OPTIMIZING YOLOV4 FOR FISH RECOGNITION

ARI KUSWANTORI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2023

KMITL-2023-EN-D-018-043

Thesis title	Optimizing YOLOv4 for Fish Recognition
Student name	Mr. Ari Kuswantori
Student id.	63601247
Degree	Doctor of Engineering
Program	Electrical Engineering
Year	2023
Advisor	Prof. Dr. Worapong Tangsrirat
Co-advisor	Assoc. Prof. Dr. Taweepol Suesut

ABSTRACT

Automatic fish detection and classification (fish recognition) using computer or machine vision are popular and intriguing research areas. Numerous researchers are engaged in its development for underwater and out-of-water environments. Recognizing fish in underwater conditions, especially in deep ocean environments, is advantageous for fish population control and sustainability. Recognizing fish in settings other than water, especially for farming fish, is useful in aquaculture, for example, for automatic sorting processes, monitoring fish quality, and other activities. Fish recognition has many challenges, including diverse background conditions, similar appearance and deformed fish, which is usually found in farmed fish cases, also the speed and random position for recognizing moving fish. Many methods have been developed, and deep learning is the most utilized, such as algorithms based on CNN, AlexNet, ResNet, Inception, YOLO, etc. Many fish datasets are also openly available and accessible to the public. YOLO is the most popular object detection algorithm known to be effective and rapidly developed today. However, it is still very rarely developed for fish recognition. In addition, there is no public dataset available on moving fish in the aquaculture industry, which is very useful for developing computer or machine vision for fish process automation. This thesis presents YOLO's optimization works for fish recognition with those various challenges. The YOLO version used is version 4 (YOLOv4), which is a relatively new version. In addition, we also created a new dataset consisting of videos of farmed

fish moving randomly on a conveyor, which is very useful for developing automation in the fish industry based on computer or machine vision, especially for automatic sorting systems. These conducted studies are expected to contribute to automatic fish recognition, which can be applied underwater for fish population control or in the aquaculture industry to develop fish process automation.

ACKNOWLEDGEMENT

Praise and gratitude, I pray to Allah SWT for the strength and determination so that I was able to complete this thesis. *Sholawat* and invocation may continue to be bestowed upon the Messenger of Allah, Muhammad SAW, who has guided us from the path of ignorance to the path of knowledge.

I want to express my deepest thankfulness to KMITL, who has given me the opportunity and scholarship to study doctoral program in the Instrumentation and Control Engineering Department, School of Engineering. I also express my deepest thanks to Prof. Dr. Worapong Tangsrirat and Assoc. Prof. Dr. Taweepol Suesut, as my supervisors who have guided and delivered much knowledge to compile this thesis and complete my study. I deliver my thankfulness too to other lecturers; Assoc. Prof. Dr. Navaphattra Nunak and Asst. Prof. Dr. Jedsada Chaishhome.

I also thank the committee of examiners of this thesis, who have examined and suggested many inputs so that this thesis can be better. To the College of Advanced Manufacturing Innovation (AMI)- KMITL (Assoc. Prof. Dr. Siridech Boonsang, Mr. Teerawat Tongloy, and team), I thank you very much for providing support by giving licenses and instructions for using the CiRA-CORE software. It is very useful for this thesis. Special thanks to Lung Nan Pare Pla, the fish shop at Bang Khla Farmer Market, 15 Rai Moo 3, Samed Nua, Bang Khla, Chachoengsao, Under the Ministry of Interior of Thailand, who has donated his fish for free for the dataset we created.

Not to forget, I also thank my colleagues in the laboratory (Kaung Myat Naing, Chanitnunt Sanbooranapunt, Anchalee Kraisalee, Badin Somlitsopak, Tanundorn Veng, Ekarin Ongwongsakul) and comrades in arms from Indonesia (Mr. Dwi Joko Suroso, Mr. Agustami Sitorus, etc.). They have helped a lot in my research, writing this thesis, and other study activities. I also thank all my friends who have helped, which I cannot mention one by one. Finally, I dedicate this thesis to my wife, parents, and beloved family. Thank you for always supporting my dreams. I love you so much.

Ari Kuswantori

TABLE OF CONTENTS

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

AP	Average Precision
BB, bb, bbox	Bounding Box
BLOB	Binary Large Object
CNN	Convolutional Neural Network
CPU	Central Processing Unit
Fast R –CNN	Fast Region Convolutional Neural Network
Faster R-CNN	Faster Region Convolutional Neural Network
FC	Fish classification
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GMM	Gaussian Mixture Models
GPU	Graphical Processing Unit
LSTMs	Long Short-Term Memory units
NBF	Naive Bayesian Fusion
OR	Occupancy Ratio
PAN	Path Aggregation Network

RAM	Random Access Memory
R-CNN	Region Convolutional Neural Network
RPNs	Region Proposal Network
SDV	Standard Deviation Value
SPP	Spatial Pyramid Pooling layer
SSD	Single Shot Detector
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Research Background

Automatic fish detection and classification (fish recognition) using computer or machine vision are popular and intriguing research areas. Numerous researchers are engaged in its development for underwater and out-of-water environments [1-6]. Recognizing fish in underwater conditions, especially in deep ocean environments, is advantageous for fish population control and sustainability [7-12]. Recognizing fish in settings other than water, especially for farming fish, is useful in aquaculture to develop automation to boost productivity. For example, for automatic sorting processes, monitoring fish quality, and other activities [3, 5, 6, 13-18]. Both are very important to support the solutions to anticipate the impact of global warming and climate change: the threat to wild fish populations and their sustainability, and the food scarcity, that is worried to occur in the future [9, 12, 19-21].

From the current state of the arts, it can be summarized that automatic fish recognition using a computer or machine vision has several challenges, including diverse background conditions, similar appearance and deformed fish, which is usually found in farmed fish cases [22, 23], also the speed and random position for recognizing moving fish, in the case of recognizing freely swimming fish in the ocean [7-9, 12, 20, 24] or fish run on conveyors [13].

Until now, fish recognition technology continues to be developed along with the development of computer hardware, the technology of cameras, image processing, and recognition algorithms. Object recognition technology has developed very rapidly since the invention of the GPU (Graphical Processing Unit). The training and testing process can be carried out much faster than before, using only the CPU (Central Processing Unit) [25]. Today, camera technology is also developed for better image capture [1]. Image processing techniques that are constantly being developed are also very helpful in conditioning the image before entering the recognition algorithm. It can greatly increase the effectiveness and efficiency of the algorithm's performance [23].

Supervised learning algorithms (deep learning) are generally used as the main pillar in fish recognition cases that are no longer simple. It is nowadays widely developed [1, 2]. The algorithms used are vary, starting from the earliest algorithms developed, such as CNN (Convolutional Neural Network) [7, 8, 26-28], algorithms that were developed later and are often used, such as Region-CNN (R-CNN) [29, 30], Fast R –CNN [31], Faster R-CNN [32-35], few-shot learning for limited training images [10], SSD (Single Shot Detector) [36-38], Alex-Net [11, 22, 39-41], VGGNet [42-45], ResNet [46-48], Inception [22, 32], GoogLeNet [43, 49, 50], etc., up to the most advanced and popular algorithms today such as YOLO (You Only Look Once) [9, 15, 23, 51].

Many fish datasets are also openly available and accessible to the public, such as the Fish4Knowledge repository, which contains video of underwater fish, or BYU and Fish-Pak, which contain static fish images of marine and farmed fish. However, there is no dataset of farmed fish moving on the conveyor which is very useful for the development of automatic moving fish recognition in the fish industry.

1.2 Thesis Objectives

As previously reviewed, deep learning algorithms are widely used in fish recognition, including YOLO. YOLO is a popular object recognition algorithm massively developed today because of its speed and accuracy. YOLO is considered the most powerful object detection algorithm, considerably utilized in many applications, including fish recognition. However, only a few studies have been conducted on fish recognition using YOLO, and many are not optimized yet [1].

This thesis aims to employ and optimize the YOLO to meet the typical and common challenges in fish recognition, which have been mentioned above. The YOLO used is the relatively new: YOLO version 4 (YOLOv4). In addition, a dataset of farmed fish moving on the conveyor will be created, and an approach to recognize it will be proposed. It is very important to develop automatic moving fish recognition in the aquaculture industry, especially for automatic sorting systems.

From all the works on this thesis, it is expected that this thesis will be able to optimize and increase the accuracy of fish recognition using YOLOv4 to overcome the challenges of similar visual conditions, physically deformed, varied backgrounds, and randomly moving fish positions. That way, it can be applied to fish recognition both for underwater and for the fish industry.

1.3 Main Contributions

The major contributions of this thesis are listed below:

1. Propose an approach to optimize and conduct a study about YOLOv4 for fish classification in similar appearance and structurally deformed conditions with some techniques.
2. Offer a solution to optimize YOLOv4 for fish classification in various background conditions with a simple method.
3. We present the dataset of aquaculture fish running on a conveyor. To the author's best knowledge, it is the first publicly available, which is very useful for the automatic farm moving fish recognition development based on deep learning and image processing.
4. We suggest a fish detection and classification method for our created dataset with the optimized YOLOv4. The method is simple but effective and developed as ready to use solution for an automatic fish sorting system.

1.4 Thesis Outline

The organization of this thesis is managed as follow:

Chapter 1 consists of the research background, thesis objectives, main contributions, and the thesis outline breakdown.

Chapter 2 presents the work in optimizing YOLOv4 for fish classification in similar and structurally deformed conditions, including why the research was conducted, the review of recent similar works by another researchers, explanation about material and method used and proposed, the experimental results, discussion, and the work summary.

Chapter 3 explains the work in optimizing YOLOv4 for fish classification in various background conditions. The explanations start from the research background, the review from recent state of arts, the fish dataset used and the method proposed, the experimental results, discussion, and the work summary.

Chapter 4 writes the work of optimizing YOLOv4 for moving fish detection and classification. It intended for automatic fish sorting system in fish or aquaculture

industry. This capter started by the introduction and the recent published works about moving fish recognition. The next section explains about the dataset used, which originally created by the author. The proposed method is then explicated, and followed by the experimental results, discussion, and the summary.

Chapter 5 is the conclusions and the recomendations.

Chapter 2
Optimizing YOLOv4 For Fish Classification In Similar And Structurally Deformed Conditions

2.1 Introduction

Fish Classification (FC) in the fish industry, which is part of the aquaculture industry, is very important to develop automation in the process in that industry. Automatic FC using a computer or machine vision is needed to develop automation for automatic sorting systems, inspection, or other processes [26]. FC in the fish industry has several unique challenges, including the appearance of similar fish between one species and another and the physical condition of fish that has been deformed or damaged, such as eyes, fins, or scales that are damaged with mild, moderate, to heavy level [13].

Until now, FC technology for the fish industry continues to be developed. Many approaches have been proposed to achieve high accuracy, efficient computation, and low cost. Many techniques are also used, such as image processing, supervised learning, unsupervised learning, or transfer learning [1-3]. Public datasets are also widely available, such as Fish-Pak, which consists of 6 species of freshwater fish which are very suitable for research development in FC, especially for the fish industry [52].

Research [13] was conducted to answer the challenge of classifying similar fish between one type and another. In addition, the fish were also physically deformed. In this research, the classic AlexNet algorithm was reported to produce less than optimal accuracy when applied to classify fish in the Fish-Pak dataset. To that end, an approach was proposed. The approach utilized several image processing techniques and segmentation and combined the classic AlexNet algorithm with Naive Bayesian Fusion (NBF).

[48] proposed an approach based on SE-ResNet152 and class-balanced focal loss to classify physically deformed and similar fish with a small and unbalanced amount of input data. This research was conducted and tested on the same dataset (Fish-Pak). The SE-ResNet152 model was constructed as a generalized feature extractor and was migrated to the target dataset after visualization analysis and image

preprocessing of the fish dataset were carried out first. Finally, the class-balanced focal loss function was applied to train the SE-ResNet152 model and realize fish species classification on every fish image's body, head, and scale.

[44] built a CNN with 32 layers to classify fish with the same challenges. It was done by modifying the standard VGGNet network by adding four convolutional layers to the training of each level in the network. The proposed approach was trained and tested on the Fish-Pak dataset as well. [53] used classic CNN to classify six types of fish in the same dataset (Fish-Pak). Another research is [54], which proposed the Deep Convolutional Autoencoder to classify the three very similar carp types. They created their dataset consisting of 1,500 images of India's three most common types of carp. Of all that work, they implemented augmentation to enrich the input data. Richer input data can increase the effectiveness of the deep learning algorithms and help solve problems with damaged or deformed fish conditions. After that, they utilized and developed deep learning algorithms to solve the problem of classifying similar fish.

In this work, a very different method is proposed, namely with YOLO, which is rarely applied to classify fish, especially with similar conditions and physically deformed. The YOLO used is a relatively new version (version 4) and will be optimized with various techniques. The impact of combining these various techniques will be studied on accuracy and other performance results. This research on the classification of fish species with YOLO is very important as a base for developing the detection and classification of fish that move automatically, such as on conveyors, which is very important for automatic sorting, inspection, or other processes in the fish industry.

2.2 Material and Methods

2.2.1 Fish Dataset and Image Augmentation

This work aims to classify fish with similar conditions and physically deformed. So Fish-Pak dataset [52] is considered very appropriate for use. This dataset consists of six fish species, including:

1. *Catla* (Thala),
2. *Hypophthalmichthys molitrix* (Silver),
3. *Labeo rohita* (Rohu),
4. *Cirrhinus mrigala* (Mori),

5. *Cyprinus carpio* (Common carpio) and
6. *Ctenopharyngodon Idella* (Grass carp).

These fish are commonly bred in South Asia, such as India and Pakistan [44], and some types can also be found in other regions, such as Southeast Asia. Image data on each fish species, consisting of images of the whole body, head only, and scales only. There are 271 images of the body, 254 images of the scale, and 390 images of the head, so that the total fish images in this dataset are 915 images.

Some types of fish are very similar and very difficult to distinguish by ordinary people. In addition, many fish have structural deformation conditions such as the eyes and scales that were damaged lightly, moderately, and severely until the contents of the stomach came out. These reasons make this dataset to be considered quite ideal for use in this work. Examples of fish images and its structural deformation condition in this dataset can be considered in Figure 2.1. Figure 2.2 shows examples of body, head, and scale images for each class and Table 2.1 contains details of the number of images in each class.

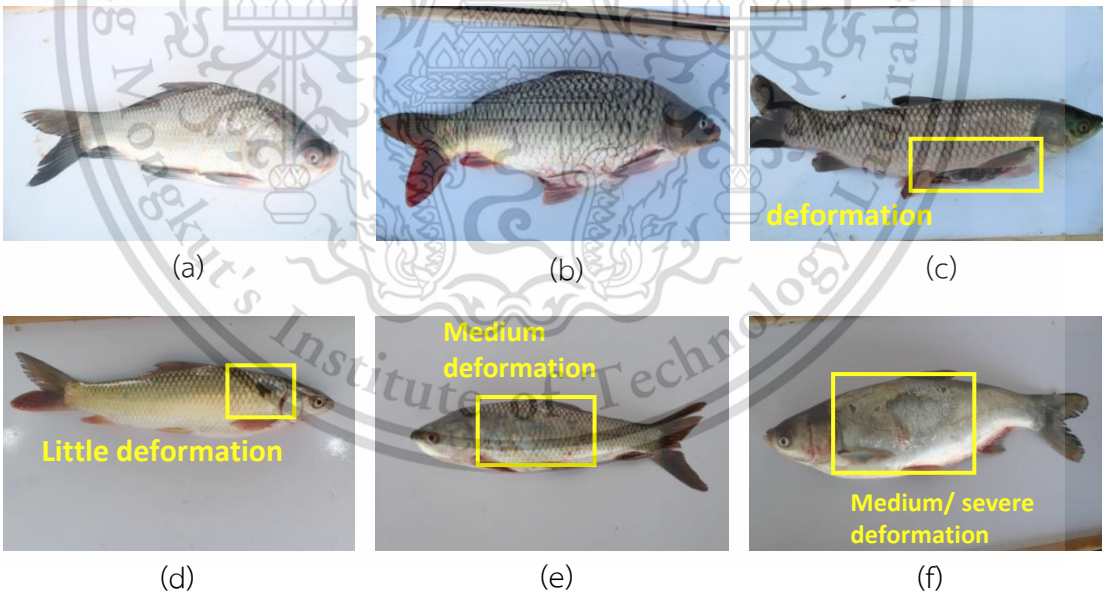


Figure 2.1 Example images from Fish-Pak dataset: (a) Catla; (b) C. Carpio; (c) G. Carp; (d) Mori; (e) Rohu and (f) Silver [52]

Figure 2.2 Example images for body, head, and scale for every classes: (a) Catla; (b) C. Carpio; (c) G. Carp; (d) Mori; (e) Rohu and (f) Silver [48]

Table 2.1 Detailed number of images in Fish-Pak dataset [52]

No.	Classes	Body	Head	Scale	Total
1	Catla	20	25	11	56
2	C. Carpio	50	64	44	158
3	G. Carp	11	16	9	36
4	Mori	70	100	71	241
5	Rohu	73	114	62	249
6	Silver	47	71	57	175
	Total	271	390	254	915

From the Table 2.1 we can observe that the number of images for each fish species in the Fish-Pak dataset is minimal (less than 100) and not balanced between one species and another. Little image data for each class type makes validation low, and unbalanced data makes the algorithm get unequal opportunities in the training process for each class type. For this purpose, image augmentation is required . The augmentation technique used in this work is flip, rotation, and translation, which is refered for fish classification [13]. The augmentation process for the images for each classes are described in (2.1) untill (2.4).

$$I_f^C = \left\{ I_f^{1C}, I_f^{2C}, ..., I_f^{N_C C} \right\}, \tag{2.1}$$

$$m_f = \left\{ m_f^C = \left| 1 - \frac{N_T}{N_C} \right|, C \in [1,2,...C], N_C < N_T \right\}, \quad (2.2)$$

$$I_{af}^{nC} = \left\{ H\left(I_f^{nC}, F_a, \theta_a, T_a \right) \middle| a \in \left[1,2,...,m_f^C\right] \middle| \right\}, \quad (2.3)$$

$$I_{Nf}^C = \left\{ I_f^{1C}, I_f^{2C}, ..., I_f^{N_C C}, I_{af}^{1C}, I_{af}^{2C}, ..., I_{af}^{N_C C} \right\}. \quad (2.4)$$

Initially, each class (*C*) of fish (*f*) images (*I*) (I_f^C) is selected with the number of images (N_c) as in (2.1). Multiplication factor (m_f^C) is obtained by comparing the target (N_T) with the number of images in each class (N_C). The target (N_T) is set at 100 for each class in this work. The value of the multiplication factor must be positive and rounded up as integers, where $N_C < N_T$. The set of multiplication factors (m_f) is obtained by repeating the procedure for each class of fish as in (2.2). The multiplication factor indicates the number of augmented images that need to be created from each image (I_f^{nC}) in each class of fish. Flip (F_a), rotation (θ_a), and translation vector (T_a) are selected randomly and generate an augmented image (I_{af}^{nC}) as in (2.3). The variances of each of the flip (F_a), rotation (θ_a), and translation vectors (T_a) were predetermined. Finally, all the augmented image sets are merged with the original image set, and a new data set (I_{Nf}^C) is formed as in (2.4).

After the augmentation procedure, the new dataset has a significant number of images, ranging from 271 to 737. In addition, the average number of images in each class increased from 45 to 123. Moreover, the dataset is now more balanced, evidenced by a decrease in the standard deviation value from 23.16 to 19.87. Following that, the images were randomly separated for the training (80%) and testing (20%) processes [52]. The detail and result of the augmentation process for body only's images is resumed in Table 2.2.

In this work, the effect of adding scale and head data to the training process on the algorithm is also tested. For this reason, scale and head image data in the dataset are also used. However, the scale and head data augmented and used for training are only limited to 3 classes, namely Catla, C. Carpio, and G. Carp, because only these three classes are challenging to detect and classify by the algorithm (low

detection and classification results). Augmentation is also done so that the data is more enrich and balanced for each class. The multiplication factor is determined in the same way that new datasets can be created. For scale image data, NT is set to 27, and the head image is set to 48. All new datasets for scale and head in the three classes are used as training data. The detail and result of the augmentation process for scale and head images are resumed in Table 2.3.

Table 2.2 Dataset image detail and augmentation (body)

Fish	Number of Images (body)	Multiplication Factor	Number of Augmented Images	New dataset (Body)	Training (80 %)	Testing (20 %)
Catla	20	4	80	100	80	20
C. Carpio	50	1	50	100	80	20
G. Carp	11	9	99	110	88	22
Mori	70	1	70	140	112	28
Rohu	73	1	73	146	117	29
Silver	47	2	94	141	113	28
Total	271	–	466	737	590	147
Average	45	–	78	123	98	25
Standard Deviation Average	23.16	–	–	19.87	–	–

Table 2.3 Scale and head image data and augmentation

Fish	Scale Image	Scale & Augmented	For Training (Scale)	Head Image	Head & Augmented	For Training (Head)	Total Training w/Scale & Head
Catla	11	33	33	25	75	75	188
C. Carpio	44	44	44	64	64	64	188
G. Carp	9	27	27	16	48	48	163
Mori	71	71	0	100	100	0	112
Rohu	62	62	0	114	114	0	117
Silver	57	57	0	71	71	0	113
Total	254	294	104	390	472	187	881
Standard Deviation Average (3 classes)	16.05	7.04	-	20.83	11.09	-	-

2.2.2 YOLO and Training Process

In this work, YOLO version 4 (YOLOv4) [55] is used. YOLO *(You Only Look Once)* is a very popular, widely used, and massively developed algorithm for object detection because of its ability to detect objects quickly and accurately. It is the reason why YOLO is used in this work. In addition, YOLO is still rarely used in fish classification. YOLO was built from 24 convolution layers and two fully connected layers at the beginning of its development [56], as shown in Figure 2.3. Until now, YOLO has continued to be developed to improve accuracy and detection time, including up to YOLOv4 used in this work. The detailed architecture or the layers of the YOLOv4 are described in Appendix A.

The YOLOv4 training process in each test was carried out with a batch size of 32 with 32 subdivisions. Data enhancement was performed by rotation with a threshold between a minimum of −180° and a maximum of 180° (with 90 steps), and contrast with a threshold between a minimum of 0.4 and a maximum of 1.1 (with 0.2 steps). Data enhancement during the training process is not done by simulating noise and blur. Each training process is carried out until the average loss value becomes acceptable, i. e., up to 0.01 to a maximum of 0.03.

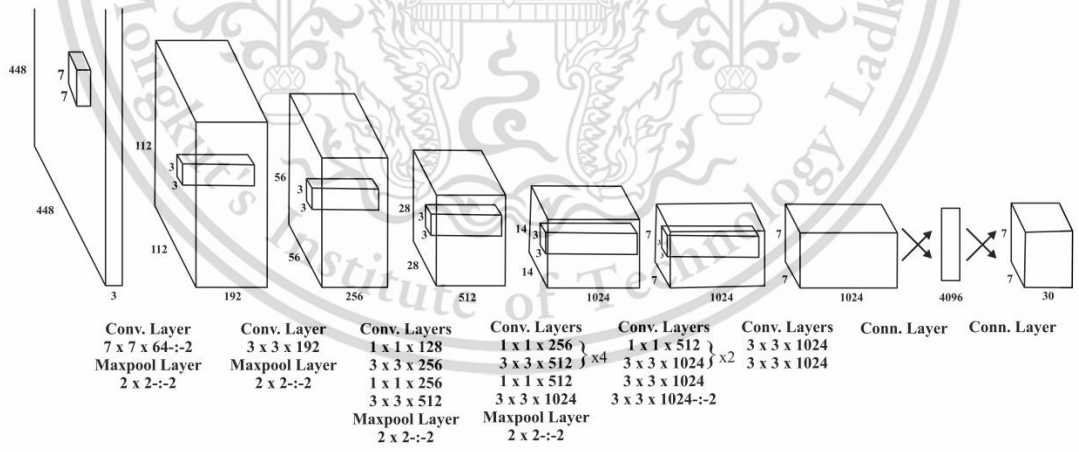


Figure 2.3 YOLO Architecture [56]

2.2.3 Optimization Techniques

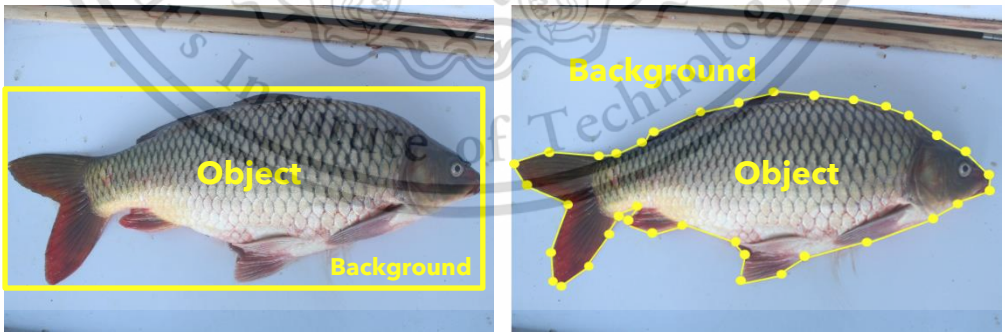
In this work, the main algoritm (YOLOv4) is combined with several techniques to to determine the impact on the final results. The techniques including: landmarking,

subclassing, adding scale data, adding head data, square labelling, and class elimination.

