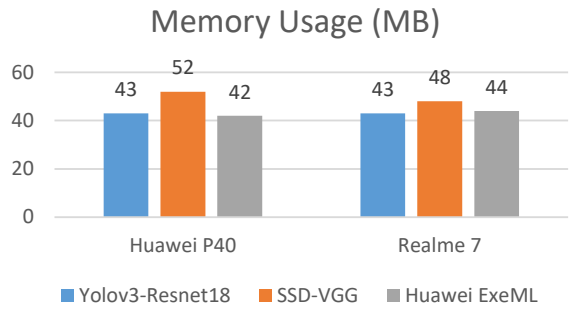

Fig. 10. Memory usage comparison of three models in mobile devices.

conducts an experiment to find a suitable model for mobile devices. YOLOv3, known for its good accuracy and fast detection speed combined with the Resnet-18 backbone, results in a smaller-size model suitable for mobile devices while maintaining good accuracy and precision. The experiment also shows that the model uses a small RAM size after being deployed on mobile devices and has a reasonable inference time. The experiment will be continued to make inference time faster and predict the weight of fishes captured in the future.

The current experiment only compares the models on two mobile devices. Meanwhile, the mobile devices used in the market have more various chipsets. So, the model can have a significantly different performance from other chipsets. For future experiments, the model will be deployed on various mobile devices to understand the impact of chipsets on the model inference time. In addition, more models will be compared to get the most suitable models for mobile devices.

References

[1] Food and Agriculture Organization of the United Nations, *The state of world fisheries and aquaculture 2020: Sustainability in action*. Food and Agriculture Organization of the United Nations, 2020.

[2] K. Kusdiantoro, A. Fahrudin, S. H. Wisudo, and B. Juanda, "Kinerja pembangunan perikanan tangkap di Indonesia," *Buletin Ilmiah Marina Sosial Ekonomi Kelautan dan Perikanan*, vol. 5, no. 2, pp. 69–84, 2019.

[3] J. Lee and W. Hwang, "Cloud-based facial expression recognition system for customer satisfaction in distribution sectors," *ICIC Express Letters, Part B: Applications*, vol. 11, no. 2, pp. 173–179, 2020.

[4] B. W. Yoon, E. Genc, O. F. Ince, and M. E. Yildirim, "Human activity recognition using inter-joint feature fusion with SVD," *ICIC Express Letters, Part B: Applications*, vol. 12, no. 3, pp. 215–221, 2021.

[5] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, 2018.

[6] D. I. Patrício and R. Rieder, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," *Computers and Electronics in Agriculture*, vol. 153, pp. 69–81, 2018.

[7] Y. D. Zhang, Z. Dong, X. Chen, W. Jia, S. Du, K. Muhammad, and S. H. Wang, "Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation," *Multimedia Tools and Applications*, vol. 78, pp. 3613–3632, 2019.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[9] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea, Oct. 27–Nov. 2, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, Montreal, Canada, Dec. 7–12, 2015, pp. 91–99.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision–ECCV 2016*. Amsterdam, The Netherlands,: Springer, Oct. 11–14, 2016, pp. 21–37.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[13] H. Yang, P. Liu, Y. Hu, and J. Fu, "Research on underwater object recognition based on YOLOv3," *Microsystem Technologies*, vol. 27, pp. 1837–1844, 2021.

[14] K. Cai, X. Miao, W. Wang, H. Pang, Y. Liu, and J. Song, "A modified YOLOv3 model for fish detection based on MobileNetv1 as backbone," *Aquacultural Engineering*, vol. 91, pp. 1–9, 2020.

[15] K. Raza and H. Song, "Fast and accurate fish detection design with improved YOLO-v3 model and transfer learning," *International Journal of*

Advanced Computer Science and Applications, vol. 11, no. 2, pp. 7–16, 2020.

[16] A. Wong, M. Famuori, M. J. Shafiee, F. Li, B. Chwyl, and J. Chung, "YOLO nano: A highly compact you only look once convolutional neural network for object detection," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, pp. 22–25.

[17] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A real-time object detection algorithm optimized for non-GPU computers," in *2018 IEEE International Conference on Big Data (Big Data)*. Seattle, USA: IEEE, Dec. 10–13, 2018, pp. 2503–2510.

[18] H. Zhao, Y. Zhou, L. Zhang, Y. Peng, X. Hu, H. Peng, and X. Cai, "Mixed YOLOv3-LITE: A lightweight real-time object detection method," *Sensors*, vol. 20, no. 7, pp. 1–18, 2020.

[19] P. Adarsh, P. Rathi, and M. Kumar, "YOLO v3-Tiny: Object detection and recognition using one stage improved model," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Coimbatore, India: IEEE, March 6–7, 2020, pp. 687–694.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, United States, June 26–July 1, 2016, pp. 770–778.

[21] W. Y. Ong, C. W. Too, and K. C. Khor, "Transfer learning on inception ResNet V2 for expiry reminder: A mobile application development," in *International Conference on Mobile Web and Intelligent Information Systems*. Virtual (Online): Springer, Aug. 23–25, 2021, pp. 149–160.

[22] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the ResNet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[23] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Smartphone-based real-time object recognition architecture for portable and constrained systems," *Journal of Real-Time Image Processing*, vol. 19, no. 1, pp. 103–115, 2022.

[24] I. Martinez-Alpiste, P. Casaseca-de-la Higuera, J. Alcaraz-Calero, C. Grecos, and Q. Wang, "Benchmarking machine-learning-based object detection on a uav and mobile platform," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. Marrakesh, Morocco: IEEE, April 15–18, 2019, pp. 1–6.

[25] L. Tobías, A. Ducournau, F. Rousseau, G. Mercier, and R. Fablet, "Convolutional neural networks for object recognition on mobile devices: A case study," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. Cancun, Mexico: IEEE, Dec. 4–8, 2016, pp. 3530–3535.

[26] R. Kostoeva, R. Upadhyay, Y. Sapar, and A. Zakhor, "Indoor 3D interactive asset detection using a smartphone," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 811–817, 2019.

[27] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[28] ——, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[29] Y. Song, Q. K. Pan, L. Gao, and B. Zhang, "Improved non-maximum suppression for object detection using harmony search algorithm," *Applied Soft Computing*, vol. 81, pp. 1–13, 2019.

[30] G. Wu and Y. Li, "Non-maximum suppression for object detection based on the chaotic whale optimization algorithm," *Journal of Visual Communication and Image Representation*, vol. 74, pp. 1–8, 2021.

[31] J. Johnson, "CNN benchmarks," 2017. [Online]. Available: https://github.com/jcjohnson/cnn-benchmarks#readme

[32] P. D. Hung and N. N. Kien, "SSD-Mobilenet implementation for classifying fish species," in *Intelligent Computing and Optimization: Proceedings of the 2nd International Conference on Intelligent Computing and Optimization 2019 (ICO 2019)*. Springer, 2020, pp. 399–408.

[33] HUAWEI CLOUD, "Introduction to ExeML," 2023. [Online]. Available: https://support.huaweicloud.com/intl/en-us/exemlug-modelarts/modelarts_21_0001.html