A. Square and Landmarking Labelling Technique

Labelling or annotation is a process that is carried out when preparing data just before training. Square-narrow-object labelling (square labelling) is the most commonly used, and landmarking is a technique that the author recently introduced in this work for fish recognition. Square labelling still includes some background, while landmarking eliminates all of it. The landmarking technique is a relatively new labelling technique currently utilized sparingly, especially on fish objects. With this technique, any part of the object can be marked and become an area that is only processed during training, so this method leads to more on-target feature extraction.

In principle, the landmarking technique will convert the object's coordinate points into a polygon and form an area on the object. Only objects or parts of objects marked as functional areas will be extracted for the training process, and anything outside the area will be ignored. Consequently, in addition to the bounding box (bbox), center point, color, label, and index label information, the output of this process includes data landmark points and landmark len, which will be used in the training process to improve its effectiveness. The difference between square and landmarking labelling is illustrated in Figure 2.4, and the concept of this landmarking technique can be explained in Figure 2.5.



(a) (b)

Figure 2.4 Labelling technique: (a) Square and (b) landmarking

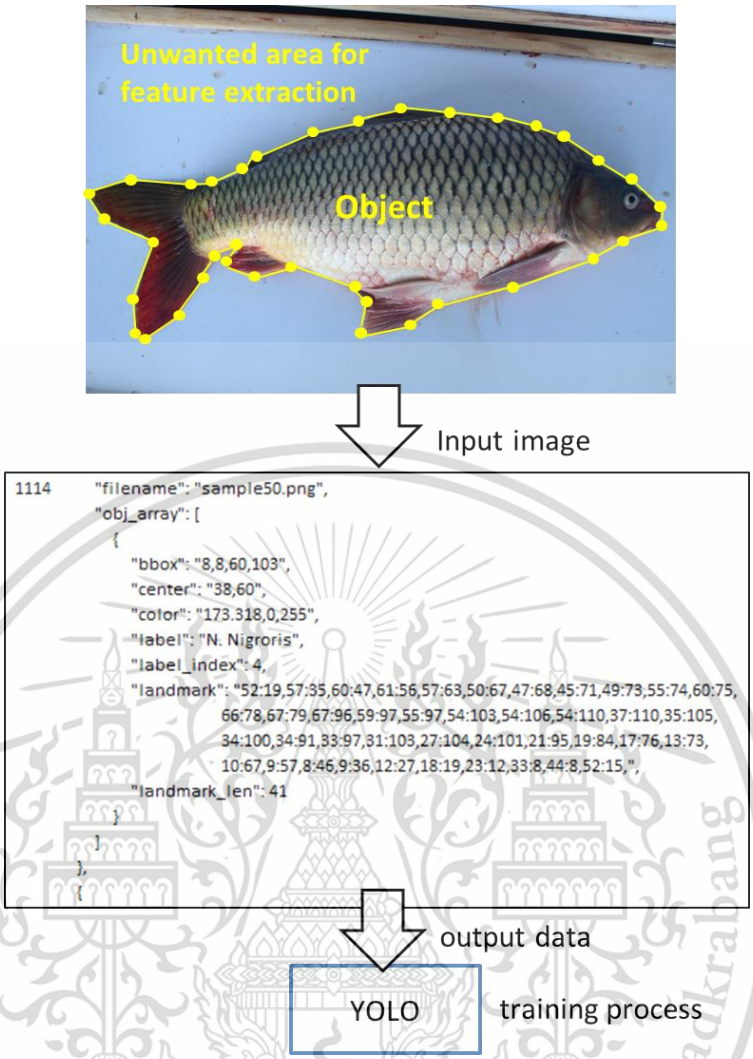


Figure 2.5 The landmarking technique's concept

B. Subclassing Technique

The subclassing technique is used when different subspecies within a class, such as different colors, patterns, or shapes, can be distinguished. As in this work, class C. Carpio has C. Carpio brown and C. Carpio red (Figure 2.6), which are in the same class (C. Carpio). For that, the class C. Carpio red will be added in this subclassing technique. This method is applied to see the impact on the recognition ability in the class and as a whole.

(a) (b)

Figure 2.6 The same class (C. Carpio) consists of different subspecies: (a) C. Carpio Brown and (b) C. Carpio Red

C. Adding Head and Scale Data

Trials were carried out by incorporating images of heads and scales into the training process. It is intended to enrich object feature references and focus on algorithm training. Head and scale images are used because each type of fish in the dataset can be distinguished from the head and scales. The head and scale images have been augmented as previously described.

D. Class Elimination Technique

Algorithm limitations in classification, are evaluated by class elimination techniques. The number of classes eliminated E_C is determined by $N_C - 1$, where N_C is the number of classes whose accuracy level cannot be accepted, as shown in (2.5):

$$E_C = N_C - 1. \quad (2.5)$$

2.2.4 Validation Matrix

A confusion matrix is used to validate the experimental results. The confusion matrix is built from 4 building blocks, namely True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). True Positive (TP) is described as when the model detects the object correctly. True Negative (TN) is described when the model does not detect an object because the object does not exist. False Positive (FP) is described when the model detects an object but is wrong, including when it detects a double, even though it has correct predictions (double prediction with the wrong class). Furthermore, False Negative (FN) is described when the model does not detect

an object even though it exists [14]. This confusion matrix can evaluate the model with accuracy and other performance parameters such as precision, recall or sensitivity, specificity, and F score.

Accuracy: is one of the evaluation metrics, which is denoted in (2.6). It is the ratio of the total number of correct predictions to the total of all predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\ \%, \tag{2.6}$$

where, *TP* – True Positives, *TN* – True Negatives, *FP* – False Positives, and *FN* – False Negatives.

Alternatively, the level of model accuracy can also be expressed by:

$$Accuracy = \frac{\sum_i^N Pi}{\sum_i^N |Qi|} \times 100\ \%, \tag{2.7}$$

where $\sum_i^N Pi$ is the number of correct predictions, and $\sum_i^N |Qi|$ is the total number of predictions.

Precision: is the ratio between correctly classified fish (*TP*) and positive detection (number of *TP* and *FP*). It calculates the percentage of fish classified accurately as:

$$Precision = \frac{TP}{TP + FP} \times 100\ \%. \tag{2.8}$$

Recall or **sensitivity**: is the ratio between the correctly classified fish (TP) and the fundamental truth fish (total number of TP and FN), which can be defined as:

$$Recall\ /\ Sensitivity = \frac{TP}{TP + FN} \times 100\ \%. \tag{2.9}$$

Specificity: is determined by the ratio of TN to the sum of FP and TN, as described:

$$Specivity = \frac{TN}{TN + FP} \times 100\ \%. \tag{2.10}$$

F Score (Measure F): is a metric calculated as the average of precision symphony and memory [39], as denoted by the following equation:

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \times 100\ \%. \quad (2.11)$$

2.3 Experimental Results and Discussion

The experimental results were obtained from the 20% of the dataset as the test data after augmentation as described previously. First of all, YOLOv4 was trained with raw images as a whole, and the test results were evaluated. From the test outcomes, the results were not satisfactory. Only one class (Mori) achieved good accuracy (96.43%), while the other classes only achieved 65% accuracy (Catla) or below (Rohu; 48.28% and Silver; 39.29%). Class G. Carp achieved very low accuracy (9.09 %), and even class C. Carpio achieved 0 % accuracy. The overall accuracy of this trial was only 43.01 %.

Then YOLOv4 was trained by applying the landmarking technique to each input image. This technique made the accuracy performance increase to 72.65 %. However, the class C. Carpio and G. Carp still got low accuracy scores at this step which are still not acceptable (15 and 40.91%).

Then the subclassing technique was applied. At this step, C. Carpio red was separated and became an additional class, but it was still grouped as C. Carpio. As a result, the overall accuracy increased to 76.64%. The accuracy results in class C. Carpio also increased significantly, from 15 to 60%, and Catla from 80 to 100 %. However, it affected other classes' accuracy became lower. The Rohu class, originally 100, fell to 86.21 %, and the G. Carp fell significantly from 40.91 to 13.64%.

The next experiment was adding the scale data to the training process. As a result, the final average accuracy increased to 77.42%. However, several classes produced low and unacceptable accuracy, namely C. Carpio (40%), G. Carp (54.55%), and Catla (70%). Then the head data was added. The result was unexpected because it did not improve accuracy but slightly lowered it. The overall accuracy average dropped to 76.47%. Some classes still produced a low and unacceptable level of accuracy, namely G. Carp (27.27%), C. Carpio (50%), and Catla (85%).

For the next step, a trial using square labelling was performed. At this step, an accuracy of 86.94% was reached. Catla and Silver were recognized perfectly 100%, Mori and Rogu got good accuracy (96.43% and 96.55%); however, C. Carpio and G. Carp got insufficient accuracy (65.00% and 63.64%). And the last experiment applied the class elimination technique. From the series of trials that had been carried out, three classes always got insufficient and unacceptable accuracy scores; G. Carp, C. Carpio, and Catla. So two classes were eliminated by applying (2.5) as described previously. So we eliminated two classes with the lowest accuracy results (G. Carp and C. Carpio). Finally, from the test results, the final accuracy value of this step was excellent and could reach 98.75%. Even so, the average value of accuracy for each class reached 95% or more.

The accuracy results of this series of experiments are summarized in Table 2.4 and Figure 2.7. Figure 2.8 shows the results of the evaluation by the confusion matrix, and Figure 2.9 shows the other parameters' results (precision, recall/ sensitivity, specificity, and F score).

Table 2.4 Experimental results (accuracy)

Class	Detection Accuracy (%)						
	Body (Whole Image)	Body (with landmarking technique)	Body (after subclassing)	Body+Scale	Body+Scale+Head	Body+ Square Labelling	Body (4 classes)
Catla	65.00	80.00	100.00	70.00	85.00	100.00	95.00
C. Carpio (+C. Carpio Red)	0.00	15.00	60.00	40.00	50.00	65.00	-
G. Carp	9.09	40.91	13.64	54.55	27.27	63.64	-
Mori	96.43	100.00	100.00	100.00	100.00	96.43	100.00
Rohu	48.28	100.00	86.21	100.00	96.55	96.55	100.00
Silver	39.29	100.00	100.00	100.00	100.00	100.00	100.00
Average	43.01	72.65	76.64	77.42	76.47	86.94	98.75

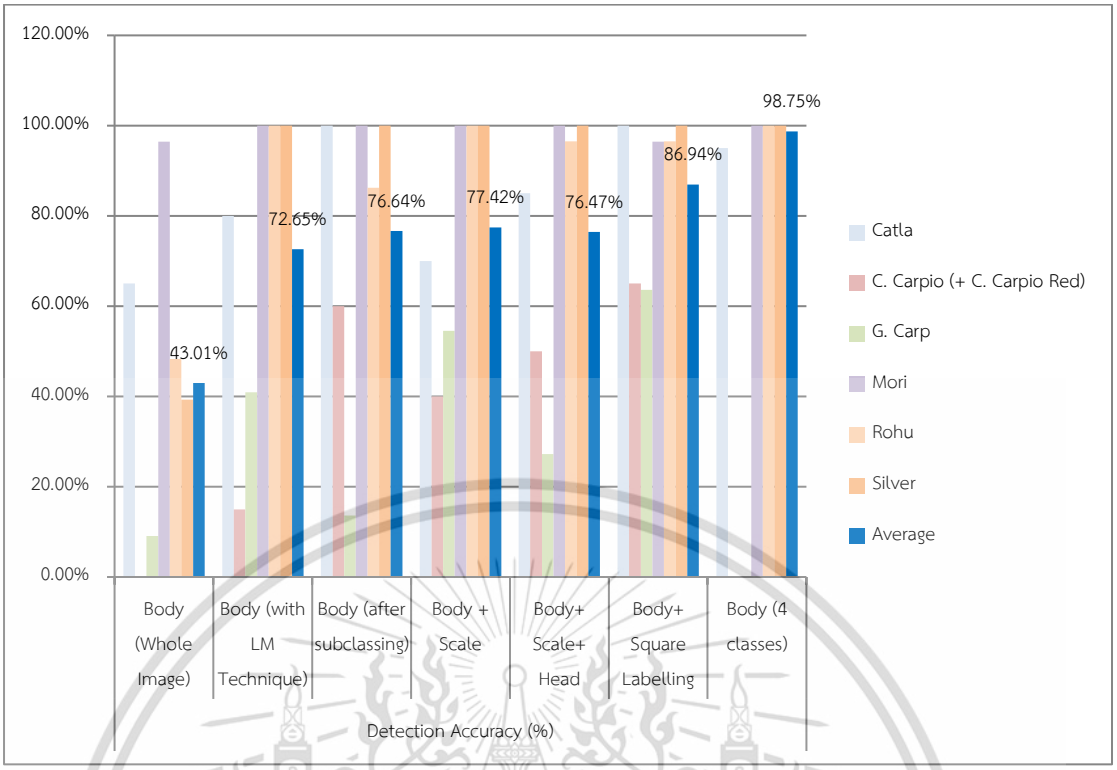
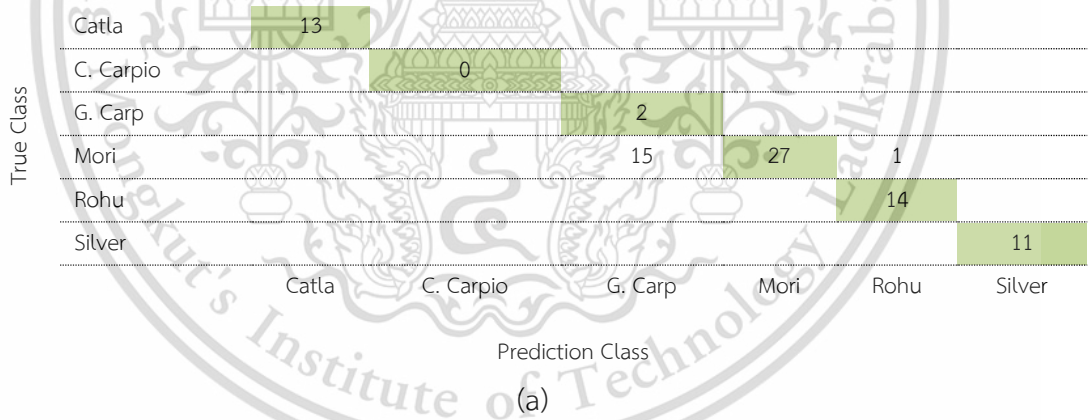
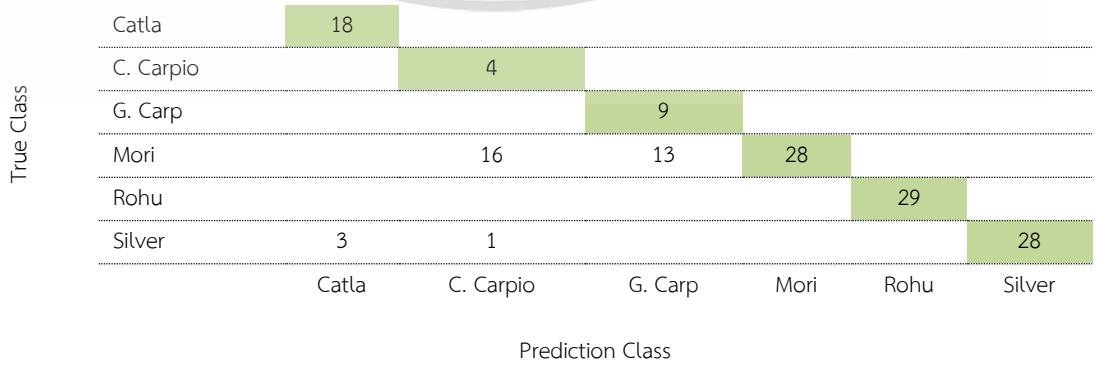


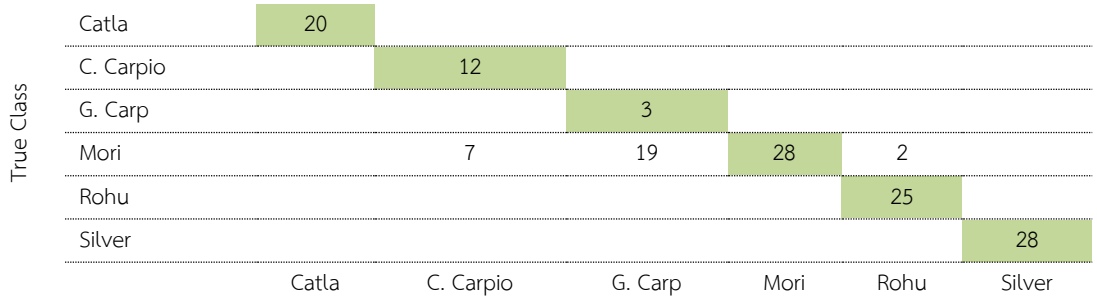
Figure 2.7 Experimental results (accuracy)



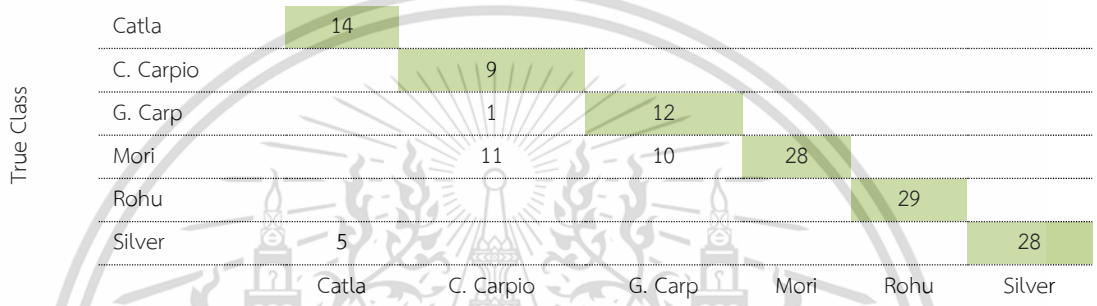
(a)



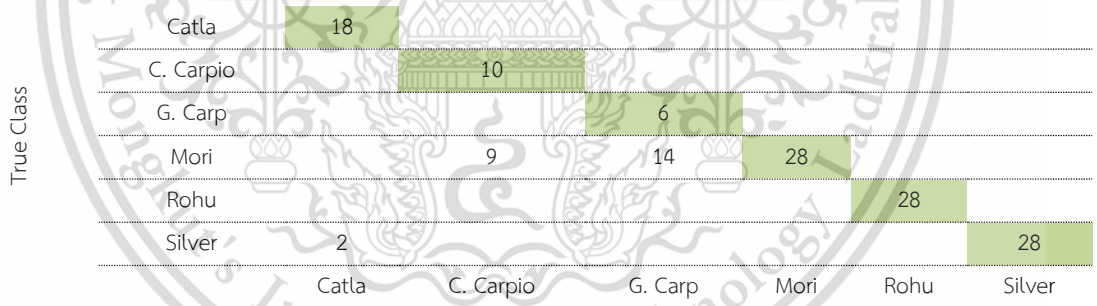
(b)


Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
20
12
3
7
19
28
2
25
28
Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
True Class
Prediction Class

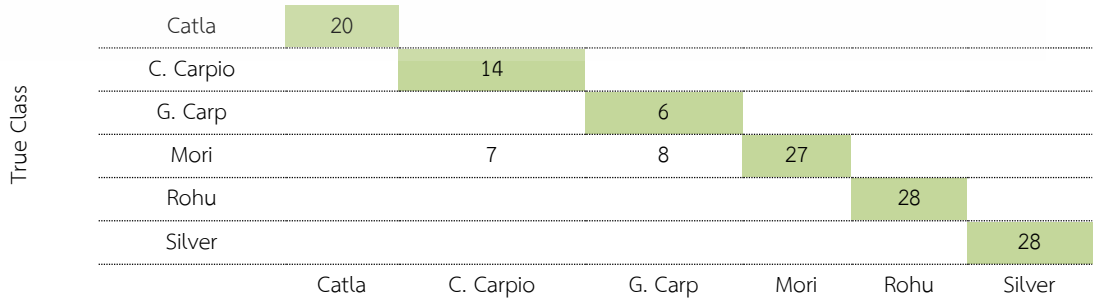
(c)


Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
14
9
1
12
11
10
28
29
5
28
Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
True Class
Prediction Class

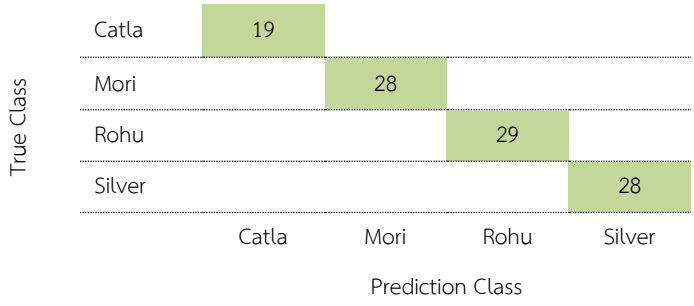
(d)


Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
18
10
6
9
14
28
28
2
28
Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
True Class
Prediction Class

(e)


Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
20
14
6
7
8
27
28
28
Catla
C. Carpio
G. Carp
Mori
Rohu
Silver
True Class
Prediction Class

(f)



(g)

Total data	
Catla	20
C. Carpio	20
G. Carp	22
Mori	28
Rohu	29
Silver	28

(h)

Figure 2.8 Confusion matrix results for: (a) training with the whole image; (b) with landmarking technique; (c) with subclassing technique; (d) with the addition of scale image; (e) with the addition of scale and head image; (f) with square labelling; (g) with only four classes (after class elimination) and (h) total data

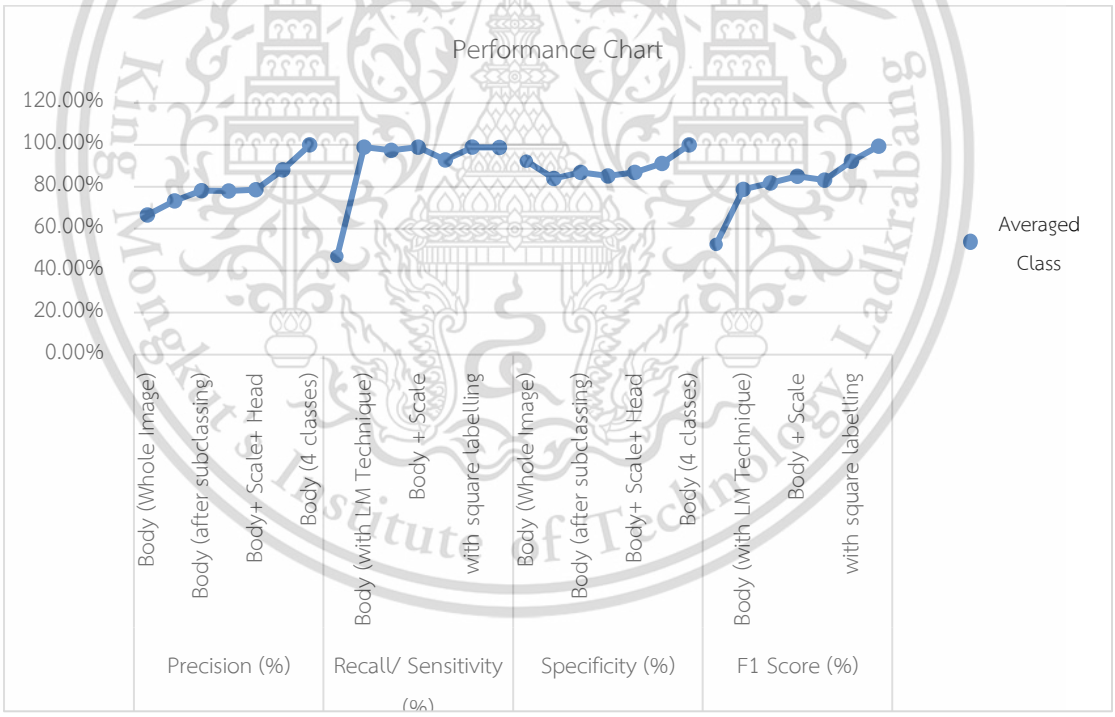


Figure 2.9 Experimental results (precision, recall/ sensitivity, specificity, and F score)

2.4 Limitations and Future Development

A limitation should also be reported in this work. This study utilized CiRA-Core software as the main tool to run YOLO as a recognition algorithm. This software has

many advantages, such as being easy to use and integrated (supports ready-to-use technology). However, the main limitation related to this work is that the YOLO algorithm provided is a patent/ cannot be modified. Because of that, it is not possible to do a study to modify the YOLO algorithm to improve recognition performance.

Another point that should also be reported is the weakness. Suppose referring to the report of the results of this work, it is possible to agree that this study is very good for finding out the impact and increase in the use of the techniques used in combination with YOLOv4 on similar and deformed fish objects on the level of accuracy or other performance parameters. However, the only major disadvantage in this study is that the best results could be achieved by eliminating (sacrificing) two classes of fish with low recognition performance. This matter makes this work incapable of fully answering the challenge.

For this reason, this study can be further developed in the future to improve accuracy and other performance parameters for an entire class. It may be done by modifying the YOLO algorithm used (by another tool) or applying image processing techniques that have not been performed in this study.

2.5 Summary

A series of experiments shows that YOLOv4 is promising for detecting and classifying fish species with similar and deformed conditions, which are characteristic fish in the aquaculture industry. On the Fish-Pak dataset, which contains six species of fish, the accuracy of YOLOv4 is only 43.01%, but the result could be optimized and rose to 72.65% with the landmarking technique, and rose to 76.64% with the subclassing technique, then rose to 77.42% by adding scale data. The accuracy did not improve to 76.47% by adding head data, 86.94% by square labelling and finally, the accuracy rose to 98.75% with the class elimination technique. However, the accuracy rate can be further improved with complete classes (without class elimination) in future work. Image processing techniques may be improved for pre-processing or also by modifying the recognition algorithm.

Chapter 3
Optimizing YOLOv4 For Fish Classification In Various Background Conditions

3.1 Introduction

One of the challenges of fish classification (FC) is its diverse backgrounds [11, 39-41, 43, 45, 47]. Images of fish with various backgrounds commonly occur because they are taken in the wild [11, 39-41, 45, 47], during the fishing process, or in temporary storage [43]. This diverse background is a difficult challenge because it can come from various causes such as water, coral reefs, seaweed, light flashes [11, 39-41, 45, 47], and various backgrounds from fishing or temporary storage areas [43]. This challenge must be overcome to optimize the classification result. The classification process will be poor if the extracted fish features mix with the background [13]. For this reason, many approaches have been proposed to optimize fish classification to overcome this challenge. The recently developed techniques used to overcome this challenge are miscellaneous, ranging from image preprocessing to modification of the classification algorithm.

To address the challenge of diverse backgrounds for fish classification, [11] modified AlexNet to classify fish images taken from the LifeClef2019 and QUT datasets against natural backgrounds. [47] employed ResNet-50 to classify aquatic fish species taken from the Fish4Knowledge dataset. They reduced the vanishing gradient problem of ResNet-50 to a minimum by using residual blocks. By using this technique, and then training only the last few layers of the ResNet-50 network with transfer learning, they reported an increase in accuracy with that approach. [45] modified VGGNet-16 and then applied it to the ImageCLEF FISH_TS dataset, which was also extracted from underwater fish images.

[13] removed the background first before entering the fish image into the classification algorithm. The foreground was identified using BLOB (Binary Large Object) then the background and noise were removed by threshold. They used the classic AlexNet algorithm combined with Naive Bayesian Fusion (NBF) for classification. The proposed approach was tested on the BYU fish dataset with various backgrounds

condition. [43] proposed a fish classification approach captured from camera vessels. The static fish images were then collected in a dataset called Kaggle. The dataset consists of images of 8 types of fish with various vessel area backgrounds. In addition, the portion of the fish object in the picture is also almost all small or not dominant. The proposed method marks the fish object on the image with a mask and then uses a convolution algorithm for classification. Their experimental results obtained the best classification results from a combination of rotating (image augmentation), mask, and CaffeNet techniques.

[41] hybridized the feature extraction from the standard AlexNet and fine-tuned approaches, applied to Fish Recognition Ground-Truth and LifeCLEF 2015 Fish dataset. Then they developed their work [40] still in the Fish Recognition Ground-Truth dataset for their research. They extracted features from foreground fish images using the pre-trained AlexNet and employed linear SVM (Support Vector Machine) for the classifier. They reported that the accuracy result was also good enough in the QUT dataset. For another work, [39] reduced AlexNet to only four convolutional and two fully connected layers. They tested their proposed method on the QUT dataset to classify fish species in various background images.

From the review above, there is no research using YOLO to classify fish with diverse background challenges. In contrast, YOLO is a very popular algorithm and deserves to be developed in this field. Meanwhile, the development of YOLO for fish classification is very important both for underwater fish recognition and in the fish industry. So, we propose a method of classifying fish on various backgrounds condition using YOLO version 4 (YOLOv4) combined with a landmarking technique in the labelling process. This approach is much simpler than the other approaches reviewed. And this landmarking technique is also relatively new and rarely used, especially for fish classification.

3.2 Material and Methods

3.2.1 Dataset and Image Augmentation

This work aims to detect and classify fish in various background conditions, such as objects, colours, and others. As a result, the BYU dataset is thought to be excellent for use in this study. The Brigham Young University (BYU) robotic vision group created this dataset [57]. This dataset consists of 16 different fish species with varying shooting

conditions. Several fish images are captured in the processing unit on a table with an image calibration plate, providing guidelines for preprocessing and required colour correction. Several additional fish photographs were captured in natural and artificial environments. Some photographs have also been processed, such as cropping and recolouring.

Furthermore, each species' image from the image data of 16 fish species is very unbalanced. For these reasons, only four species were used in this study, with a large amount of image data, raw images (not yet processed), and varying backdrop conditions (natural and underwater backgrounds). Table 3.1 lists the four fish species as B.C. Trout, Kokanee, UT Sucker, and Steal Head. Figure 3.1 shows each example of the fish in the BYU dataset.

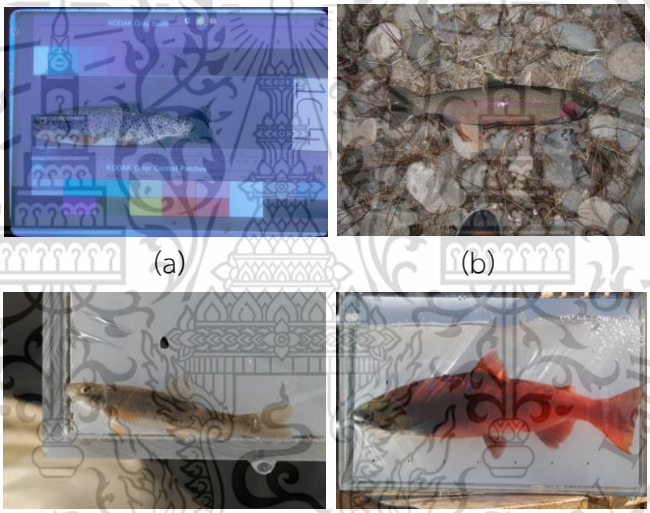


Figure 3.1 Sample Images of BYU dataset: (a) B.C. Trout; (b) Steal Head; (c) UT Sucker and (d) Kokanee

Table 3.1 BYU dataset and image augmentation

Fish class	No. of images	Multiplication factor	No. of augmented images	New image dataset	For Training (80%)	For testing (20%)
B.C. Trout	191	0	0	191	153	38
Kokanee	60	1	60	120	96	24
UT Sucker	87	1	87	174	139	35
Steal Head	25	3	75	100	80	20
Total	363	-	222	585	468	117
Standard Deviation Value (SDV)	61.91	-	-	37.42	-	-
Average	91	-	-	146	-	-

It can be observed from Table 3.1 that the amount of image data for the three classes, namely Kokanee, UT Sucker, and Steal Head, is small and unbalanced in comparison to the overall class image data. Small data (less than 100) reduce validation, and imbalanced data prevent the algorithm from having the same opportunity during the training process. For this reason, an augmentation process is needed [13]. The augmentation techniques used in this work are flip, rotation, and translation. This approach is thought to be appropriate for fish augmentation [58]. The augmentation process can be described by (2.1) -(2.4) from the Chapter 2.2.1.

After the augmentation procedure, the new dataset has a significant number of images, ranging from 363 to 585. In addition, the average number of images in each class increased from 91 to 146. Moreover, the dataset is now more balanced, evidenced by a decrease in the standard deviation value from 61.91 to 37.42. Following that, the images were randomly separated for the training (80%) and testing (20%) processes [11].

3.2.2 Occupancy Ratio and Landmarking Technique

The occupancy ratio of an image or image in a bounding box (OR_{bb}) is a comparison between the object area or object bounding box and the overall image area or image bounding box, which includes the background. The high occupancy ratio minimizes the deep learning algorithm capturing unwanted backgrounds during the

training process. It will improve the effectiveness of deep learning in capturing features that can only be found in objects. The occupancy ratio is denoted by (3.1) [13].

$$OR_{bb} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{bb}(i,j)}{M \times N} \tag{3.1}$$

where, $\sum_{i=1}^{M}\sum_{j=1}^{N} I_{bb}(i,j)$ is the object area or object bounding box, and $M\ x\ N$ is the overall image area or image bounding box.

The landmarking technique is a relatively new labelling technique currently utilized sparingly, especially on fish objects. This technique is employed immediately before the image is utilized for the training process. With this landmarking technique, any part of the object can be marked and become an area that is only processed during training. With this method, the occupancy ratio will become very high or may even reach one, leading to more effective and on-target feature maps. As a result, the training process will be more effective, and the recognition results will be more accurate and robust.

This landmarking technique is carried out with the CiRA-Core software, which developed by the College of Advanced Manufacturing Innovation (AMI-KMITL) Thailand and available on https://git.cira-lab.com/cira/cira-core. This software was advanced introduced and used at work [59] and [60]. How to run the landmarking on this software is we put the coordinate points on the object in the image, and then the program will convert it into a polygon and form an area on the object. Only objects or parts of objects marked as functional areas will be extracted for the training process, and anything outside will be ignored. Consequently, in addition to the bounding box (bbox), centre point, colour, label, and index label information, the output of this process includes data landmark points and landmark len, which will be used in the training process to improve its effectiveness. The concept of this landmarking technique and the illustrations on the software can be observed in Figure 3.2.

Furthermore, this paper also presents fish recognition using the most commonly used conventional labelling technique. The comparison between the landmarking technique and the conventional technique can be observed in Figure 3.3.

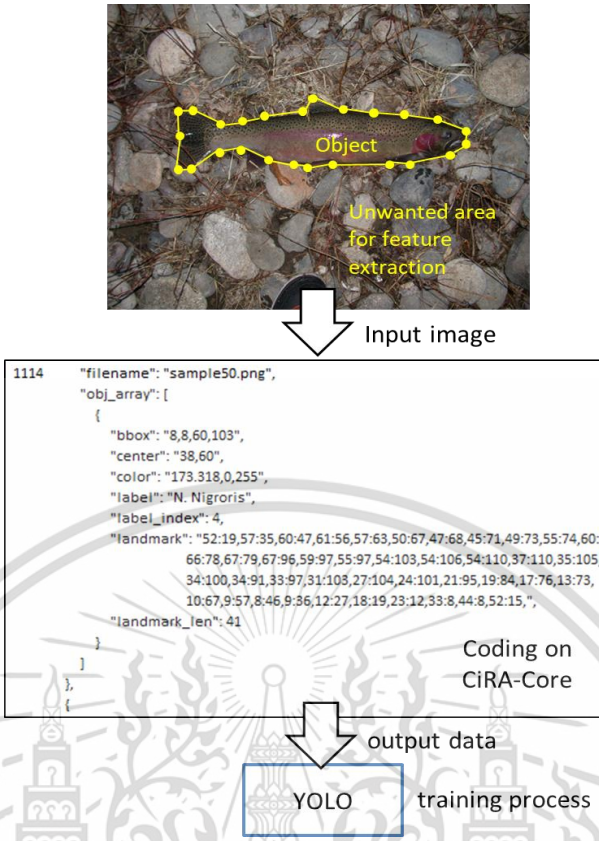


Figure 3.2 The concept of landmarking technique

(a) (b)

Figure 3.3 Labelling technique: (a) using the landmarking technique and (b) using the most common or conventional technique

3.2.3 YOLOv4

In this work, the detection and classification algorithm is also implemented in YOLO version 4 (YOLOv4). Still in the same reasons, YOLO *(You Only Look Once)* is viral and extensively used object detection and classification algorithm because of its known and reasonable detection rate, accuracy, and fast detection time. Futhuremore,

YOLO is still rarely used in fish classification. So YOLOv4 is expected to be promising to overcome the challenge as the aim of this work. The initial architecture of the YOLO can be observed in Figure 2.3 in Chapter 2.2.2. The detailed architecture or the layers of the YOLOv4 are described in Appendix A.

YOLOv4 was learned using training data from the BYU dataset (as shown in Table 3.1) and achieved good results with an average loss= 0.01. Training is done in batches of 32 data points with 32 subdivisions. Data enhancement is accomplished by rotation with a threshold ranging from -180° to 180° in 90 steps and contrast with a threshold ranging from 0.4 to 1.1 in 0.2 steps.

3.2.4 Validation Matrix

The confusion matrix is used to describe the model's performance, and the model's accuracy may be assessed using (2.6) and (2.7), which are already described in Chapter 2.2.4.

3.3 Experimental Results and Discussion

The proposed approach was tested on the BYU testing data. As explained before, testing data consists of 20% images from every class after augmentation. The trained algorithm was tested on one by one test images and got four result conditions; *correct classification, wrong classification, not detect,* and *double classification*. *Correct classification* is when the model can classify fish correctly (as the true class). The *wrong classification* is when the model predicts incorrect fish, *not detect* when the model fails to detect the fish, and *double classification* is when the model classifies one as two different fish.

Every output or model's prediction has a classification or confident score from 0 to 100%. Moreover, we set the threshold to 50% in the first step. It will make the model will ignore the decision with less than 50% of confidence or classification score. The experimental result of the trained algorithm with the landmarking technique is resumed in Table 3.2 and described in Figure 3.4.

From Table 3.2, we can observe that the BC Trout class has 38 images as testing data, and the model could correctly classify 35 fish images but failed to detect fish in 3 images. So we can define the accuracy is 92.11% by the equation (2.7) with the total

prediction of 38 (35 Correct classification+ 0 Wrong classification+ 3 of Not detect+ 0 Double classification). By all correct classifications, the average classification (confident) score is 99.67%. From the confusion matrix as in Figure 3.4 (a), we can see the prediction vs true class for the BC Trout. It shows that only 35 were correctly classified as BC Trout.

For the Kokanee class, the model could perfectly classify 24 testing images. So the accuracy is 100%, and in the confusion matrix in Figure 3.4 (a), 24 is defined to Kokanee as the true class. However, the UT Sucker and Steelhead classes have wrong and double classifications. Furthermore, the details of what miss class is classified to (for the wrong classification) or what other species are detected too (for double classification) can be observed in the confusion matrix. For example, Steelhead has 1 double classification. One fish image was classified as Steelhead and also BC Trout. At this step, the average accuracy for all classes was 93.92%.

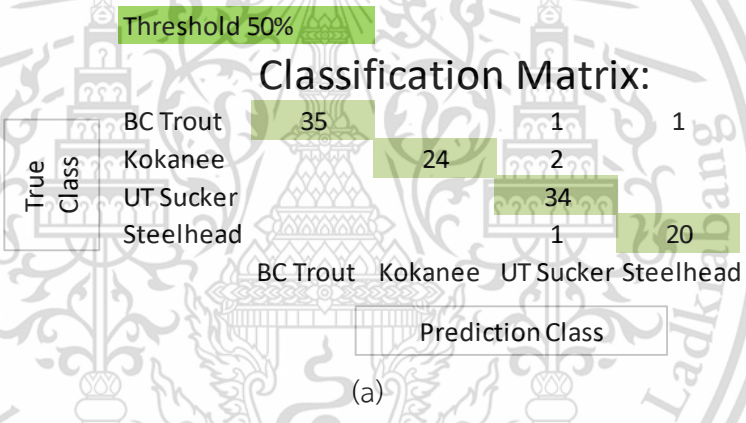
At the 50% threshold, we found that the model often makes the double classifications but in a lower (less than 60%) classification score for the incorrect fish (the double classification detects one correct fish and one incorrect). So in the second step of the experiment, we set the threshold to 60% to optimize the overall accuracy. By increasing the threshold, the final accuracy could reach 96.60%.

Table 3.2 Experimental results of YOLOv4 with landmarking technique at 50% and 60% threshold

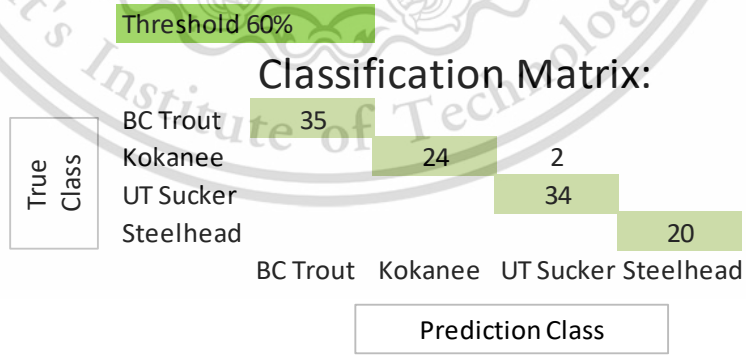
Threshold: 50%

Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	0	3	0	38	92.11	99.67
Kokanee	24	24	0	0	0	24	100.00	99.93
UT Sucker	35	31	1	0	3	35	88.57	99.20
Steelhead	20	19	0	0	1	20	95.00	99.09
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	93.92	99.47

Threshold: 60%								
Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	0	3	0	38	92.11	99.67
Kokanee	24	24	0	0	0	24	100.00	99.93
UT Sucker	35	33	1	0	1	35	94.29	99.24
Steelhead	20	20	0	0	0	20	100.00	99.83
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	96.60	99.67



(a)



(b)

Figure 3.4 Confusion matrix for experiment results of YOLOv4 with landmarking technique: (a) at 50% threshold and (b) at 60% threshold

This work also sets the trained YOLOv4 algorithm model toward the test using the conventional labelling approach, which is routinely utilized on the same test data. We also did it in two steps; taking data at a 50% threshold and then 60%. The final average accuracy result of the conventional labelling technique is 89.19% at the 50% threshold, and 91.66% at the 60% threshold. At the 50% threshold, the accuracy result is 4.73% lower than the method using the landmarking and 4.94% lower at the 60% threshold. Table 3.3 summarizes the findings of YOLOv4 detection using this conventional labelling technique, and Figure 3.5 depicts the confusion matrix.

Table 3.3 YOLOv4 detection results with conventional labelling techniques at 50% and 60% threshold

Threshold: 50%

Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	1	2	0	38	92.11	99.12
Kokanee	24	21	0	0	3	24	87.50	99.99
UT Sucker	35	27	0	0	8	35	77.14	99.96
Steelhead	20	20	0	0	0	20	100.00	99.87
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	89.19	99.74

Threshold: 60%

Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	1	2	0	38	92.11	99.12
Kokanee	24	22	0	0	2	24	91.67	99.99
UT Sucker	35	29	0	0	6	35	82.86	99.96
Steelhead	20	20	0	0	0	20	100.00	99.87
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	91.66	99.74

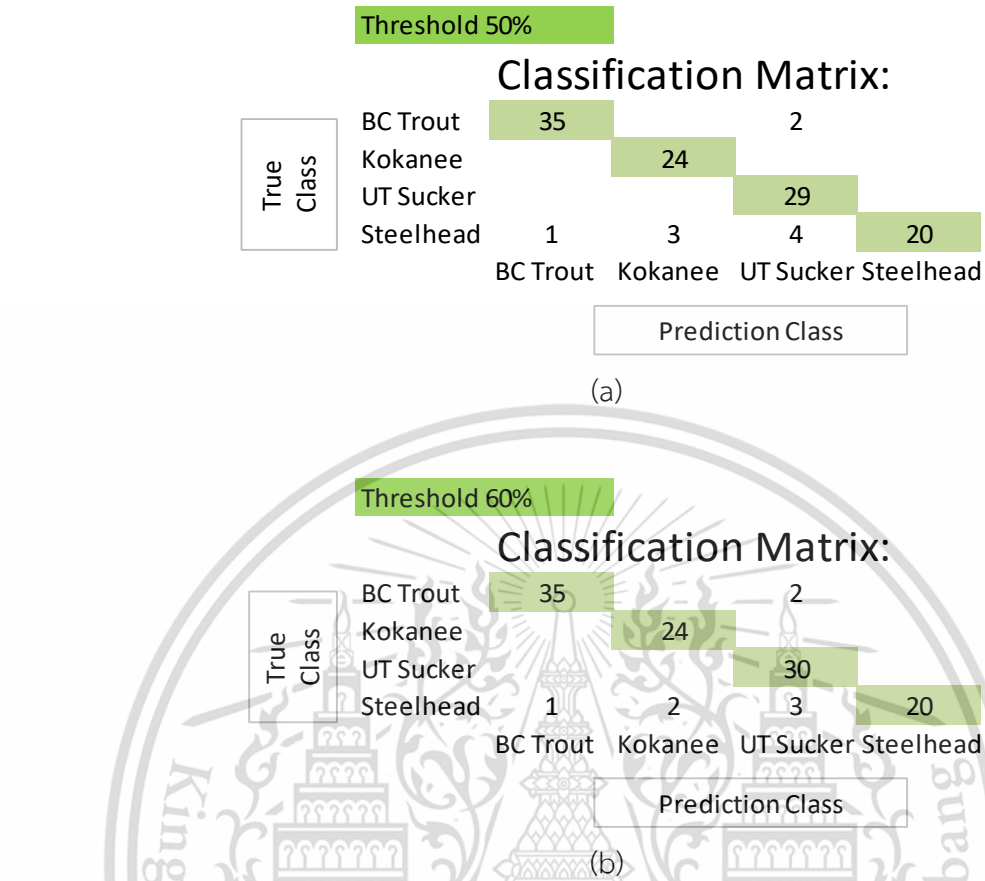


Figure 3.5 YOLOv4 detection results with conventional labelling techniques at: (a) 50% threshold and (b) 60% threshold

3.4 Limitations and Future Development

Another issue that we want to report is that although the final accuracy results obtained are quite good (>95%), these results can still be improved by modifying the YOLO algorithm used or using image processing techniques that have not been carried out in this work. Modifying the YOLO algorithm cannot be carried out in this work because the tools used do not allow it, which is the biggest limitation of this research. Using image processing techniques will emphasize feature maps on the object before the training process so the captured features can be more effective and improve the final accuracy result.

However, this work is considered quite good as a start and can be developed in the future. Image processing techniques can be used, and the YOLO algorithm can be modified to increase accuracy or make computations more efficient.

3.5 Summary

The purpose of this study is to provide an approach for identifying fish in a variety of background conditions. The methodology consists of combining the landmarking technique with YOLO version 4. The proposed approach was tested with four different fish species using the BYU dataset. The detection accuracy findings from the experimental tests are 93.92% at the 50% threshold and 96.60% at the 60% threshold. This result is higher than the conventional labelling procedure, which yielded 4.73% at the 50% and 4.94% at the 60% thresholds. This trial's outcomes are considered quite good, and the method used is quite straightforward. The proposed approach does not necessitate any complicated work steps, and it is ready to be used as a ready solution.

Chapter 4
Optimizing YOLOv4 In Moving Fish Recognition For Automatic Sorting System

4.1 Introduction

Automatic fish recognition using computer or machine vision is a key part of automating the fish industry, which is part of the food industry, to boost productivity. One of them is for the automatic sorting system, which is being worked on by many researchers [61-64]. Automatic detection and classification of moving fish are the keys to developing it, and they have some unique challenges. The main challenges are the speed and the random position of the fish [13].

For moving fish recognition, many studies have been carried out. The study in [7] employed CNN (Convolutional Neural Network) to classify fish by training them with the number of species and their environments, such as reef bottoms and water. They applied their proposed method to 116 underwater fish videos recorded using a GoPro underwater camera. The best results were achieved in classifying 9 of the 20 types of fish that appear most often in the videos. In [8], a multi-cascade object detection network with an ensemble of seven CNN components and two RPNs (Region Proposal Network) linked by sequentially jointly trained LSTMs (Long Short-Term Memory units) was performed. For training and testing, they used a set of 18 underwater fish videos that were also recorded with a GoPro underwater camera. Even though their proposed method can reliably find and count fish objects in a variety of benthic backgrounds and lighting conditions, it is only used to find fish and not to classify them. The moving fish recognition in [9] utilized Optical flow, GMM (Gaussian Mixture Models), and ResNet-50, then combined the output with YOLOv3. The combination of those methods enabled the robust detection and classification of fish, which was applied to the LifeCLEF 2015 benchmark dataset from the Fish4Knowledge repository [65] and a dataset collected by the University of Western Australia (UWA) which was explained in detail by [66]. The GMM and Pixel-wise posteriors were proposed in [11], and then further developed by combining them with CNN [30]. They also used a fish dataset extracted from the Fish4Knowledge repository in their work. Similar to the work [8],

the approaches proposed in their papers were only for detecting fish without classifying them.

Abinaya et al. [13] segmented the fish into three parts; head, body, and scales. Then an Alex-Net was used to classify each of these parts. Naive Bayesian Fusion (NBF) was then utilized to determine the final classification results. The accuracy obtained from this approach was quite well applied to the Fish-Pak [52] and BYU [57] datasets. Still, the fish images used were only static, even though the narrative of this work was intended for an automatic sorting system. Mohamed et al. [15] proposed an approach for fish detection in aquaculture ponds. Image enhancement was used to improve fish detection in cloudy water conditions, and then YOLOv3 was utilized to detect the fish. However, this approach is not intended for classification but for counting and tracking fish trajectory. Xu et al. [16] applied Faster R-CNN and compared it with YOLOv3 to detect and record fish trajectory to study its behavior and relationship to ammonia levels in pond water.

However, the works reported in [9],[12],[30] recognized moving fish for underwater (ocean) environments, while some only detected fish without classifying them. Moving fish in aquaculture was discovered by [15],[16], but not used for classification. The work in [13] classified fish with narration for the automatic sorting system, but the datasets used were static images. To the best of the authors' knowledge, there is no published work to detect and classify moving fish for the fish culture industry, especially for automatic sorting based on fish species using deep learning and computer vision. No public dataset is also available for cultured fish run on conveyors, which is very useful for developing it. This work will fill that gap. We present a dataset of aquacultured freshwater fish moving along a conveyor belt and propose a method for detecting and classifying it using an optimized YOLOv4. YOLO is the most popular object detection algorithm, but it is still rarely employed for fish recognition.

By using the proposed method and using real videos of freshwater fish running on a conveyor, it is anticipated that this work will serve as a guide and provide solutions to the challenges of detecting and classifying fish for automatic sorting, which is very close to the actual condition.

4.2 Materials and Methods

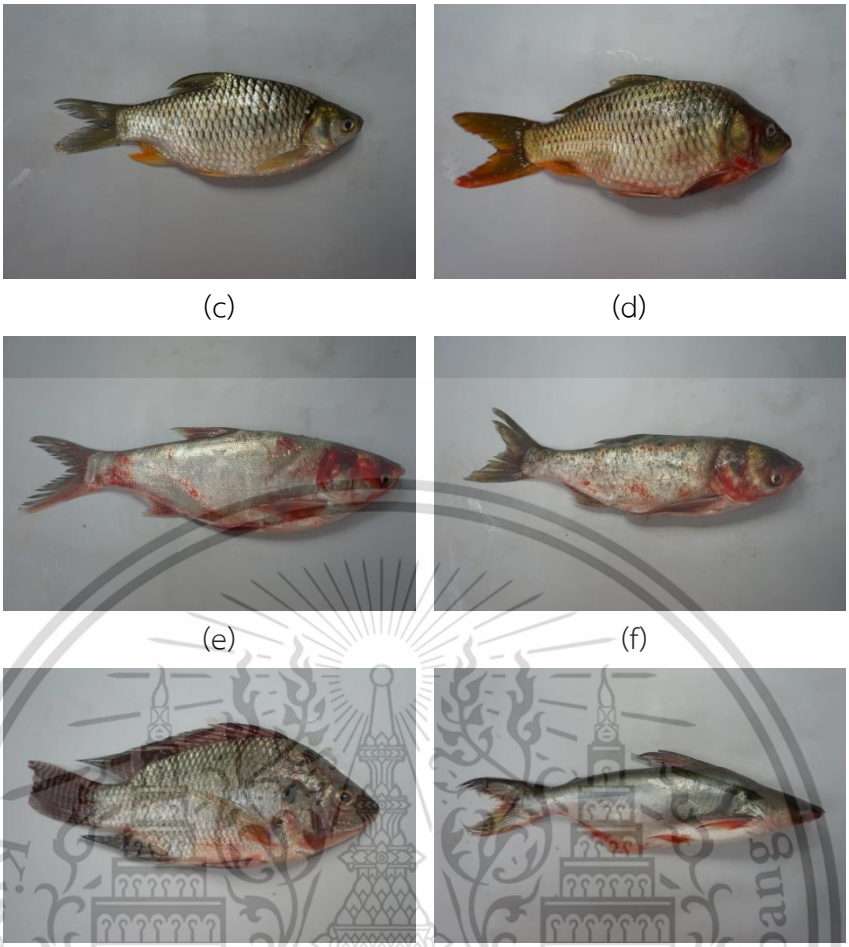
4.2.1 Images Dataset and The Experimental Set-Up

The purpose of this work is to develop an approach to automatically detect and classify fish for automatic sorting systems in the fish industry. As far as the authors know, there is no publicly available dataset for cultured fish run on conveyors that can be used for this purpose. For that, we created our own dataset for this work. We took fish samples from eight types of farmed fish species called "Ben-Cak," which are generally bred, sold, and consumed in and around Thailand. Three of these fish species are endemic, originally from the Mekong and Chao Phraya rivers, which are also in Thailand. The eight fish species are:

1. Yeesok *(Labeo rohita)*,
2. Nuanchan *(Cirrhinus microlepsis)*,
3. Tapian *(Barbonymus gonionotus)*,
4. Nai *(Cyprinus carpio)*,
5. Jeen Ban (*Hypophthalmichthys molitrix)*,
6. Jeen To *(Hypophthalmichthys nobilis)*,
7. Nin *(Oreochromis niloticus)*, and
8. Sawai *(Pangasianodon hypophthalmus)*.

Sample pictures of each type of fish can be seen in Figure 4.1. In Thailand, these fish are bred together (mixed) in the same pond on the fish farm, and then when harvested, these fish will be brought to the fish hub for sorting and weighing for further sale to consumers. For this reason, these fish are considered very suitable for this work because the sorting process carried out at the fish hub is still done manually by humans.

(a) (b)



(c) (d)

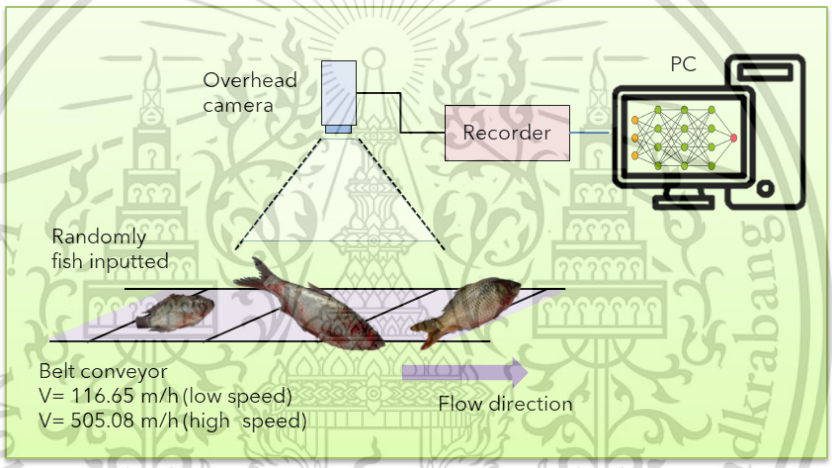
(e) (f)

(g) (h)

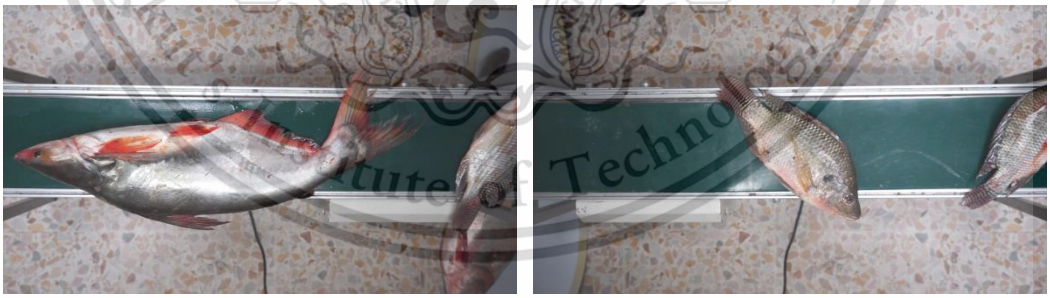
Figure 4.1 Sample picture of fish: (a) Yeesok; (b) Nuanchan; (c) Tapian; (d) Nai; (e) Jeen Ban; (f) Jeen To; (g) Nin and (h) Sawai

In creating the dataset, we did not just take static photos of each fish sample from several views, as shown in Figure 4.1. We ran a conveyor and randomly put the fish on it in both position and order. Then we recorded it with the overhead camera with two-speed settings; low (116.65 m/h) and high (505.08 m/h). The recordings were carried out in the Food Engineering laboratory at King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand, with room lighting conditions and additional light from an LED lamp. The measured light intensity was 846 lux/79 FC at all times during the recordings. We produced three videos; one low-speed video with a duration of 17 min and 13 s, and two high-speed videos, with a duration of 8 min and 24 s (later referred to as high-speed video 1) and 17 min and 13 s (later referred to as high-speed video 2). The camera used was a SONY Model ILCE-7 with a frame size setting of 1920 × 1080 (W × H) and a 29.97 frames/second frame rate.

In the videos produced, the background of the fish object is not entirely a conveyor, with a monotone condition. The conveyor is small and does not dominate the entire frame; it also has other objects in the background of the frame. So its condition makes this dataset more challenging. According to the authors, this is the first work that utilizes videos of aquacultured fish running on a conveyor. The experimental setup to create the dataset and the sample for capturing results can be seen in Figure 4.2. Table 4.1 and 4.2 show images from the dataset that was created. And this dataset is uploaded and made publicly available at: https://drive.google.com/drive/folders/1OrEUIYdiDWbvDyUV46WSrr6Yd83OTuwt?usp=sharing.



(a)



(b) (c)

Figure 4.2 The experimental set-up for: (a) taking the videos; while (b, c) are examples of capturing video results

Table 4.1 Images dataset (static pictures)

Fish Class	Yeesok	Nuanchan	Tapian	Nai	Jeen Ban	Jeen To	Nin	Sawai
No. of images	20	20	20	20	20	20	20	20
Total				160				
Average per class				20				

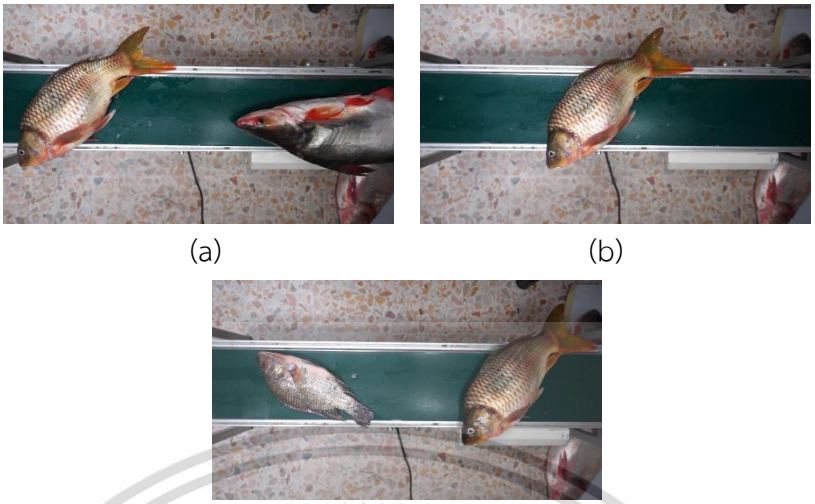
Table 4.2 Images dataset (videos)

No.	Name	Conveyor Speed (m/h)	Duration	Note
1	low-speed video	116.65	17 min 13 s	later extracted for training data (scheme 2)
2	high-speed video 1	505.08	8 min 24 s	for testing data
3	high-speed video 2	505.08	17 min 13 s	for testing data

4.2.2 Training Images and Augmentation

The algorithm for detecting and classifying (recognition) needs to be trained, so images for training should be prepared. This work uses two schemes for using or generating training images. First, using static pictures from each fish class, and second, generating and using extracted pictures taken from one of the video recordings from the dataset. Each image prepared is then augmented in both the first and second schemas to enrich the data for a better training process [13, 67, 68]. The augmentation techniques used in this work are vertical and horizontal flips, which are suitable for the case of fish objects running on conveyors [58].

Scheme 1 employed 160 static images of each fish from eight classes. The images were then augmented to enrich the data. Each augmentation technique was applied to each original image. So, every one of the 160 new images was obtained using vertical and horizontal flip techniques. All images are then combined (original and augmented) and used as training images. In scheme 2, the low-speed video was used and extracted into 188 static images. Each fish that appears in the video was extracted by taking screenshots at three positions, as shown in Figure 4.3: when the fish appears in its entirety (a), at the exact center position (b), and just before the fish's snout or tail hits the right-hand frame border to leave the frame (c). Then augmentation was applied to enrich the data with the same method as in scheme 1. Furthermore, training images were also obtained by combining the original images with augmentations, as in scheme 1 as well. The training images and augmentations used in this work are summarized in Table 4.3.



(a) (b)

(c)

Figure 4.3 Example of extracting images for one fish from the low-speed video: (a) when the fish appears in its entirety; (b) at the exact center position and (c) just before the fish's snout or tail hits the right-hand frame border to leave the frame

Table 4.3 Images for training and augmentation

Scheme	Original Images (8 Classes)	Averaged per Class	Augmented Images			Total Images for Training	Averaged per Class
			Flip-Vertically	Flip-Horizontally	Total		
Scheme-1 (from static pictures)	160	20	160	160	320	480	60
Scheme-2 (from extracted pictures)	188	24	188	188	376	564	71

4.2.3 Labelling Techniques

The labelling process is utilized during the training step. There are three types of labelling used in this work: conventional, using landmarking, and a combination of conventional and landmarking. The conventional labelling technique is the most commonly used, but it has a weakness. The object's background is included so that the algorithm's feature extraction is carried out both on the object and background. It can make the learning process less effective and impact the recognition results [13, 69]. Labelling using the landmarking technique was introduced by the authors for the first time in previous works [23, 70]. Table 4.4 provides a summary of the previously stated works. This table contains important findings regarding the effect of certain

labelling techniques on the recognition accuracy of fish. The landmarking technique can optimize the recognition algorithm significantly, especially in object recognition in various background conditions. Because by using this technique, all of the object's backgrounds can be removed so that the recognition algorithm will extract only the object's features without the background. The difference between conventional labelling techniques and using landmarking can be observed in Figure 4.4. For the third method, the combination labelling technique is proposed in this work. This technique combines conventional imaging with landmarking, which will be explained in detail in Section 4.3.4.

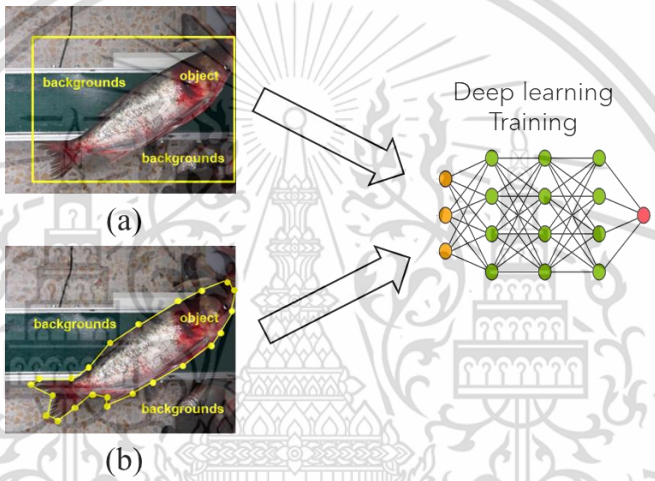


Figure 4.4 Labelling techniques in training process: (a) conventional and (b) using landmarking

Table 4.4 Summary of recent works by authors that are related to this work

References	Fish Object	Important Findings Related to this Work
[23]	- Static pictures. 6 classes. - Constant backgrounds. - Aquaculture fish with similar appearance and structurally deformed. - Taken from the Fish-Pak dataset.	Applying YOLOv4 with conventional labelling resulted in 14.29% higher accuracy than using landmarking for 6 classes.
[70]	- Static pictures. 4 classes. - Various backgrounds. - Ocean fish images captured in various background conditions, such as rocks, water, seaweed, etc. - Taken from the BYU dataset.	Combining YOLOv4 with landmarking labelling techniques resulted in 4.94% higher accuracy than using conventional.

4.2.4 YOLOv4, YOLOv4-Tiny, and The Training Process

In this work, YOLO (You Only Look Once) is employed as the recognition algorithm. This algorithm is very popular and known as a real-time object detector because of its speed and accuracy [71-75]. In addition, the version used is the relatively new, YOLO version 4, which was released in April 2020. We use this version because it accommodates our resources (currently, our resources only support this version), and we suppose that this version will achieve good performance with proper optimization. YOLOv4 consists of CSPDarknet53 as the backbone, SPP (Spatial Pyramid Pooling layer) & PAN (Path Aggregation Network) as the neck, and YOLOv3 as the head. A simple architecture of YOLOv4 is shown in Figure 4.5, and the output of this algorithm can be represented as (4.1) [55]. The detailed architecture or the layers of the YOLOv4 are described in Appendix A.

$$y = (P_c, B_y, B_x, B_w, B_h, C_1, C_2, \cdots C_8) \tag{4.1}$$

where y is the output of the YOLO, *Pc* will become 1 if the algorithm detects objects (fish) and 0 if otherwise, (B_y, B_x) is the center point of the produced bounding boxes of the fish, (B_w, B_h) is the width and height of the bounding boxes, and C_1 untill C_8 represent each class for the fish [55].

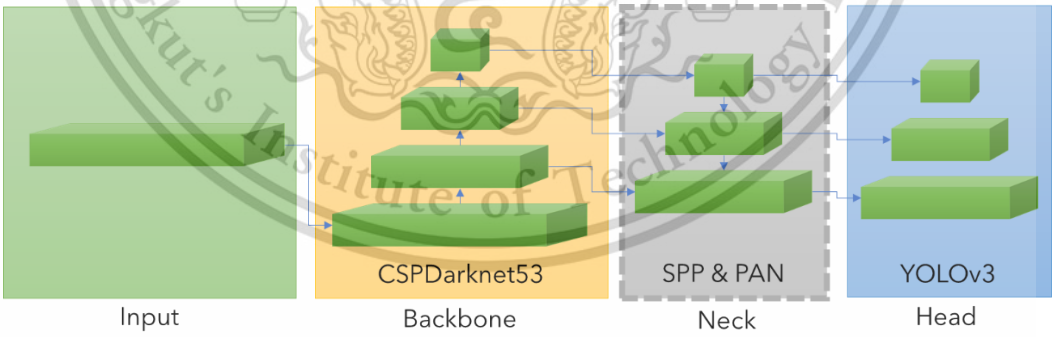


Figure 4.5 Simple architecture of YOLOv4 [55]

In this work, YOLOv4-Tiny is also used for study and comparison. YOLOv4-Tiny is a compressed and lite version of YOLOv4. The purpose of this compression is to reduce the computation so it can run on hardware with lower capacities, even for mobile or embedded devices. In this way, economic reasons can also be achieved. In

addition, this algorithm has a higher speed but is less accurate than the full version (YOLOv4). In its research, YOLOv4-Tiny achieved 22.0% AP (average precision) or 42.0% AP50 at a speed of 443 FPS (frames per second), while YOLOv4 was able to achieve 43.5% AP or 65.7% AP50 at a real-time speed of 65 FPS for the MS COCO dataset. According to these results, YOLOv4-Tiny is approximately seven times faster than YOLOv4 but only has 2/3 of the accuracy. It makes YOLOv4-Tiny more suitable for cases that require high detection speed, unnecessarily high accuracy, applied to hardware with lower capacities (economic reasons), or for mobile or embedded devices [76]. YOLOv4-Tiny reduces the layers of some components of the original YOLOv4 to achieve a faster detection speed. First and foremost, the number of layers in the CSP backbone is reduced from 137 to only 29 pre-trained convolutional layers. In addition, YOLOv3 reduces the head from 3 to 2, and there are only a few anchor boxes for prediction [76].

These recognition algorithms were executed on CiRA-Core, a deep learning platform originally developed by Advanced Manufacturing Innovation (AMI) KMITL and first described in [59, 60]. The advantages of this platform include its ease of use, user-friendliness, plethora of interface options, and provision of several automated processes, including during training, which will automatically select the most effective parameters for optimal results. In the training process, both YOLOv4 and YOLOv4-Tiny delivered good accuracy, between 0.15 and 0.03. The training was carried out with a batch size of 64 and 16 subdivisions, a learning rate of 0.001, and an adam optimizer. Data enrichment was carried out using rotation techniques, with a setting of 90 images per rotation (360°). The hardware used was a desktop PC with an Intel® 1151 Core™ i7-9700 3.0 GHz CPU (Central Processing Unit), NVIDIA GeForce RTX 3070 8 GB GDDR6 GPU (Graphical Processing Unit), and 32 GB DDR4/3200 RAM (Random Access Memory). Training time for each scheme took approximately 4–6 h.

4.2.5 Validation Matrix

The model performance in this work is also evaluated with a confusion matrix. The confusion matrix consists of 4 building blocks, as comprehensively explained in Chapter 2.2.4. From this confusion matrix, accuracy and other performance parameters are obtained, which are used to evaluate the model in this work: precision, recall/sensitivity, specificity, and F-score, denoted by (2.6) to (2.11).

In this work, the model's output is also categorized into three groups: correct detection, false detection (which includes wrong and double detection), and not-detect. Correct detection is defined as the detection and classification of the model that can be carried out correctly and consistently while the fish is fully visible in the frame. The wrong detection is determined if the model detects the wrong fish class more than once for more than 1 s. Double detection is specified if another fish class is also detected and appears more than once for more than 1 s. Not-detect is counted if the model cannot detect at all, can detect only momentarily (less than 1 s even though several times), or is unstable with a break of more than once for more than 1 s. The accuracy can be defined from a comparison between the number of correct detections and the total number of detections, as expressed in (2.7).

4.3 Experimental Results and Discussion

After the image data for training was prepared, the algorithm was trained with various image data input schemes, labelling techniques, and network versions to determine which approach was the most effective, including the proposed method. The trained algorithm was then applied to two high-speed videos, 1 and 2, for evaluation (also referred to as video tests after). The entire flow of this work can be seen in Figure 4.6, while the experimental results are summarized in Table 4.5 and 4.6, Figures 4.7–4.9. The following is the explanation and discussion for every test result's scheme.

Table 4.5 Experimental results (accuracy)

Approches	Video Test-1					Video Test-2					Average (Final Accuracy) (%)
	Correct Detection	False (Double/Wrong) Detection	Not Detect	Total Detection	Accuracy (%)	Correct Detection	False (Double/Wrong) Detection	Not Detect	Total Detection	Accuracy (%)	
YOLOv4 with Static Pics	2	0	69	71	2.82	11	0	160	171	6.43	4.62
YOLOv4-Tiny	67	4	0	71	94.37	154	17	0	171	90.06	92.21
YOLOv4	69	1	1	71	97.18	156	7	8	171	91.23	94.21
YOLOv4 + LM	68	6	0	74	91.89	159	11	1	171	92.98	92.44
Proposed method	72	0	1	73	98.63	167	4	0	171	97.66	98.15

Table 4.6 Experimental results (other performance parameters)

Approches	Video Test 1 and 2						
	Correct Detection (TP)	Wrong/Double Detection (FP)	Not Detect (FN)	Total Detection	Precision (%)	Sensitivity (%)	F-Score (%)
YOLOv4 With Static Pics	13	0	229	242	100.00	5.37	10.20
YOLOv4-Tiny	221	21	0	242	91.32	100.0	95.46
YOLOv4	225	8	9	242	96.57	96.15	96.36
YOLOv4 + LM	227	17	1	245	93.03	99.56	96.19
Proposed Method	239	4	1	244	98.35	99.58	98.96

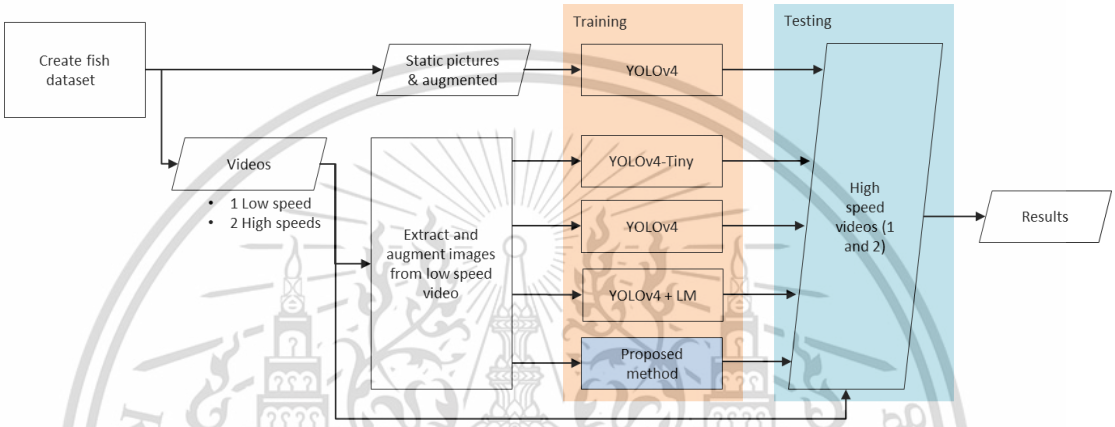
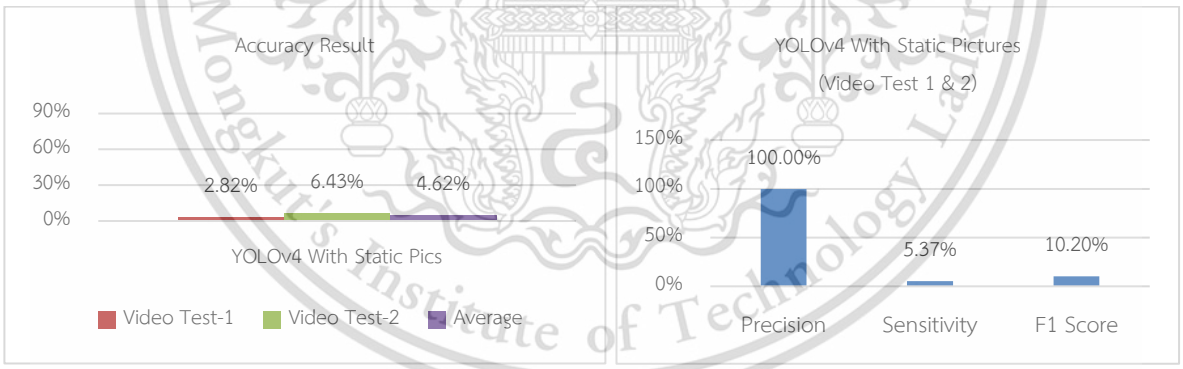


Figure 4.6 The work flowchart



(a) (b)

Figure 4.7 Test results of YOLOv4 with static pitures: (a) accuracy and (b) other parameters

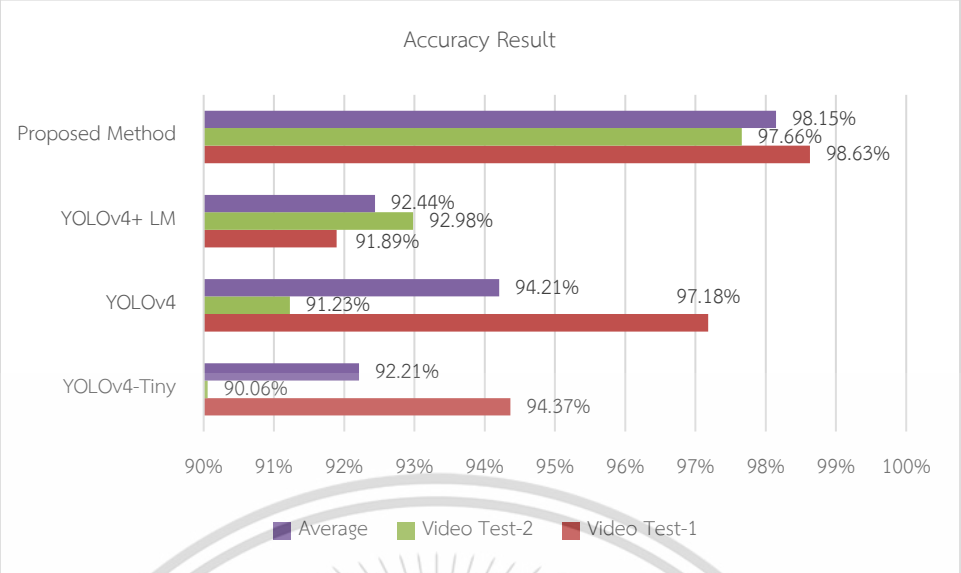


Figure 4.8 Experimental results (accuracy)

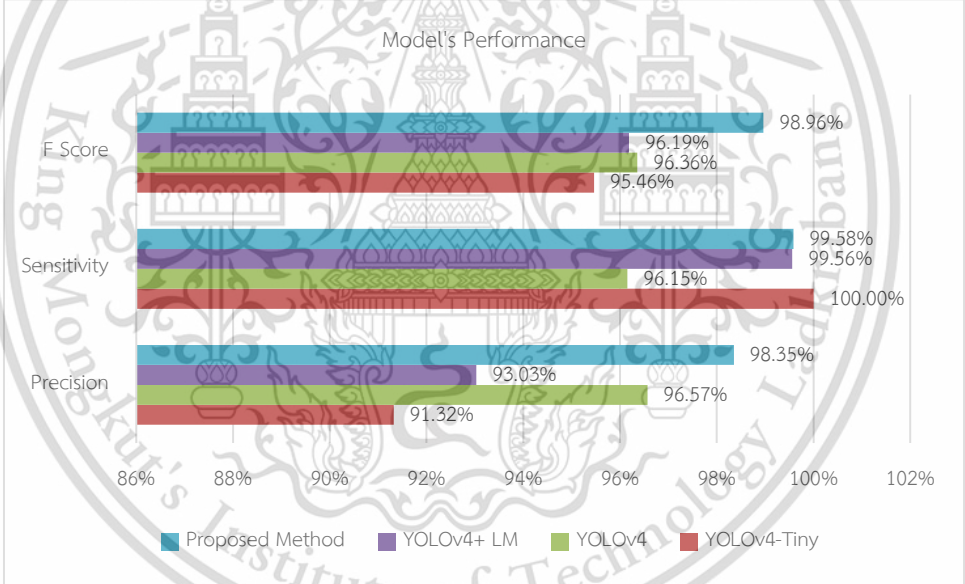


Figure 4.9 Experimental results (other performance parameters)

4.3.1 Using Static Pictures for Training Data

In this scheme, static images were used as training data in YOLOv4. The use of static images for this training is described in Section 4.2.2. Then, the algorithm that had been trained was tested on the video tests. From the experimental results, the output accuracy obtained was very low. In video test 1, the model could only correctly detect 2 of a total of 71 fish/detections. This means that the accuracy obtained is only 2.82%. In video test 2, the model could correctly detect 11 of 171 fish/detections, so the

accuracy obtained is only 6.43%. The final average accuracy of this model (the average accuracy of video tests 1 and 2) is only 4.62%.

Certainly, the output accuracy result of this scheme is unacceptable. It indicates that using static images as training data for the algorithm, which is then used to recognize different images, is very ineffective. Although humans are able to classify fish species on static images, the learning process in the algorithm cannot capture the same thing. Static images have different sizes than the video tests used. The results of the accuracy and other performance parameters of this scheme can be seen in Figure 4.7.

4.3.2 Using Static Pictures for Training Data

For the next step, the extracted pictures were used as training image data. Extracted pictures were obtained as described in Section 4.2.2. In this scheme, the lite version of YOLOv4 (YOLOv4-Tiny) was employed. This algorithm could be well trained and then tested on the same video tests. The output accuracy results obtained from the model were good enough. In video test 1, the model only detected four fish incorrectly out of 71 fish/detections. In video test 2, the model could correctly detect 154 out of 171 fish/detections while making 17 detection errors. This gives an accuracy score on video tests 1 and 2 of 94.37% and 90.06%, respectively, so the average score of both is 92.21%.

Of all the errors, the model made a double classification, i.e., one type of fish was detected as two different fish. It often appears between Yeesok and Nuanchan, Nuanchan and Tapian, and Jeen Ban and Jeen To. These fish are very similar to each other.

4.3.3 With YOLOv4 Using Conventional and Landmarking Labelling Techniques

At this stage, extracted pictures were used as training image data, and we used YOLOv4 as the algorithm. The first experiment was carried out using conventional labelling techniques. With this model, the accuracy output was better. In video test 1, an accuracy score of 97.18% was obtained, and in video test 2, it was 91.23%, so the average final accuracy of this model could reach 94.21%.

Although the final accuracy is only slightly better than with YOLOv4-Tiny, the total number of false (double) detections is much lower. From 4 to only 1 in video test 1, and from 17 to only 7 in video test 2. This means YOLOv4 has a higher classification accuracy capability than the lite version (YOLOv4-Tiny). However, another problem arose: the detection failure occurred nine times, all of which were in the Sawai class.

In the labelling process for the next scheme, YOLOv4 will be combined with the landmarking technique. With this approach, the results obtained were 91.89% accuracy for video test 1 and 92.98% accuracy for video test 2, so the average final accuracy was 92.44%. This model did many double detections, especially for the Nuanchan and Tapian, which have the same appearance of scales but different shapes. Many double detections were also carried out for the Jeen Ban-Jeen To classes. In addition, the model also detected scattered fish outside the conveyor incorrectly. The three scattered Yeesok fish were detected as Nin.

4.3.4 With the Prophosed Approach

Up to this step, some of the findings from various trials can be highlighted:

1. Using extracted pictures as training data provided much more effective results than static pictures.
2. YOLOv4 provided better accuracy results than its lite version (YOLOv4-Tiny).
3. Using conventional labelling techniques on YOLOv4 gave fairly accurate detection results, even for fish classes that were similar, but many of them failed to detect the Sawai class.
4. Combining YOLOv4 with the landmark labelling technique provided a fairly accurate detection result. Still, it generated many double detections for similar classes of fish, mostly for Nuanchan and Tapian, as well as Jeen Ban and Jeen To.

We can focus on the YOLOv4 algorithm with extracted pictures as image training data, and this approach produced the best accuracy. Then, we focus on the labelling technique used. The Sawai class was detected poorly using YOLOv4 and conventional labelling technique. If observed, this type of fish is the biggest. When it appears on the

video, the size of this fish almost fills the entire frame. So that when extracted images of this Sawai fish are generated and used as a training image, utilizing conventional labelling techniques, many other objects in the fish's background will be included in the training process. These objects are not constant (not the same in every extracted picture), making the YOLOv4 algorithm less effective in capturing Sawai fish features because it mixes with other objects in the background [70].

In contrast to other smaller fish (seven other classes) such as Nin, Nuanchan, Tapian, and even Jeen Ban and Jeen To, even though the background is involved in the training process, the background tends to be constant (almost just a conveyor background). This condition, on the other hand, has a good effect on the learning process. The model only detects fish on the conveyor, so it does not detect fish scattered outside it. The model can better extract the shape features of the fish so that it can better distinguish between Nuanchan and Tapian fish, which have a similar appearance of scales, tails, and heads but can be distinguished by their shape.

YOLOv4 combined with the landmark labelling technique, resulted in lower classifiability for similar fish such as Nuanchan-Tapian and Jeen Ban-Jeen To. Because, as previously described, with this technique, the background of the fish is completely removed so that the algorithm is not affected by the background during the training process. The advantage of this approach is that the model can better detect Sawai fish because the algorithm is not affected by the background object, which is inconsistent. However, this makes the algorithm not good at extracting shape features. Hence, the model experiences a lot of false (double) detection for fish that should be able to be distinguished from their shapes, such as Nuanchan-Tapian and Jeen Ban-Jeen To. In addition, the model also detects fish that are outside the conveyor (scattered fish).

This hypothesis can be supported by visualizing the feature maps—the features captured by the algorithm during the learning process. The visualization of these feature maps can be seen in Figure 4.10 for the algorithm using the conventional labelling technique and in Figure 4.11 when using landmarking. The example taken is the same fish for easy comparison. The detailed python codes are described in Appendix B and publicly accessible on github (https://github.com/AriKus77/Fish-Recognition-for-Auto-Sorting--Feature-Extraction).

Figure 4.10 shows that the background is included in the training process and extracted by the algorithm (see the first convolution process (conv2d)). Even the

background is still carried over and becomes part of the extracted features in a deeper layer; pooling 1 (max_pooling2d), convolution 2 (conv2d_1), pooling 2 (max_pooling2d_1), and convolution 3 (conv2d_2). This means that the background also becomes one of the determinants or features that are considered when the algorithm runs to detect the fish. In addition, the features of the fish's shape are better because there is a background comparison.

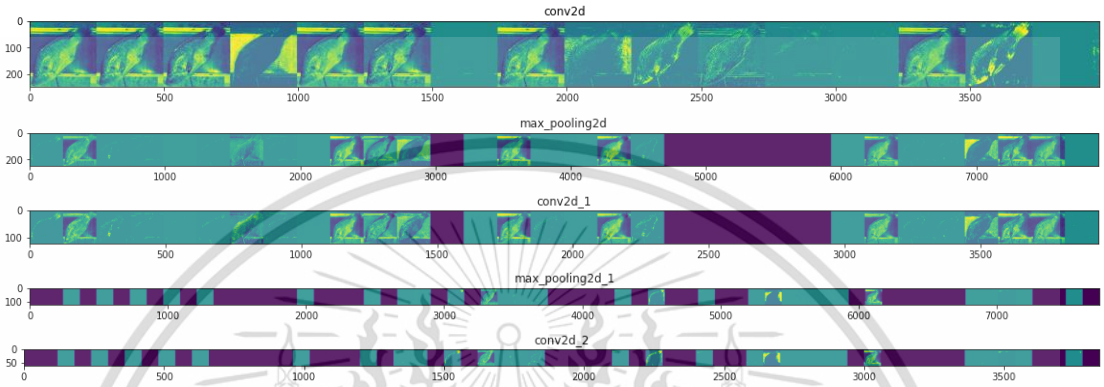


Figure 4.10 Feature maps visualization with conventional labelling

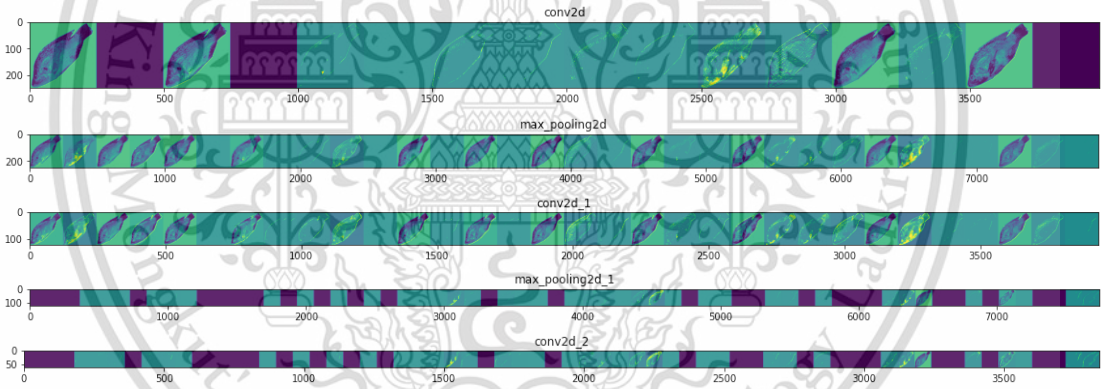


Figure 4.11 Feature maps visualization with landmarking labelling

On the other hand, the extraction process only happens on fish objects on feature maps made from the learning process on YOLOv4 and the landmark labelling technique. This is because the background of the fish is completely removed with this technique. It can be seen in Figure 4.11. Because the learning process only focuses on the fish object, it is not disturbed by other objects. However, the extraction of shape features is less good because there is no background for comparison. For that, an approach is proposed. This approach uses extracted images as training data, YOLOv4, and a combination of labelling techniques. The combined labelling technique uses the

landmarking technique for large fish (Sawai) and conventional for other classes. Then this approach is tested on the same video tests.

From the test results, the accuracy obtained was 98.63% for video test 1 and 97.66% for video test 2. This means that the average final accuracy was 98.15%. This achievement increased significantly compared to using the previous methods. There were only 4 false (double) detections, which had decreased a lot from the YOLOv4 with the landmark labelling technique (17 detections) and the conventional (8 detections). In addition, the detection failure was also significantly reduced to only one from the previous nine (using YOLOv4 with conventional labelling techniques). In other words, this approach can reach its optimum point by combining existing labelling techniques and avoiding each of their weaknesses.

4.3.5 Comparison with Recent State of Art

The method proposed in this work is compared to the current state of the art in order to identify its most significant contributions. Recent years have seen the development of at least six studies on recognizing swimming fish, which were also discussed in the Section 1. Table 4.7 provides a comparison summary. The table demonstrates that the proposed Method with simple and effective algorithms has the best average recognition results and has been tested on aquaculture fish video datasets; therefore, this work is most applicable to fish sorting systems in the fish industry.

Table 4.7 Comparison chart of previously related works with the proposed method

Related Work	Fish Dataset	Fish Type	No. of Fish Classes	Method/ Algorithm	Accuracy (%)	Precision (%)	Sensitivity (%)	F Score (%)	Advantage	Disadvantage
[7]	own dataset	deep ocean fish	20	CNN	94.90	-	-	-	(1) The best accuracy is achieved by recognizing nine species of fish. (2) Recognition results are accurate even though the backgrounds are various or the fish only partially appear.	(1) The results of accuracy can still be increased. (2) High accuracy is expected to apply to all classes.
[8]	own dataset	deep ocean fish	1	Multi-cascade object detection Network, 7 CNNs 2 RPNs, trained LSTMs	-	67.28	68.25	67.76	Promising to detect and count fish under various benthic backgrounds and illumination conditions.	Only detect fish, not classify them.
[9]	Fish4-Knowledge & UWA	deep ocean fish	17	Optical flow, GMM, ResNet-50, YOLOv3	91.64	-	-	95.47	Quite effective, even applied to many classes with diverse backgrounds and illumination challenges.	The results of accuracy can still be improved
[15]	own dataset	Aquacultured fish	1	Image enhancement, YOLOv3	100	-	-	-	(1) Effectively detect all fish in the test images. (2) Image Enhancement can optimize the work of the algorithm significantly.	Only to detect fish and trajectory, not for classification.
[16]	own dataset	Aquacultured fish	1	Faster R-CNN, YOLOv3	98.13	-	-	-	(1) High accuracy is obtained from Faster R-CNN. (2) Simple with good results	Only to detect fish and trajectory, not for classification.
[30]	Fish4-Knowledge	deep ocean fish	1	GMM, Pixel-wise posteriors, CNN	-	-	-	87.44	Increased the result fairly from the previous work.	(1) Only detect fish, not classify them. (2) The result can still be improved
Proposed method	own dataset	Aquacultured fish	8	Optimized YOLOv4	98.15	98.35	99.58	98.96	(1) Simple method but delivers high results. (2) Ready to implement for aquaculture fish sorting system.	(1) Using not open access deep learning software. (2) YOLOv4 can not be modified.

-: not reported.

4.3. Limitations and Future Developments

Limitations of this work also need to be reported. The biggest limitation is that the algorithm is standard and cannot be modified with the tools used in this work. So optimization studies cannot be carried out within the scope of modifications to the YOLOv4 or YOLOv4-Tiny algorithms. For this reason, this limitation can also be developed in the future. Optimization studies can be implemented by modifying them to achieve more optimal accuracy output or more efficient computations by the other tools. Or the use of a higher version (YOLO versions 5, 6, 7, or 8) can also be considered. In addition, future developments can also be conducted using image processing techniques such as reducing glare on the appearance of fish objects or combining approaches with other classification algorithms or unsupervised learning. A tuning or inference method can be proposed to optimize the accuracy of the approach in this work. Moreover, a statistical hypothesis analysis could be conducted.

4.4 Summary

This work aims to propose an optimal approach for detecting and classifying fish intended for the aquaculture industry, especially for making automatic sorting processes. In this work, we created and presented a real video dataset of freshwater fish running on a conveyor, which is the first and only, as far as the authors know. The dataset includes eight types of freshwater fish that are grown and eaten most often in Thailand and nearby countries. Some of these fish are native to Thailand. This work uses YOLOv4, the most viral algorithm for object detection, and a relatively new version. Several studies were conducted to determine the level of accuracy, and finally, an approach was proposed.

This approach utilizes YOLOv4, optimized with a combination/custom labelling technique, and extracted images as training data. From the test results on the video of eight types of freshwater fish running on a conveyor with a total duration of 25 min and 37 s at a speed of 505.08 m/h, the model could produce an accuracy of 98.15%. These results are considered quite good and can even be improved in the future. By using real videos of freshwater fish running on a conveyor, this work is expected to contribute to the development of fish detection and classification, especially for the automatic sorting process in the fish industry, which is very close to real conditions.

Chapter 5
Conclusions And Recommendations

5.1 Conclusions

In recent decades, much research has been carried out on image classification and object detection, including for fish recognition. Many methods have been developed to recognize underwater fish, as well as in the aquaculture industry. Both have many challenges, including various backgrounds, similar fish visual appearance, deformed fish conditions, speed, and random fish positions. Many approaches have been proposed to address these challenges, and deep learning is the most popular and widely developed. Many deep learning algorithms are employed, such as algorithms based on CNN, SSD, Alex-Net, VGGNet, ResNet, Inception, GoogLeNet, YOLO, etc. The overarching goal of this thesis is optimizing YOLO for fish classification and recognition, and the version used is version 4 (YOLOv4). YOLO is the most popular object detection algorithm known to be effective and rapidly developed today. However, it is rarely used, and many are not optimized for fish recognition. Various methods and studies were conducted in this thesis to optimize YOLOv4 for fish recognition to address all the challenges mentioned above. This chapter summarizes and concludes the work presented in this thesis and discusses some future directions.

The first work is presented in Chapter 2. We optimized YOLOv4 for fish classification in similar and structurally deformed conditions. This work aims to classify fish in aquaculture, which has that unique challenges. We utilized the Fish-Pak, a publicly available farming fish dataset of 6 species. The first optimization was carried out by enriching and balancing the training data for each class of fish. The dataset, which originally had an average value per class of 45 images and a Standard Deviation Average of 23.16, had become richer and balanced with an average value per class of 123 images and a Standard Deviation Average of 19.87. Applying YOLOv4 for classification on each whole image only produced an average accuracy of 43.01%. The next optimization study was carried out by combining YOLOv4 with several techniques, namely with the landmarking technique on labelling, which was able to increase the average accuracy to 72.65%, with the subclassing technique so raised the average accuracy became 76.64%, by adding a data scale so improved the average accuracy

became 77.42%, by square labelling so increased the average accuracy to 86.94%, and finally with class elimination so boosted the final average accuracy became 98.75%.

Chapter 3 describes the second work of optimizing YOLOv4 for fish classification in various background conditions. In this work, we used the BYU dataset and utilized four species of fish from it, which have diverse background challenges. This dataset is also publicly available, and for the four classes of fish used, the background images of fish objects vary greatly, such as water, rocks, aquatic plants, mixtures, etc. We also did the first optimization as in the first work, namely enriching and balancing the data for training with the augmentation process. After the augmentation process, the training data became richer and more balanced, which could be assessed from the average image per class and the Standard Deviation Average, which was 146 and 37.42 from the original values of 91 and 61.91. Then the YOLOv4 classification was optimized by combining it with the landmarking labelling technique, which increased the accuracy to 93.92%. This landmarking labelling technique was first introduced by the author for fish recognition, and this combination with YOLOv4 increased accuracy by 4.73% compared to conventional methods. Further optimization was obtained by adjusting the threshold to the most effective 60% so that the final average classification result became 96.60%.

The third and final work is explained in Chapter 4, optimizing YOLOv4 in moving fish recognition for the automatic sorting system. Unlike the previous works, this work detected and classified moving fish and is intended for an automatic sorting process because it was tested directly on videos of fish moving randomly on the conveyor. Since no public dataset is available for farmed fish that move randomly on the conveyor, the author created and published the dataset himself. This dataset is expected to contribute significantly because it is very useful for developing automatic aquaculture fish recognition based on computer or machine vision. The dataset contains eight types of cultivated fish which are widely consumed by the people of Thailand and its surroundings. Some of these fish species are also endemic to the Mekong and Chao Phraya rivers. Several YOLOv4 studies and optimizations were then conducted to find the most effective method to answer these challenges, starting from the use of training data, the YOLOv4 variant employed, and the labelling technique utilized to obtain the method with the highest accuracy. Using YOLOv4 with static images as training data delivered insufficient accuracy, only 4.62%. Using extracted

images as training data gave high results, amounting to 92.21% if using the lite version (YOLOv4-Tiny) and 94.21% using the original version (YOLOv4). Combining YOLOv4 with the landmarking labelling technique provided no better accuracy, only 92.44%. And finally, using YOLOv4 with a combination labelling technique delivered the best accuracy of 98.15%.

From the experimental test results of all works, it can be concluded that using YOLOv4 for fish recognition produces fairly good accuracy and can be optimized with certain techniques. It is proven from works in this thesis that it can produce more than 95% accuracy to overcome the previously described challenges. Hopefully, this thesis will contribute to the development of automatic fish recognition both under the sea and for the aquaculture industry. In addition, the aquaculture fish dataset moving on a conveyor created by the author himself, and the approach proposed to recognize it is expected to contribute a lot and become a solution for automatic sorting systems in the aquaculture industry because it is very close to actual conditions and is built on a ready-to-use technology platform.

5.2 Future Directions

The approaches proposed in this thesis can potentially be developed in the future. The YOLOv4 algorithm can be modified to improve accuracy. Image processing techniques can also be applied to optimize the features of the image before YOLOv4 extracts it. Several techniques can be used, including resampling or image enhancement, such as reducing glare, especially for work 3. Combination with other algorithms and even with unsupervised learning can also be considered. In addition, using higher YOLO versions and then optimized is also worth considering, such as versions 5, 6, 7 and even 8.

APPENDIX A

YOLOv4 Detail Of Layers

```
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=64
# subdivisions=16
width=608
height=608
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
learning_rate=0.001
burn_in=1000
max_batches = 500500
policy=steps
steps=400000,450000
scales=.1,.1

#cutmix=1
#mosaic=1

#:104x104  54:52x52  85:26x26  104:13x13  for 416

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
```

```
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-7

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

# Downsample
[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
```

```
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-10

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

# Downsample
[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
```

```
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish
```

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
```

```
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish
```

```
from=-3

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-28

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
```

```
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear
```

```
[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
```

```
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear
```

```
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-28

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

# Downsample
[convolutional]
batch_normalize=1
filters=1024
size=3
stride=2
pad=1
activation=mish

[convolutional]
```

```
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
```

```
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-16

[convolutional]
batch_normalize=1
filters=1024
size=1
stride=1
pad=1
activation=mish
stopbackward=800

####################
#######
```

```
[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

### SPP ###
[maxpool]
stride=1
size=5

[route]
```

```
layers=-2

[maxpool]
stride=1
size=9

[route]
layers=-4

[maxpool]
stride=1
size=13

[route]
layers=-1,-3,-5,-6
### End SPP ###

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
```

```
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = 85

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -1, -3

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = 54
```

```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -1, -3

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

##################
#######

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=39
activation=linear

[yolo]
mask = 0,1,2
anchors = 12, 16,
19, 36, 40, 28, 36,
75, 76, 55, 72, 146,
142, 110, 192, 243,
459, 401
classes=8
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
scale_x_y = 1.2
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5

[route]
```

```
layers = -4

[convolutional]
batch_normalize=1
size=3
stride=2
pad=1
filters=256
activation=leaky

[route]
layers = -1, -16

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=39
activation=linear

[yolo]
mask = 3,4,5
anchors = 12, 16,
19, 36, 40, 28, 36,
75, 76, 55, 72, 146,
142, 110, 192, 243,
459, 401
classes=8
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
scale_x_y = 1.1
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5

[route]
```

```
layers = -4

[convolutional]
batch_normalize=1
size=3
stride=2
pad=1
filters=512
activation=leaky

[route]
layers = -1, -37

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
```

```
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky
```

```
[convolutional]
size=1
stride=1
pad=1
filters=39
activation=linear

[yolo]
mask = 6,7,8
anchors = 12, 16,
19, 36, 40, 28, 36,
75, 76, 55, 72, 146,
142, 110, 192, 243,
459, 401
classes=8
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
scale_x_y = 1.05
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5
```

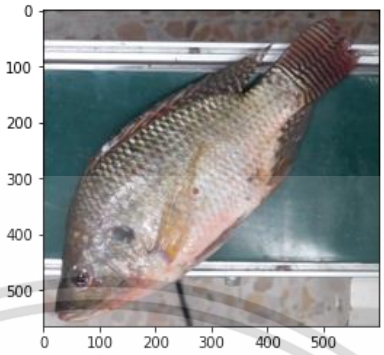
APPENDIX B

Python Scrips For Feature Maps Visualization

1. With the conventional labelling technique

```
#import the basic libaries
import tensorflow as tf
import cv2 #to do some image operations (cv2=OpenCv)
import os #to iterate through directories and join paths
import numpy as np #to do various array of operations (numpy=numerical
python)

#import plot to show the image
import matplotlib.pyplot as plt

#import image
from tensorflow.keras.preprocessing import image

#import image data generator to generate label automatically as the
folder's name where the image is
from tensorflow.keras.preprocessing.image import ImageDataGenerator

#loading images
img = image.load_img("basedata/training/Nin/1.PNG")#sample image

#showing the image
plt.imshow(img)
```

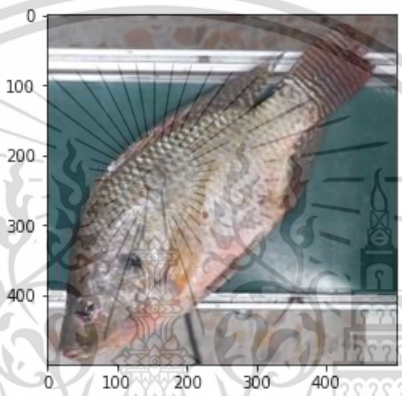
```
Out [] : <matplotlib.image.AxesImage at 0x2761952f220>
```




```
#showing the size of image
cv2.imread("basedata/training/Nin/1.PNG").shape  #will be shown in
high, width, channel
Out [] : (564, 600, 3)


#generating training and validation dataset with ImageDataGenerator
#rescale=1/255 will convert to grayscale. Because RGB has value 0-255
. By dividing with 255, the value will become 0 to 1
train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)


#convert the training images to fit the neural network (NN)
train_dataset = train.flow_from_directory('basedata/training',
                                          target_size = (500,500),
                                          batch_size = 3,)
#flow from directory will generate the label automatically from the
folder's name
#images feeded to the NN are can not be in various sizes, so should be
resized
#because the images are not too much, batch size of 3 is considered
enough
validation_dataset = train.flow_from_directory('basedata/validation',
```

```
                                                target_size = (500,500),
                                                batch_size = 3,)

#define the model

model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),a
ctivation = 'relu',input_shape = (500,500,3)),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    #
                                    tf.keras.layers.Conv2D(32,(3,3),a
ctivation = 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    #
                                    tf.keras.layers.Conv2D(64,(3,3),a
ctivation = 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    ##
                                    tf.keras.layers.Flatten(),
                                    ##
                                    tf.keras.layers.Dense(512,activat
ion = 'relu'),
                                    ##
                                    tf.keras.layers.Dense(1,activatio
n = 'sigmoid')
                                   ])



#compiling model

model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=

['accuracy'])



#fit up the model and start training

model_fit = model.fit(train_dataset,
                      steps_per_epoch = 3,
                      epochs = 30,
                      validation_data = validation_dataset)

import random
from tensorflow.keras.preprocessing.image import img_to_array, load_i

mg



# Get list of layers from model

layer_outputs = [layer.output for layer in model.layers[1:]]

#it means we will get the layer list from the first ntill the last



# Create a visualization model
```

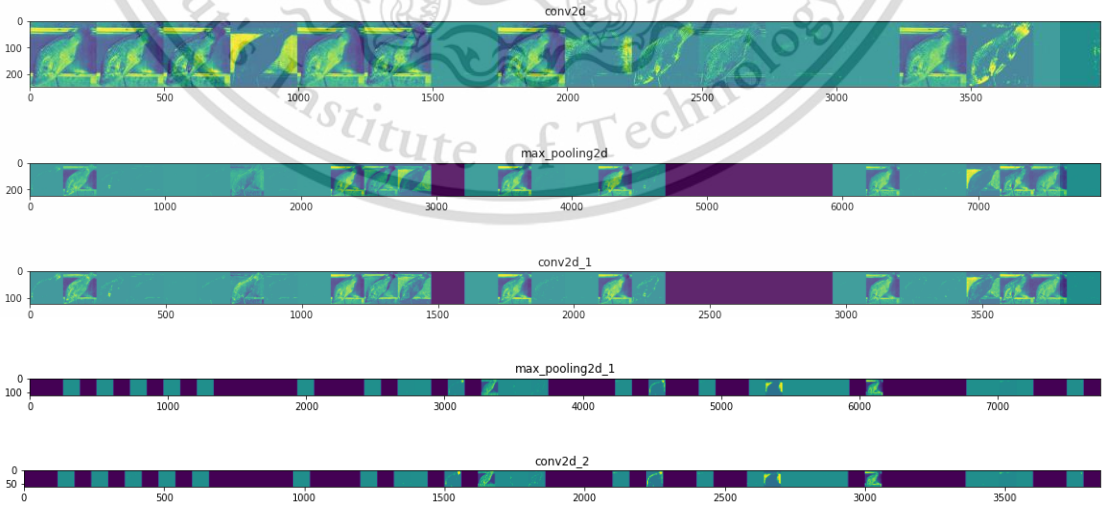
```
visualize_model = tf.keras.models.Model(inputs = model.input, outputs
 = layer_outputs)


# Load image for prediction

img=load_img('basedata/training/Nin/1.PNG',target_size=(500,500))


#showing the image

plt.imshow(img)
Out [] :
```




```
# Convert image to array

x = img_to_array(img)



# Print shape of array

x.shape


Out [] : (500, 500, 3)


# Reshape image for passing it to prediction

x=x.reshape((1,500,500,3))
print(x.shape)
Out [] : (1, 500, 500, 3)

# Rescale the image

x = x /255 #because when training was also rescaled



# Get all layers feature maps for image

feature_maps=visualize_model.predict(x)
```

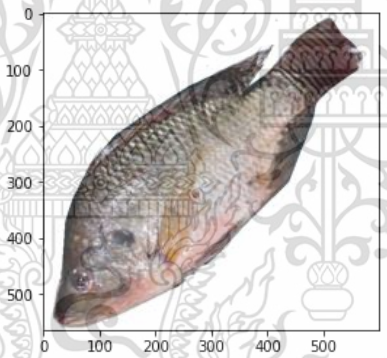
```
print(len(feature_maps))

# Show names of layers available in model

layer_names = [layer.name for layer in model.layers]
print(layer_names)

# import required libraries

%matplotlib inline

# Plotting the graph

for layer_names, feature_maps in zip(layer_names,feature_maps):
  print(feature_maps.shape)
  if len(feature_maps.shape) == 4 :
    channels = feature_maps.shape[-1]
    size = feature_maps.shape[1]
    display_grid = np.zeros((size, size * channels))
    for i in range(channels):
      x = feature_maps[0, :, :, i]
      x -= x.mean()
      x /= x.std()
      x *= 64
      x += 128
      x = np.clip(x, 0, 255).astype('uint8')
   # We'll tile each filter into this big horizontal grid

      display_grid[:, i * size : (i + 1) * size] = x
    scale = 20. / channels
    plt.figure(figsize=(scale * channels, scale))
    plt.title(layer_names)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

Out[]:



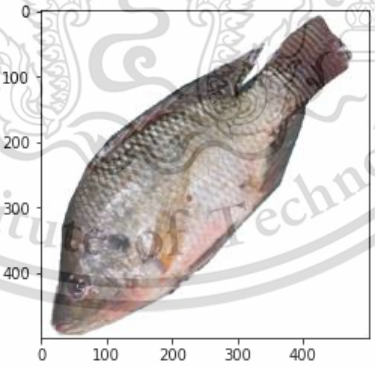
2. With the landmarking labelling technique

```
import tensorflow as tf

import cv2

import os

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator


img = image.load_img("basedata/training/Nin/1.PNG")

plt.imshow(img)

out []: <matplotlib.image.AxesImage at 0x1637ea899a0>
```




```
cv2.imread("basedata/training/Nin/1.PNG").shape
out []: (564, 600, 3)

train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)

train_dataset = train.flow_from_directory('basedata/training',
                                          target_size = (500,500),
                                          batch_size = 3,)
validation_dataset = train.flow_from_directory('basedata/validation',
                                          target_size = (500,500),
                                          batch_size = 3,)

model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),a
ctivation = 'relu',input_shape = (500,500,3)),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    #
                                    tf.keras.layers.Conv2D(32,(3,3),a
ctivation = 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
```

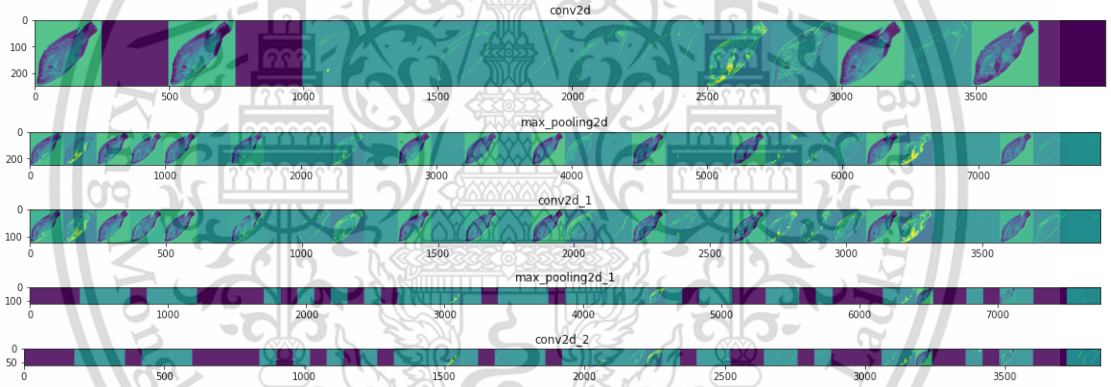
```
                                    #
                                    tf.keras.layers.Conv2D(64,(3,3),a
ctivation = 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    ##
                                    tf.keras.layers.Flatten(),
                                    ##
                                    tf.keras.layers.Dense(512,activat
ion = 'relu'),
                                    ##
                                    tf.keras.layers.Dense(1,activatio
n = 'sigmoid')
                                   ])

model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=
['accuracy'])

model_fit = model.fit(train_dataset,
                      steps_per_epoch = 3,
                      epochs = 30,
                      validation_data = validation_dataset)

import random
from tensorflow.keras.preprocessing.image import img_to_array, load_i
mg

layer_outputs = [layer.output for layer in model.layers[1:]]

visualize_model = tf.keras.models.Model(inputs = model.input, outputs
 = layer_outputs)

img=load_img('basedata/training/Nin/1.PNG',target_size=(500,500))
plt.imshow(img)
out []:
```




```
x = img_to_array(img)

x = x /255

feature_maps=visualize_model.predict(x)
print(len(feature_maps))

%matplotlib inline


for layer_names, feature_maps in zip(layer_names,feature_maps):
```

```
print(feature_maps.shape)
if len(feature_maps.shape) == 4 :
  channels = feature_maps.shape[-1]
  size = feature_maps.shape[1]
  display_grid = np.zeros((size, size * channels))
  for i in range(channels):
    x = feature_maps[0, :, :, i]
    x -= x.mean()
    x /= x.std()
    x *= 64
    x += 128
    x = np.clip(x, 0, 255).astype('uint8')
 # We'll tile each filter into this big horizontal grid

    display_grid[:, i * size : (i + 1) * size] = x
  scale = 20. / channels
  plt.figure(figsize=(scale * channels, scale))
  plt.title(layer_names)
  plt.grid(False)
  plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

Out[]:



AUTHOR BIOGRAPHY

Author: Mr. Ari Kuswantori

Degree: Doctor of Engineering

Date: June 23rd, 2023

Date of Birth: August 17th, 1989

Place of Birth: Lamongan, Indonesia

Undergraduate and Graduate Education:

Doctor of Engineering in Instrumentation and Control Engineering,

King Mongkut's Institute of Technology Ladkrabang, Bangkok, 2023

Master of Engineering in Electrical Engineering and Control System,

National Institute of Science and Technology, Jakarta, 2016

Bachelor of Engineering in Control and Instrumentation Engineering,

Mitra Karya University, Bekasi, 2013

Major: Instrumentation and Control Engineering

Presentations and Publications:

Kuswantori A., Suesut T., Tangsrirat W., and Satthamsakul S. "Fish Recognition Optimization in Various Backgrounds Using Landmarking Technique and YOLOv4". Proceedings of the 37th International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Phuket, Thailand, pp. 943-946, July 2022.

Kuswantori A., Suesut T., Tangsrirat W., and Nunak N. “Development of object detection and classification with YOLOv4 for similar and structural deformed fish”. EUREKA: Physics and Engineering, Vol. 2, pp. 154–165, 2022.

Kuswantori A., Suesut T., Tangsrirat W., Schleining G., and Nunak N. “Fish Detection and Classification for Automatic Sorting System with an Optimized YOLO Algorithm”. Applied Sciences, Vol. 13(6), pp. 3812, 2023.

S. Satthamsakul, A. Kuswantori, W. Sriratana, W. Tangsrirat, and T. Suesut. "Landmarking Technique for Improving YOLOv4 Fish Recognition in Various Background Conditions". Advances in Science, Technology and Engineering Systems Journal, vol. 8, no. 3, pp. 100-107 (2023).

Kuswantori A. “Memahami ANN, Deep Learning, CNN, dan YOLO”. Madza Media. 2022. Available on Google Books: https://play.google.com/store/books/details/Memahami_ANN_Deep_Learning_CNN_dan_YOLO?id=VI6EEAAAQBAJ&hl=en&gl=JP

REFERENCES

[1] D. Li, Q. Wang, X. Li, M. Niu, H. Wang, and C. Liu, "Recent advances of machine vision technology in fish classification," *ICES Journal of Marine Science,* vol. 79, pp. 263-284, 2022.

[2] M. K. Alsmadi and I. Almarashdeh, "A survey on fish classification techniques," *Journal of King Saud University - Computer and Information Sciences,* vol. 34, pp. 1625-1638, 2020.

[3] S. Zhao, S. Zhang, J. Liu, H. Wang, J. Zhu, D. Li*, et al.*, "Application of machine learning in intelligent fish aquaculture: A review," *Aquaculture,* vol. 540, p. 736724, 2021.

[4] A. Saleh, M. Sheaves, and M. Rahimi Azghadi, "Computer vision and deep learning for fish classification in underwater habitats: A survey," *Fish and Fisheries,* vol. 23, pp. 977-999, 2022.

[5] D. Li and L. Du, "Recent advances of deep learning algorithms for aquacultural machine vision systems with emphasis on fish," *Artificial Intelligence Review,* vol. 55, pp. 4077-4116, 2022.

[6] X. Yang, S. Zhang, J. Liu, Q. Gao, S. Dong, and C. Zhou, "Deep learning for smart fish farming: applications, opportunities and challenges," *Reviews in Aquaculture,* vol. 13, pp. 66-90, 2021.

[7] S. Villon, D. Mouillot, M. Chaumont, E. S. Darling, G. Subsol, T. Claverie*, et al.*, "A Deep learning method for accurate and fast identification of coral reef fishes in underwater images," *Ecological Informatics,* vol. 48, pp. 238-244, 2018/11/01/ 2018.

[8] A. B. Labao and P. C. Naval, "Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild," *Ecological Informatics,* vol. 52, pp. 103-121, 2019/07/01/ 2019.

[9] A. Jalal, A. Salman, A. Mian, M. Shortis, and F. Shafait, "Fish detection and species classification in underwater environments using deep learning with temporal information," *Ecological Informatics,* vol. 57, p. 101088, 2020/05/01/ 2020.

[10] S. Villon, C. Iovan, M. Mangeas, T. Claverie, D. Mouillot, S. Villéger*, et al.*, "Automatic underwater fish species classification with limited data using few-shot learning," *Ecological Informatics,* vol. 63, p. 101320, 2021/07/01/ 2021.

[11] Z. Ju and Y. Xue, "Fish species recognition using an improved AlexNet model," *Optik,* vol. 223, p. 165499, 2020/12/01/ 2020.

[12] A. Salman, S. Maqbool, A. H. Khan, A. Jalal, and F. Shafait, "Real-time fish detection in complex backgrounds using probabilistic background modelling," *Ecological Informatics,* vol. 51, pp. 44-51, 2019.

[13] A. N.S, S. D, and R. K. S, "Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries," *Ecological Informatics,* vol. 61, p. 101248, 2021/03/01/ 2021.

[14] M. S. Ahmed, T. T. Aurpa, and M. A. K. Azad, "Fish Disease Detection Using Image Based Machine Learning Technique in Aquaculture," *Journal of King Saud University - Computer and Information Sciences,* vol. 34, pp. 5170-5182, 2021.

[15] H. E.-D. Mohamed, A. Fadl, O. Anas, Y. Wageeh, N. ElMasry, A. Nabil*, et al.*, "MSR-YOLO: Method to Enhance Fish Detection and Tracking in Fish Farms," *Procedia Computer Science,* vol. 170, pp. 539-546, 2020.

[16] W. Xu, Z. Zhu, F. Ge, Z. Han, and J. Li, "Analysis of Behavior Trajectory Based on Deep Learning in Ammonia Environment for Fish," *Sensors,* vol. 20, p. 4425, 2020.

[17] A. Waleed, H. Medhat, M. Esmail, K. Osama, R. Samy, and T. M. Ghanim, "Automatic Recognition of Fish Diseases in Fish Farms," presented at the 2019 14th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 17-17 December 2019.

[18] N. Ubina, S.-C. Cheng, C.-C. Chang, and H.-Y. Chen, "Evaluating fish feeding intensity in aquaculture with convolutional neural networks," *Aquacultural Engineering,* vol. 94, p. 102178, 2021/08/01/ 2021.

[19] F. Bader and S. Rahimifard, "Challenges for Industrial Robot Applications in Food Manufacturing," presented at the ISCSIC '18: Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, Stockholm, Sweden, 21-23 September 2018.

[20] N. F. F. Alshdaifat, A. Z. Talib, and M. A. Osman, "Improved deep learning framework for fish segmentation in underwater videos," *Ecological Informatics,* vol. 59, p. 101121, 2020/09/01/ 2020.

[21] A. Hollowed, M. Barange, K. Brander, K. Cochrane, K. Drinkwater, M. Foreman*, et al.*, "Projected impacts of climate change on marine fish and fisheries," *ICES Journal of Marine Science,* vol. 70, pp. 1023–1037, 07/06 2013.

[22] A. N.S, S. D, and R. K. Sidharthan, "Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries," *Ecological Informatics,* vol. 61, p. 101248, 2021/03/01/ 2021.

[23] A. Kuswantori, T. Suesut, W. Tangsrirat, and N. Nunak, "Development of object detection and classification with YOLOv4 for similar and structural deformed fish," *EUREKA: Physics and Engineering,* vol. 2, pp. 154–165, 2022a.

[24] H. Qin, X. Li, J. Liang, Y. Peng, and C. Zhang, "DeepFish: Accurate underwater live fish recognition with a deep architecture," *Neurocomputing,* vol. 187, pp. 49-58, 2016.

[25] H. Park and S. Kim, "Chapter Three - Hardware accelerator systems for artificial intelligence and machine learning," in *Advances in Computers*. vol. 122, S. Kim and G. C. Deka, Eds., ed: Elsevier, 2021, pp. 51-95.

[26] A. A. Dos Santos and W. N. Gonçalves, "Improving Pantanal fish species recognition through taxonomic ranks in convolutional neural networks," *Ecological Informatics,* vol. 53, p. 100977, 2019/09/01/ 2019.

[27] T. Miyazono and T. Saitoh, "Fish Species Recognition Based on CNN Using Annotated Image," 2018.

[28] R. B S, D. S. G N, S. Reddy, D. Kakwani, and N. Bhattad, "Fish detection and classification using convolutional neural networks," ed, 2020, pp. 1221-1231.

[29] S. C. Mana and T. Sasipraba, "Automated fish detection and tracking system using pre-trained Mask R-CNN for Ecological biodiversity," ed: Research Square, 2022.

[30] A. Salman, S. A. Siddiqui, F. Shafait, A. Mian, M. R. Shortis, K. Khurshid*, et al.*, "Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system," *ICES Journal of Marine Science,* vol. 77, pp. 1295-1307, 2020.

[31] L. Xiu, S. Min, H. Qin, and C. Liansheng, "Fast accurate fish detection and recognition of underwater images with Fast R-CNN," in *OCEANS 2015 - MTS/IEEE Washington*, 2015, pp. 1-5.

[32] Y. Adiwinata, A. Sasaoka, I. A. Bayupati, and O. Sudana, "Fish species recognition with Faster R-CNN Inception-v2 using QUT FISH dataset," *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi,* vol. 11, pp. 144-154, 2020.

[33] M. A. Rosales, M. G. B. Palconit, V. J. D. Almero, R. S. Concepcion, J.-A. V. Magsumbol, E. Sybingco*, et al.*, "Faster R-CNN based Fish Detector for Smart Aquaculture System," in *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2021, pp. 1-6.

[34] A. Ben Tamou, A. Benzinou, and K. Nasreddine, "Multi-stream fish detection in unconstrained underwater videos by the fusion of two convolutional neural network detectors," *Applied Intelligence,* vol. 51, pp. 5809-5821, 2021/08/01 2021.

[35] X. Li, M. Shang, J. Hao, and Z. Yang, "Accelerating fish detection and recognition by sharing CNNs with objectness learning," in *OCEANS 2016 - Shanghai*, 2016, pp. 1-5.

[36] P. D. Hung and N. N. Kien, "SSD-Mobilenet Implementation for Classifying Fish Species," in *Intelligent Computing and Optimization*, Cham, 2020, pp. 399-408.

[37] G. Yu, L. Wang, M. Hou, Y. Liang, and T. He, "An adaptive dead fish detection approach using SSD-MobileNet," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 1973-1979.

[38] G. Tian, D. Li, W. Li, L. Zhang, H. Zhang, and Q. Duan, "A detection method of the turned white belly fish based on improved SSD," *Journal of Physics: Conference Series,* vol. 1856, p. 012035, 2021/04/01 2021.

[39] M. A. Iqbal, Z. Wang, Z. A. Ali, and S. Riaz, "Automatic Fish Species Classification Using Deep Convolutional Neural Networks," *Wireless Personal Communications,* vol. 116, pp. 1043-1053, 2021/01/01 2021.

[40] A. B. Tamou, A. Benzinou, K. Nasreddine, and L. Ballihi, "Underwater Live Fish Recognition by Deep Learning," in *Image and Signal Processing*, Cham, 2018, pp. 275-283.

[41] T. A. B, B. A, N. K, and B. L, "Transfer Learning with deep Convolutional Neural Network for Underwater Live Fish Recognition," in *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, 2018, pp. 204-209.

[42] E. Prasetyo, N. Suciati, and C. Fatichah, "Multi-level residual network VGGNet for fish species classification," *Journal of King Saud University - Computer and Information Sciences,* 2021/06/05/ 2021.

[43] Z. Zheng, C. Guo, X. Zheng, Z. Yu, W. Wang, H. Zheng, *et al.*, "Fish Recognition from a Vessel Camera Using Deep Convolutional Neural Network and Data Augmentation," in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, 2018, pp. 1-5.

[44] H. T. Rauf, M. I. U. Lali, S. Zahoor, S. Z. H. Shah, A. U. Rehman, and S. A. C. Bukhari, "Visual features based automated identification of fish species using deep convolutional neural networks," *Computers and Electronics in Agriculture,* vol. 167, p. 105075, 2019/12/01/ 2019.

[45] D. Gomes and A. S. Saif, "Robust Underwater Fish Detection Using an Enhanced Convolutional Neural Network," 2021.

[46] A. K. Agarwal, R. G. Tiwari, V. Khullar, and R. K. Kaushal, "Transfer Learning Inspired Fish Species Classification," in *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2021, pp. 1154-1159.

[47] M. Mathur and N. Goel, "FishResNet: Automatic Fish Classification Approach in Underwater Scenario," *SN Computer Science,* vol. 2, p. 273, 2021/05/14 2021.

[48] X. Xu, W. Li, and Q. Duan, "Transfer learning and SE-ResNet152 networks-based for small-scale unbalanced fish species identification," *Computers and Electronics in Agriculture,* vol. 180, p. 105878, 2021/01/01/ 2021.

[49] S. Choi, "Fish Identification in Underwater Video with Deep Convolutional Neural Network: SNUMedinfo at LifeCLEF Fish task 2015," in *CLEF (Working Notes)*, 2015.

[50] L. Meng, T. Hirayama, and S. Oyanagi, "Underwater-Drone With Panoramic Camera for Automatic Fish Recognition Based on Deep Learning," *IEEE Access,* vol. 6, pp. 17880-17886, 2018.

[51] 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, p. 102117, 2020/11/01/ 2020.

[52] S. Z. H. Shah, H. T. Rauf, M. IkramUllah, M. S. Khalid, M. Farooq, M. Fatima*, et al.*, "Fish-Pak: Fish species dataset from Pakistan for visual features based classification," in *Data in Brief* vol. 27, D. i. Brief, Ed., ed: Data in Brief, 2019, p. 104565.

[53] S. A. Shammi, S. Das, M. Hasan, and S. R. H. Noori, "FishNet: Fish Classification using Convolutional Neural Network," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1-5.

[54] A. Banerjee, A. Das, S. Behra, D. Bhattacharjee, N. T. Srinivasan, M. Nasipuri*, et al.*, "Carp-DCAE: Deep convolutional autoencoder for carp fish classification," *Computers and Electronics in Agriculture,* vol. 196, p. 106810, 2022/05/01/ 2022.

[55] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv,* 2020.

[56] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016.

[57] K. D. Lillywhite, Lee, D.J. (2013, 19/09/21). *Robotic vision lab, Brigham Young University, Fish dataset*. Available: http://roboticvision.groups.et.byu.net/Machine_Vision/BYUFish/BYU_Fish.html

[58] Z. Liu, X. Jia, and X. Xu, "Study of shrimp recognition methods using smart networks," *Computers and Electronics in Agriculture,* vol. 165, p. 104926, 2019/10/01/ 2019.

[59] V. Kittichai, T. Pengsakul, K. Chumchuen, Y. Samung, P. Sriwichai, N. Phatthamolrat*, et al.*, "Deep learning approaches for challenging species and gender identification of mosquito vectors," *Scientific reports,* vol. 11, pp. 1-14, 2021.

[60] V. Kittichai, M. Kaewthamasorn, S. Thanee, R. Jomtarak, K. Klanboot, K. M. Naing*, et al.*, "Classification for avian malaria parasite Plasmodium gallinaceum blood

stages by using deep convolutional neural networks," *Scientific reports,* vol. 11, pp. 1-10, 2021.

[61] T. T. E. Vo, H. Ko, J.-H. Huh, and Y. Kim, "Overview of smart aquaculture system: Focusing on applications of machine learning and computer vision," *Electronics,* vol. 10, p. 2882, 2021.

[62] J. Gladju, B. S. Kamalam, and A. Kanagaraj, "Applications of data mining and machine learning framework in aquaculture and fisheries: A review," *Smart Agricultural Technology,* vol. 2, p. 100061, 2022.

[63] Y. Wu, R. Zhuang, and Z. Cui, "Automatic Sorting System of Large Yellow Croaker Based on Machine Vision," presented at the 2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), Shenzhen, China, 09-11 May 2019.

[64] S. Tappi, P. Rocculi, A. Ciampa, S. Romani, F. Balestra, F. Capozzi*, et al.*, "Computer vision system (CVS): A powerful non-destructive technique for the assessment of red mullet (Mullus barbatus) freshness," *European Food Research and Technology,* vol. 243, pp. 2225-2233, 2017.

[65] R. B. Fisher, Y.-H. Chen-Burger, D. Giordano, L. Hardman, and F.-P. Lin, *Fish4Knowledge: collecting and analyzing massive coral reef fish video data* vol. 104: Springer, 2016.

[66] S. A. Siddiqui, A. Salman, M. I. Malik, F. Shafait, A. Mian, M. R. Shortis*, et al.*, "Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data," *ICES Journal of Marine Science,* vol. 75, pp. 374-389, 2018.

[67] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data,* vol. 6, pp. 1-48, 2019.

[68] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv,* 2017.

[69] T. Bouwmans, S. Javed, M. Sultana, and S. K. Jung, "Deep neural network concepts for background subtraction: A systematic review and comparative evaluation," *Neural Networks,* vol. 117, pp. 8-66, 2019.

[70] A. Kuswantori, T. Suesut, W. Tangsrirat, and S. Satthamsakul, "Fish Recognition Optimization in Various Backgrounds Using Landmarking Technique and

YOLOv4," presented at the The 37th International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Phuket, Thailand, 05-08 July 2022.

[71] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo algorithm developments," *Procedia Computer Science,* vol. 199, pp. 1066-1073, 2022.

[72] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications,* pp. 1-33, 2022.

[73] R. Chandana and A. Ramachandra, "Real Time Object Detection System with YOLO and CNN Models: A Review," *arXiv,* 2022.

[74] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems,* vol. 30, pp. 3212-3232, 2019.

[75] A. K. Shetty, I. Saha, R. M. Sanghvi, S. A. Save, and Y. J. Patel, "A review: Object detection models," presented at the 2021 6th International Conference for Convergence in Technology (I2CT), Pune, India, 02-04 April 2021.

[76] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny," *ArXiv,* 2020.

Original Research Article: full paper

(2022), «EUREKA: Physics and Engineering»
Number 2

DEVELOPMENT OF OBJECT DETECTION AND CLASSIFICATION WITH YOLOV4 FOR SIMILAR AND STRUCTURAL DEFORMED FISH

Ari Kuswantori
Department of Instrumentation and Control Engineering[1]

Taweepol Suesut ✉
Department of Instrumentation and Control Engineering[1]
taweepol.su@kmitl.ac.th

Worapong Tangsrirat
Department of Instrumentation and Control Engineering[1]

Navaphattra Nunak
Department of Food Engineering[1]

[1]*School of Engineering*
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Chalongkrung Road No. 1, Ladkrabang District, Bangkok Province, Thailand, 10520

✉ **Corresponding author**

Abstract

Food scarcity is an issue of concern due to the continued growth of the human population and the threat of global warming and climate change. Increasing food production is expected to meet the challenges of food needs that will continue to increase in the future. Automation is one of the solutions to increase food productivity, including in the aquaculture industry, where fish recognition is essential to support it. This paper presents fish recognition using YOLO version 4 (YOLOv4) on the «Fish-Pak» dataset, which contains six species of identical and structurally damaged fish, both of which are characteristics of fish processed in the aquaculture industry. Data augmentation was generated to meet the validation criteria and improve the data balance between classes. For fish images on a conveyor, flip, rotation, and translation augmentation techniques are appropriate. YOLOv4 was applied to the whole fish body and then combined with several techniques to determine the impact on the accuracy of the results. These techniques include landmarking, subclassing, adding scale data, adding head data, and class elimination. Performance for each model was evaluated with a confusion matrix, and analysis of the impact of the combination of these techniques was also reviewed. From the experimental test results, the accuracy of YOLOv4 for the whole fish body is only 43.01 %. The result rose to 72.65 % with the landmarking technique, then rose to 76.64 % with the subclassing technique, and finally rose to 77.42 % by adding scale data. The accuracy did not improve to 76.47 % by adding head data, and the accuracy rose to 98.75 % with the class elimination technique. The final result was excellent and acceptable.

Keywords: computer vision, cultured-fish recognition, fish automation, fish classification, YOLO.

DOI: 10.21303/2461-4262.2022.002345

1. Introduction

Global warming and climate change are issues that have received much attention and concern from many scientists and researchers after the COVID-19 pandemic disaster [1]. The biggest problem that is feared to arise from this issue is that it will lead to food scarcity [2]. Food scarcity has also become a threat to the world community because the human population is growing [3]. For this reason, increasing food production is expected as one solution to answer the challenges of food needs that will continue to increase in the future. One of the strategies to increase productivity is automation. Besides increasing production, automation in the food industry will also maintain food quality and safety factors [2, 3].

In the aquaculture industry, especially in the fish industry and its processing, the fish recognition process has been needed to support the automation process during the sorting process,

inspection, or another process [4]. Fish recognition is an exciting and challenging topic because it presents various challenges [5, 6]. One of the challenges is that the types of fish are usually very similar to one another. In addition, the condition of the fish outside the water undergoes structural deformation, such as changed or damaged eyes, scales, and fins. These things become a unique challenge in the fish recognition process, especially for fish in the aquaculture industry [7].

Fish classification (FC) using computer vision and machine learning is an exciting research topic and has been continuously developed over the last two decades [7]. In recent years, new methods or approaches have been developed to achieve a high level of accuracy.

Image processing plays a vital role before the recognition algorithm is run. For example, it removes noise with a median filter, detects fish objects, and separates them from the background with a histogram, BLOB analysis, and threshold [7]. The other is image enhancement with contrast enrichment, auto segmentation of fish objects with various techniques [7–9], orientation correction using multi-stage exhaustive enumerative (MSEE) [7], Color Multi-Scale Retinex (MSR) to overcome water turbidity [10], and GMM, Pixel-Wise Posteriors [11], and Optical Flow [12] for fish detection in complex background scenarios.

Features on fish objects are essential for machine learning for the classification process [13]. The literature reported a maximum of 133 features for FC [14]. Features widely used for FC include geometric, statistical, color, and textual features [15]. Then, Artificial Neural Networks (ANN) with supervised learning algorithms were widely chosen for fish classification [9, 16–18].

The algorithms used also vary, such as CNN [4, 19-23], YOLO [10, 12, 24], few-shot learning for limited training images [25], Alex- Net, ResNet-18, ResNet-50, Inception-V3, and GoogLeNet [7]. In addition, modifications were also made to the algorithm to increase performance, such as modifications to Alex-Net [26], modification of CNN [27], and modification of Res-Net [9]. Another approach is combining standard algorithms with decision algorithms such as SVM (Support Vector Machine) [28, 29], Naive Bayesian classifier (NBC) [7, 30], KNN [31], decision tree [29], and backpropagation classifier [18]. Classification performance improves with integrating traditional classifiers such as random forest trees and SVM as layers compared to other standard deep learning networks [32]. In cloudy and blurred underwater images, hyperspectral imaging systems are reported to be more suitable than visual systems [33]. In addition, a transfer learning approach has also been attempted to use a pre-trained network for FC [34].

This paper presents the results of fish recognition using the viral YOLO algorithm with a relatively new version (version 4) which is still rarely used in fish. The condition of very similar fish between species and fish with structural deformation becomes a difficult challenge. In addition to testing the recognition of fish under these conditions using YOLOv4, a significant contribution of this work is the trial of combining YOLOv4 with various techniques to determine its impact on accuracy results. Applying landmarking, subclassing, adding fish scale data, adding fish head data, and class elimination strategies are among the techniques mentioned.

2. Material and methods

2. 1. Fish dataset

In this work, the «Fish-Pak» dataset has been used, in which its dataset consists of six fish species, namely:

1) *Catla* (Thala);
2) *Hypophthalmichthys molitrix* (Silver);
3) *Labeo rohita* (Rohu);
4) *Cirrhinus mrigala* (Mori);
5) *Cyprinus carpio* (Common carpio);
6) *Ctenopharyngodon Idella* (Grass carp) [35].

These fish are commonly bred in South Asia, such as India and Pakistan [36], and some types can also be found in other regions, such as Southeast Asia. Image data on each fish species, consisting of images of the whole body, head only, and scales only. There are 271 images of the body, 254 images of the scale, and 390 images of the head, so that the total fish images in this dataset are 915 images.

Some types of fish are very similar and very difficult to distinguish by ordinary people. In addition, the pictures of the fish were taken in out of water conditions with misalignment and structural deformation conditions such as the eyes and scales that were damaged lightly, moderately, and severely until the contents of the stomach came out. This causes this dataset to be considered quite ideal in this work. Examples of fish images in this dataset can be considered in **Fig. 1**.

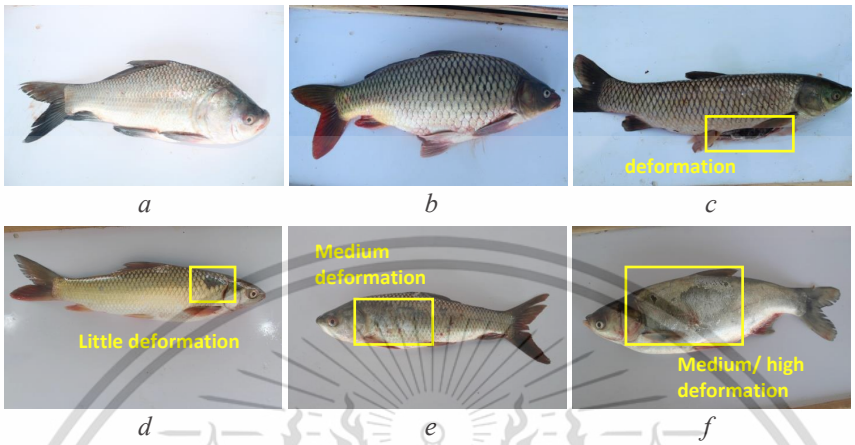


Fig. 1. Example images from Fish-Pak dataset:
a – Catla; *b* – C. Carpio; *c* – G. Carp; *d* – Mori; *e* – Rohu; *f* – Silver

2. 2. Image augmentation

The number of images for each fish species in the Fish-Pak dataset is minimal (less than 100) and not balanced between one species and another. Little image data for each class type makes validation low, and unbalanced data makes the algorithm get unequal opportunities in the training process for each class type. For this purpose, image augmentation is required [7]. The augmentation technique used in this work is flip, rotation, and translation, which is suitable for fish objects on the conveyor [32].

(1) is the number of original images in each class of fish (I_f^C), where $\left\{I_f^{1C}, I_f^{2C}, ..., I_f^{N_cC}\right\}$ indicates each fish image. The multiplier factor (m_f) is obtained through (2). It is obtained by comparing the number of images in each class (N_C) with the target images (N_T) where $N_C<N_T$. In this work, the fish body's target images are set to 100 [7]. By repeating this procedure for each class of fish, the set of multiplication factors can be obtained for all classes ($m_f^C, C \in [1,2,...,C]$). The multiplier factor indicates the number of augmented images for each class to be generated (I_{af}^{mc}). The augmentation techniques of flip (F_a), rotation (θ_a), and translation (T_a) were performed randomly for each image in each class (I_f^{nc}) according to the multiplier factor ($a \in \left[1,2,...,m_f^C\right]$) as in (3). Finally, all augmented images $\left\{I_{af}^{1C}, I_{af}^{2C}, ..., I_{af}^{N_cC}\right\}$ were collected with the original images so that a new dataset was formed (I_{Nf}^c).

As a consequence, the image is randomly divided into 80 % for training and 20 % for testing [26]. A description of this dataset can be summarized in **Table 1**.

With augmentation, the number of images becomes more prosperous, and now the data is more balanced, as indicated by the STD (Standard Deviation) value, which was initially 23.16 and then decreased to 19.87.

$$I_f^C = \left\{I_f^{1C}, I_f^{2C}, ..., I_f^{N_cC}\right\}, \tag{1}$$

$$m_f = \left\{m_f^c = \left|1 - \frac{N_C}{N_T}\right|, C \in [1,2,...C], N_C < N_T\right\}, \tag{2}$$

$$I_{af}^{mc} = \left\{H\left(I_f^{nc}, F_a, \theta_a, T_a\right) \middle| a \in \left[1,2,...,m_f^c\right]\right\}, \tag{3}$$

$$I_{Nf}^C = \left\{I_f^{1C}, I_f^{2C}, ..., I_f^{N_cC}, I_{af}^{1C}, I_{af}^{2C}, ..., I_{af}^{N_cC}\right\}. \tag{4}$$

Table 1
Dataset image detail and augmentation (body)

Fish	Number of Images (body)	Multiplication Factor	Number of Augmented Images	New dataset (Body)	Training (80 %)	Testing (20 %)
Catla	20	4	80	100	80	20
C. Carpio	50	1	50	100	80	20
G. Carp	11	9	99	110	88	22
Mori	70	1	70	140	112	28
Rohu	73	1	73	146	117	29
Silver	47	2	94	141	113	28
Total	271	–	466	737	590	147
Standard Deviation Average	23.16	–	–	19.87	–	–

In this work, the effect of adding scale and head data to the training process on the algorithm used is also tested. For this reason, scale and head image data in the dataset are also used. However, the scale and head data augmented and used for training are only limited to 3 classes, namely Catla, C. Carpio, and G. Carp, because only these three classes are challenging to detect and classify by the algorithm (low detection and classification results). Augmentation is also done so that the data is balanced for each class. The multiplication factor is determined in the same way that new datasets can be created. For scale image data, N_T is set to 27, and the head image is set to 48. All new datasets for scale and head in the three classes are used as training data. It can be observed in detail in **Table 2**.

Table 2
Scale and head image data and augmentation

Fish	Scale Image	Scale & Augmented	For Training (Scale)	Head Image	Head & Augmented	For Training (Head)	Total Training w/Scale & Head
Catla	11	33	33	25	75	75	188
C. Carpio	44	44	44	64	64	64	188
G. Carp	9	27	27	16	48	48	163
Mori	71	71	0	100	100	0	112
Rohu	62	62	0	114	114	0	117
Silver	57	57	0	71	71	0	113
Total	254	294	104	390	472	187	881
Standard Deviation Average (3 classes)	16.05	7.04	–	20.83	11.09	–	–

2. 3. YOLO and training process

In this work, YOLO version 4 (YOLOv4) is used. YOLO (You Only Look Once) is a very popular, widely used, and widely developed algorithm for object detection because of its ability to quickly and accurately detect objects. It is the reason why YOLO is used in this work. YOLO was built from 24 convolution layers and two fully connected layers at the beginning of its development, as shown in **Fig. 2**. Until now, YOLO has continued to be developed to improve accuracy and detection time, including up to YOLOv4 used in this work [37]. The YOLO training process in each test was carried out with a batch size of 32 with 32 subdivisions. Data enhancement was performed by rotation with a threshold between a minimum of –180° and a maximum of 180° (with 90 steps), and contrast with a threshold between a minimum of 0.4 and a maximum of 1.1 (with 0.2 steps). Data enhancement during the training process is not done by simulating noise and blur. Each training process is carried out until the average loss value becomes acceptable, i. e., up to 0.01 to a maximum of 0.03.

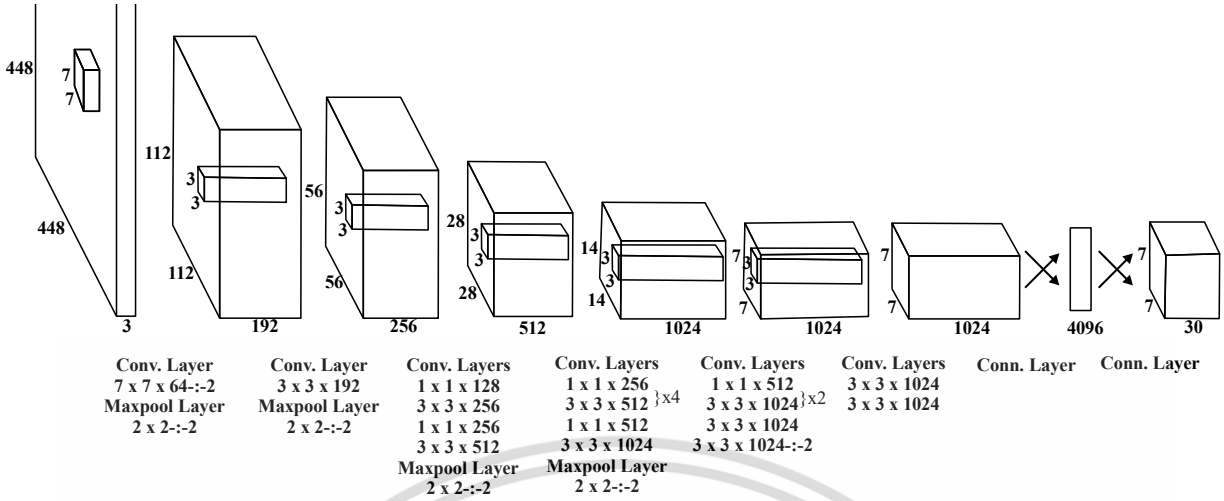


Fig. 2. YOLO Architecture [38]

2. 4. Experimental testing

YOLOv4, which is expected to work well to detect and classify similar and deformed fish, is applied to the selected dataset. Then several experiments were carried out to determine the impact on the results. The experiment combined the YOLOv4 with several techniques: landmarking, subclassing, adding scale data in the training process, adding scale and head data, and class elimination.

Recognition optimization using the landmarking technique (LM technique): this method makes the occupancy ratio of objects in the image effective during the training process. The occupancy ratio is the ratio between the object that wants to be recognized and the total area in the image or bounding box, including the background [7]. A higher excellent occupancy ratio (1 or close to 1) means expected more effectiveness for the algorithm to extract the features of the object during the training process, so that it is expected that the recognition result will be more accurate. The occupancy ratio of the image or bounding box (OR_{bb}) is obtained from the comparison between the object area in the image or bounding box (I_{bb}) and the total area of the row (M) and column (N) in the image or bounding box as in (5). In this work, let's compared the YOLOv4 training process with the whole image input and the landmarking method to determine the impact of the occupancy ratio on objects during training on the recognition output, as shown in **Fig. 3**.

a *b*

Fig. 3. Object segmentation method for YOLOv4 training: *a* – insert the whole image; *b* – use the landmarking technique

This landmarking technique is carried out with the CiRA-Core software:

$$OR_{bb} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{bb}(i,j)}{M \times N}. \quad (5)$$

Subclassing: the subclassing technique is used when different subspecies within a class, such as different colors, patterns, or shapes, can be distinguished. As in this work, class C. Carpio

has C. Carpio brown and C. Carpio red (**Fig. 4**), which are in the same class (C. Carpio). For that, the class C. Carpio red will be added in this subclassing technique. This method is applied to see the impact on the recognition ability in the class and as a whole.

a *b*

Fig. 4. The same class (C. Carpio) consists of different subspecies:
a – C. Carpio Brown; *b* – C. Carpio Red

Trials were carried out by incorporating images of heads and scales into the training process. It is intended to enrich object feature references and focus on algorithm training. Head and scale images are used because each type of fish in the dataset can be distinguished from the head and scales. The head and scale images have been augmented as previously described.

Class elimination technique: algorithm limitations in classification, are evaluated by class elimination techniques.

The number of classes eliminated (E_C) is determined by $N_C - 1$, where N_C is the number of classes whose accuracy level cannot be accepted, as shown in (6):

$$E_C = N_C - 1. \tag{6}$$

2. 5. Validation matrix

A confusion matrix is used to validate the experimental results. The confusion matrix is built from 4 building blocks, namely True Positive (*TP*), True Negative (*TN*), False Positive (*FP*), and False Negative (*FN*). True Positive (*TP*) is described as when the model detects the object correctly. True Negative (*TN*) is described when the model does not detect an object because the object does not exist. False Positive (*FP*) is described when the model detects an object but is wrong, including when it detects a double, even though it has correct predictions (double prediction with the wrong class). Furthermore, False Negative (*FN*) is described when the model does not detect an object even though it exists [8].

Accuracy is one of the evaluation metrics, which is denoted in (7). It is the ratio of the total number of correct predictions to the total of all predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\,\%, \tag{7}$$

where, *TP* – True Positives, *TN* – True Negatives, *FP* – False Positives, and *FN* – False Negatives.

Alternatively, the level of model accuracy can also be expressed by:

$$Accuracy = \frac{\sum_i^N Pi}{\sum_i^N |Qi|} \times 100\,\%, \tag{8}$$

where $\sum_i^N Pi$ is the number of correct predictions, and $\sum_i^N |Qi|$ is the total number of predictions.

Precision is the ratio between correctly classified fish (*TP*) and positive detection (number of *TP* and *FP*). It calculates the percentage of fish classified accurately as:

$$Precision = \frac{TP}{TP + FP} \times 100\,\%. \tag{9}$$

Recall or sensitivity is the ratio between the correctly classified fish (*TP*) and the fundamental truth fish (total number of *TP* and *FN*), which can be defined as:

$$Recall\ /\ Sensitivity = \frac{TP}{TP + FN} \times 100\,\%. \tag{10}$$

Specificity is determined by the ratio of *TN* to the sum of *FP* and *TN*, as described:

$$Specivity = \frac{TN}{TN + FP} \times 100\,\%. \tag{11}$$

*F*1*Score* (Measure *F*) is a metric calculated as the average of precision symphony and memory [39], as denoted by the following equation:

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \times 100\,\%. \tag{12}$$

3. Results and discussions

The experimental results were obtained from testing 20 % of the test data from the augmented dataset as described previously. First of all, YOLOv4 is trained with raw images as a whole, and the results are evaluated. From the test results, the results are not good. Only one class (Mori) achieved good accuracy (96.43 %), while the other classes only achieved 65 % accuracy (Catla) or below (Rohu; 48.28 % and Silver; 39.29 %). Class G. Carp achieves very low accuracy (9.09 %), and even class C. Carpio achieves 0 % accuracy. The overall accuracy was only 43.01 %.

Then YOLOv4 is trained by applying the landmarking technique to each image data set. The result is that the accuracy performance has increased by 72.65 %. However, in class C. Carpio and G. Carp still got poor accuracy scores at this step and were still not acceptable (15 and 40.91 %).

Then the subclassing technique is applied. At the final evaluation, C. Carpio red was separated and became a separate class, but it was still grouped as C. Carpio. As a result, the overall accuracy increased to 76.64 %. The accuracy results in class C. Carpio also increased significantly, from 15 to 60 %, and Catla from 80 to 100 %. However, it turns out that this also affects the level of accuracy of other classes that are lower. The Rohu class, originally 100, fell to 86.21 %, and the G. Carp, fell significantly from 40.91 to 13.64 %.

Then the data scale is added during the training process. As a result, the final average accuracy increased to 77.42 %. However, several classes produced low and unacceptable accuracy, namely C. Carpio (40 %), G. Carp (54.55 %), and Catla (70 %). Then the head data is also added. The result turned out to be unexpected because it did not improve accuracy, but slightly lowered it. The overall accuracy average dropped to 76.47 %. Some classes still produce a low and unacceptable level of accuracy, namely G. Carp (27.27 %), C. Carpio (50 %), and Catla (85 %).

The last is the data training trial by applying the class elimination technique. From a series of trials that have been carried out, three classes always get poor and unacceptable accuracy scores; G. Carp, C. Carpio, and Catla. By applying (6), which has been described previously, two classes will be eliminated. Those classes are two classes with the lowest accuracy results (G. Carp and C. Carpio). From the test results, the final accuracy value is excellent and could reach 98.75 %. Even so, the average value of accuracy for each class reaches 95 % or more. The accuracy results of this series of experiments are summarized in **Table 3** and **Fig. 5**. **Fig. 6** shows the results of the evaluation by the confusion matrix, and **Fig. 7** shows the other parameters' results (precision, recall/sensitivity, specificity, and F1 score).

A limitation should also be reported in this work. This study used CiRA-Core software as the main tool to run YOLO as a recognition algorithm. This software has many advantages, such as being easy to use and being integrated (supports ready-to-use technology). However, the main limitation related to this work is that the YOLO algorithm provided is patent/cannot be modified.

Because of that, it is not possible to do a study to modify the YOLO algorithm to improve recognition performance.

Table 3

Experimental results (accuracy)

Class	Detection Accuracy (%)					
	Body (Whole Image)	Body (with land-marking technique)	Body (after subclassing)	Body+ Scale	Body+ Scale+Head	Body (4 classes)
Catla	65.00	80.00	100.00	70.00	85.00	95.00
C. Carpio (+C. Carpio Red)	0.00	15.00	60.00	40.00	50.00	–
G. Carp	9.09	40.91	13.64	54.55	27.27	–
Mori	96.43	100.00	100.00	100.00	100.00	100.00
Rohu	48.28	100.00	86.21	100.00	96.55	100.00
Silver	39.29	100.00	100.00	100.00	100.00	100.00
Average	43.01	72.65	76.64	77.42	76.47	98.75

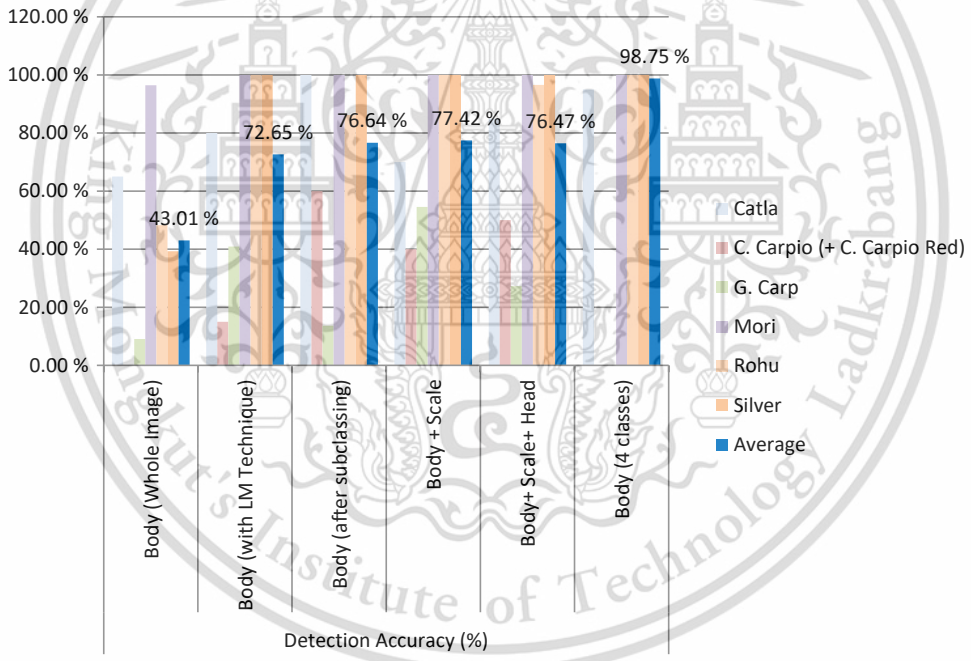


Fig. 5. Experimental results (accuracy)

Another point that should also be reported is the weakness. Suppose referring to the report of the results of this work, it is possible to agree that this study is very good to find out the impact and increase in the use of the techniques used in combination with YOLOv4 on similar and deformed fish objects on the level of accuracy or other performance parameters. However, the only major disadvantage in this study is that the best results could be achieved by eliminating (sacrificing) two classes of fish with low recognition performance. This matter makes this work incapable of fully answering the challenge.

For this reason, this study can be further developed in the future to improve accuracy and other performance parameters for an entire class. It may be done by modifying the YOLO algorithm used (by another tool) or applying image processing techniques that have not been performed in this study.

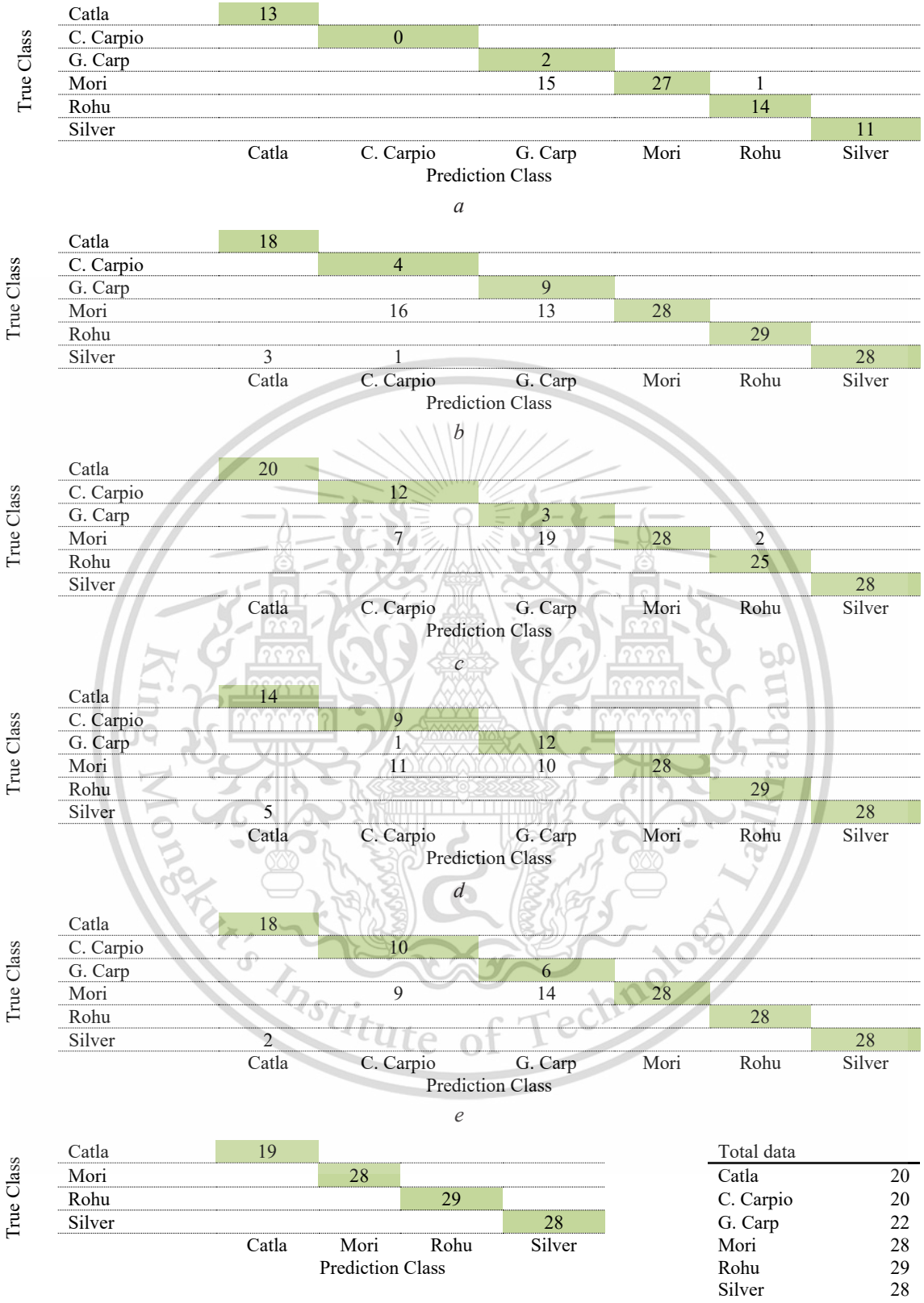


Fig. 6. Confusion matrix results for: *a* – training with the whole image; *b* – with landmarking technique; *c* – with subclassing technique; *d* – with the addition of scale image; *e* – with the addition of scale and head image; *f* – with only four classes (after class elimination); *g* – total data

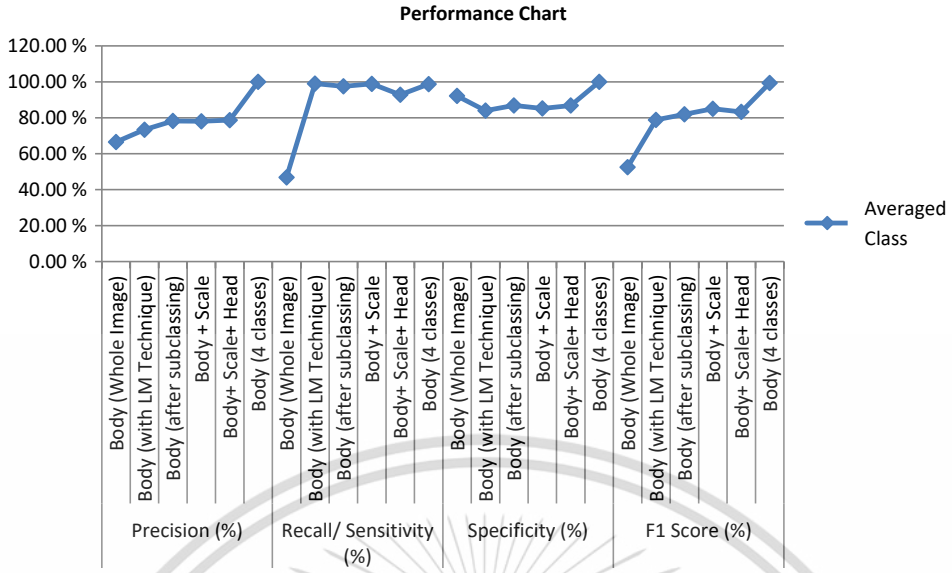


Fig. 7. Experimental results (precision, recall/sensitivity, specificity, and F1 score)

4. Conclusions

A series of experiments shows that YOLOv4 is promising for detecting and classifying fish species with similar and deformed conditions, both of which are characteristic of fish in the aquaculture industry. On the «Fish-Pak» dataset, which contains six species of fish, the accuracy of YOLOv4 is only 43.01 %, but the result rose to 72.65 % with the landmarking technique, and finally rose to 76.64 % with the subclassing technique, then rose to 77.42 % by adding scale data. The accuracy did not improve to 76.47 % by adding head data, and finally, the accuracy rose to 98.75 % with the class elimination technique. However, the accuracy rate can be further improved with complete classes (without class elimination) in future work. Image processing techniques may be improved for pre-processing or also by modifying the recognition algorithm.

Acknowledgments

This work was fully supported by King Mongkut's Institute of Technology Ladkrabang. The authors would also like to thank the College of Advanced Manufacturing Innovation (AMI-KMITL Thailand) for granting the CiRA-Core software license.

References

[1] Ranney, M. A., Velautham, L. (2021). Climate change cognition and education: Given no silver bullet for denial, diverse information-hunks increase global warming acceptance. Current Opinion in Behavioral Sciences, 42, 139–146. doi: https://doi.org/10.1016/j.cobeha.2021.08.001

[2] Bader, F., Rahimifard, S. (2018). Challenges for industrial robot applications in food manufacturing. ISCSIC'18: Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, 1–8. doi: https://doi.org/10.1145/3284557.3284723

[3] Goncharuk, A. (2015). Food business and food security challenges in research. Journal of Applied Management and Investments, 4 (4), 223–230. Available at: http://www.jami.org.ua/Papers/JAMI_4_4_223-230.pdf

[4] Dos Santos, A. A., Gonçalves, W. N. (2019). Improving pantanal fish species recognition through taxonomic ranks in convolutional neural networks. Ecological Informatics, 53, 100977. doi: https://doi.org/10.1016/j.ecoinf.2019.100977

[5] Alsmadi, M. K., Almarashdeh, I. (2020). A survey on fish classification techniques. Journal of King Saud University – Computer and Information Sciences. doi: https://doi.org/10.1016/j.jksuci.2020.07.005

[6] Zhao, S., Zhang, S., Liu, J., Wang, H., Zhu, J., Li, D., Zhao, R. (2021). Application of machine learning in intelligent fish aquaculture: A review. Aquaculture, 540, 736724. doi: https://doi.org/10.1016/j.aquaculture.2021.736724

[7] Abinaya, N. S. M., Susan, D., Rakesh Kumar, S. (2021). Naive bayesian fusion based deep learning networks for multi-segmented classification of fishes in aquaculture industries. Ecological Informatics, 61, 101248. doi: https://doi.org/10.1016/j.ecoinf.2021.101248

[8] Ahmed, M. S., Aurpa, T. T., Azad, M. A. K. (2021). Fish disease detection using image based machine learning technique in aquaculture. Journal of King Saud University – Computer and Information Sciences. doi: https://doi.org/10.1016/j.jksuci.2021.05.003

[9] Alshdaifat, N. F. F., Talib, A. Z., Osman, M. A. (2020). Improved deep learning framework for fish segmentation in underwater videos. Ecological Informatics, 59, 101121. doi: https://doi.org/10.1016/j.ecoinf.2020.101121

[10] Mohamed, H. E.-D., Fadl, A., Anas, O., Wageeh, Y., ElMasry, N., Nabil, A., Atia, A. (2020). Msr-yolo: Method to enhance fish detection and tracking in fish farms. Procedia Computer Science, 170, 539–546. doi: https://doi.org/10.1016/j.procs.2020.03.123

[11] Salman, A., Maqbool, S., Khan, A. H., Jalal, A., Shafait, F. (2019). Real-time fish detection in complex backgrounds using probabilistic background modelling. Ecological Informatics, 51, 44–51. doi: https://doi.org/10.1016/j.ecoinf.2019.02.011

[12] 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. Ecological Informatics, 57, 101088. doi: https://doi.org/10.1016/j.ecoinf.2020.101088

[13] Fouad, M. M. M., Zawbaa, H. M., El-Bendary, N., Hassanien, A. E. (2013). Automatic nile tilapia fish classification approach using machine learning techniques. 13th International Conference on Hybrid Intelligent Systems (HIS 2013). doi: https://doi.org/10.1109/HIS.2013.6920477

[14] Kutlu, Y., Iscimen, B., Turan, C. (2017). Multi-stage fish classification system using morphometry. Fresenius Environmental Bulletin, 26 (3), 1910–1916. Available at: https://www.researchgate.net/publication/314284234_MULTI-STAGE_FISH_CLASSIFICATION_SYSTEM_USING_MORPHOMETRY

[15] Hu, J., Li, D., Duan, Q., Han, Y., Chen, G., Si, X. (2012). Fish species classification by color, texture and multi-class support vector machine using computer vision. Computers and Electronics in Agriculture, 88, 133–140. doi: https://doi.org/10.1016/j.compag.2012.07.008

[16] Andayani, U., Wijaya, A., Rahmat, R. F., Siregar, B., Syahputra, M. F. (2019). Fish species classification using probabilistic neural network. Journal of Physics: Conference Series, 1235, 012094. doi: https://doi.org/10.1088/1742-6596/1235/1/012094

[17] Mohammadi Lalabadi, H., 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. Aquacultural Engineering, 90, 102076. doi: https://doi.org/10.1016/j.aquaeng.2020.102076

[18] Pornpanomchai, C., Lurstwut, B., Leerasakultham, P., Kitiyanan, W. (2013). Shape- and texture-based fish image recognition system. Kasetsart Journal - Natural Science, 47 (4), 624–634. Available at: https://www.researchgate.net/publication/289604551_Shape-_and_texture-based_fish_image_recognition_system

[19] Miyazono, T., Saitoh, T. (2017). Fish species recognition based on cnn using annotated image. IT Convergence and Security 2017, 156–163. doi: https://doi.org/10.1007/978-981-10-6451-7_19

[20] Rekha, B. S., Srinivasan, G. N., Reddy, S. K., Kakwani, D., Bhattad, N. (2020). Fish detection and classification using convolutional neural networks. Computational Vision and Bio-Inspired Computing, 1221–1231. doi: https://doi.org/10.1007/978-3-030-37218-7_128

[21] Taheri-Garavand, A., Nasiri, A., Banan, A., Zhang, Y.-D. (2020). Smart deep learning-based approach for non-destructive freshness diagnosis of common carp fish. Journal of Food Engineering, 278, 109930. doi: https://doi.org/10.1016/j.jfoodeng.2020.109930

[22] Villon, S., Mouillot, D., Chaumont, M., Darling, E. S., Subsol, G., Claverie, T., Villéger, S. (2018). A deep learning method for accurate and fast identification of coral reef fishes in underwater images. Ecological Informatics, 48, 238–244. doi: https://doi.org/10.1016/j.ecoinf.2018.09.007

[23] Labao, A. B., Naval, P. C. (2019). Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild. Ecological Informatics, 52, 103–121. doi: https://doi.org/10.1016/j.ecoinf.2019.05.004

[24] 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. Aquacultural Engineering, 91, 102117. doi: https://doi.org/10.1016/j.aquaeng.2020.102117

[25] Villon, S., Iovan, C., Mangeas, M., Claverie, T., Mouillot, D., Villéger, S., Vigliola, L. (2021). Automatic underwater fish species classification with limited data using few-shot learning. Ecological Informatics, 63, 101320. doi: https://doi.org/10.1016/j.ecoinf.2021.101320

[26] Ju, Z., Xue, Y. (2020). Fish species recognition using an improved alexnet model. Optik, 223, 165499. doi: https://doi.org/10.1016/j.ijleo.2020.165499

[27] Qin, H., Li, X., Liang, J., Peng, Y., Zhang, C. (2016). DeepFish: Accurate underwater live fish recognition with a deep architecture. Neurocomputing, 187, 49–58. doi: https://doi.org/10.1016/j.neucom.2015.10.122

[28] Islam, M. A., Howlader, M. R., Habiba, U., Faisal, R. H., Rahman, M. M. (2019). Indigenous fish classification of bangladesh using hybrid features with svm classifier. 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2). doi: https://doi.org/10.1109/IC4ME247184.2019.9036679

[29] Robotham, H., Castillo, J., Bosch, P., Perez-Kallens, J. (2011). A comparison of multi-class support vector machine and classification tree methods for hydroacoustic classification of fish-schools in Chile. Fisheries Research, 111 (3), 170–176. doi: https://doi.org/10.1016/j.fishres.2011.07.010

[30] Kutlu, Y., Reyhaniye, A. N., Turan, C. (2014). Image analysis methods on fish recognition. 2014 22nd Signal Processing and Communications Applications Conference (SIU). doi: https://doi.org/10.1109/SIU.2014.6830503

[31] Badawi, U., Alsmadi, M. (2014). A general fish classification methodology using meta-heuristic algorithm with back propagation classifier. Journal of Theoretical and Applied Information Technology, 66 (3), 803–812. Available at: http://www.jatit.org/volumes/Vol66No3/18Vol66No3.pdf

[32] Liu, Z., Jia, X., Xu, X. (2019). Study of shrimp recognition methods using smart networks. Computers and Electronics in Agriculture, 165, 104926. doi: https://doi.org/10.1016/j.compag.2019.104926

[33] Pettersen, R., Braa, H., Gaweł, B., Letnes, P., Sæther, K., Aas, L. (2019). Detection and classification of Lepeophterius salmonis (Krøyer, 1837) using underwater hyperspectral imaging. Aquacultural Engineering, 87, 102025. doi: https://doi.org/10.1016/j.aquaeng.2019.102025

[34] Liawatimena, S., Heryadi, Y., Lukas, Trisetyarso, A., Wibowo, A., Abbas, B. S., Barlian, E. (2018). A Fish Classification on Images using Transfer Learning and Matlab. 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), 108–112. doi: https://doi.org/10.1109/INAPR.2018.8627007

[35] Shah, S. Z. H., Rauf, H. T., IkramUllah, M., Khalid, M. S., Farooq, M., Fatima, M., Bukhari, S. A. C. (2019). Fish-pak: Fish species dataset from pakistan for visual features based classification. Data in Brief, 27, 104565. doi: https://doi.org/10.1016/j.dib.2019.104565

[36] Rauf, H. T., Lali, M. I. U., Zahoor, S., Shah, S. Z. H., Rehman, A. U., Bukhari, S. A. C. (2019). Visual features based automated identification of fish species using deep convolutional neural networks. Computers and Electronics in Agriculture, 167, 105075. doi: https://doi.org/10.1016/j.compag.2019.105075

[37] Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv. doi: https://doi.org/10.48550/arXiv.2004.10934

[38] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv. doi: https://doi.org/10.48550/arXiv.1506.02640

[39] Minh, D. H. T., Ienco, D., Gaetano, R., Lalande, N., Ndikumana, E., Osman, F., Maurel, P. (2018). Deep Recurrent Neural Networks for Winter Vegetation Quality Mapping via Multitemporal SAR Sentinel-1. IEEE Geoscience and Remote Sensing Letters, 15 (3), 464–468. doi: https://doi.org/10.1109/LGRS.2018.2794581

Received date 04.11.2021
Accepted date 20.03.2022
Published date 31.03.2022

How to cite: *Kuswantori, A., Suesut, T., Tangsrirat, W., Nunak, N. (2022). Development of object detection and classification with YOLOv4 for similar and structural deformed fish. EUREKA: Physics and Engineering, 2, 154–165. doi: https://doi.org/10.21303/2461-4262.2022.002345*

1st CALL FOR PAPERS

ITC-CSCC 2022

The 37th International Technical Conference on Circuits /Systems, Computers and Communications (ITC-CSCC 2022)

July 5-8, 2022, Duangjitt Resort & Spa, Patong, Phuket, Thailand

With the great success of the International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) as the world leading conference devoted to the advancement of high technologies in Circuits, Systems, Computers, and Communications, we would like to invite all the scholars and experts around the world to attend the 37th ITC-CSCC 2022 to be hosted in Phuket, Thailand.

Topics

The conference is open to researchers from all regions of the world. Participation from Asia Pacific region is particularly encouraged. Proposals for special sessions are welcome. Papers with original work in all aspects of Circuits, Systems, Computers, and Communications are invited. Topics include, but not limited to, the followings

Circuit and systems

- Analog Circuits
- Computer Aided Design
- Intelligent Transportation Systems & Technology
- Linear / Nonlinear Systems
- Medical Electronics & Circuits
- Modern Control
- Neural Networks
- Power Electronics & Circuits
- RF Circuits
- Semiconductor Devices & Technology
- Sensors & Related Circuits
- Verification & Testing
- VLSI Design

Communications

- Antenna & Wave Propagation
- Audio / Speech Signal Processing
- Circuits & Components for Communications
- IP Networks & QoS
- MIMO & Space-Time Codes
- Multimedia Communications
- Mobile & Wireless Communications
- Network Management & Design
- Optical Communications & Components
- Radar / Remote Sensing
- Communication Signal Processing
- Ubiquitous Networks
- UWB
- Visual Communications
- Wireless Sensor Networks
- Underwater Communications

Computers

- Artificial Intelligence
- Biocomputing
- Computer Systems & Applications
- Computer Vision
- Face Detection & Recognition
- Image Coding & Analysis
- Image Processing
- Internet Technology & Applications
- Motion Analysis
- Multimedia Service & Technology
- Object Extraction & Technology
- Security
- Watermarking
- Blockchain
- Data Analytics
- Internet of Things
- Virtual Reality

Submission of Papers

Prospective authors are invited to submit an original paper with 2 – 4 pages in length of PDF format written in English. Paper submission procedures are available at https://itc-cscc2022.org

Proceedings and Publications

All registered participants are provided with online conference proceedings. Upon requested, accepted papers will be published in IEEE Xplore. Moreover, authors of the accepted papers are encouraged to submit full-length manuscripts to IEIE JSTS (Korea), IEIE SPC (Korea), IEICE Transactions (Japan), or ECTI Transactions (Thailand). All the submissions need to follow the standard procedure of their publication and be published on regular issues.

Contact: secretary@itc-cscc2022.org , http://www.itc-cscc2022.org

Important Dates

Deadline of Manuscript Submission	: April 1, 2022
Notification of Acceptance	: May 14, 2022
Submission of Camera-Ready Paper	: June 6, 2022

IEIE The Institute of Electronics and Information Engineers

ECTI Association

Fish Recognition Optimization in Various Backgrounds Using Landmarking Technique and YOLOv4

Ari Kuswantori
Instrumentation and Control Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Bangkok, Thailand
63601247@kmitl.ac.th

Taweepol Suesut
Instrumentation and Control Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Bangkok, Thailand
taweepol.su@kmitl.ac.th

Worapong Tangsrirat
Instrumentation and Control Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Bangkok, Thailand
worapong.ta@kmitl.ac.th

Sutham Satthamsakul
Instrumentation and Control Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Bangkok, Thailand
sutham.sa@kmitl.ac.th

2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) | 978-1-6654-8559-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITC-CSCC55581.2022.9895101

***Abstract*— The identification and categorization of fish is a popular and fascinating research topic. Many researchers have developed expertise in fish detection, both underwater and outside the water, which is particularly beneficial for population management and aquaculture. This paper proposes a fish recognition approach using the landmarking methodology with YOLO version 4 to identify and categorize fish with different backdrop circumstances. The approach can be used both underwater and on land. The proposed approach was evaluated using four distinct types of fish from the BYU dataset. The final test result determined that the accuracy reached 96.60%, with an average classification score of 99.67% at the 60% threshold. The result is 4.94% better than the most frequent traditional labelling approach.**

Keywords*— *fish classification, fish recognition, YOLO, landmarking technique, computer vision

I. INTRODUCTION

Fish recognition presents several complicated and varied problems, both in and out of the water [1], and one of them is the various backdrop challenges [2-8]. Many approaches have been attempted to overcome this problem and optimize the recognition results. Work [2] coped with this challenge by including backgrounds such as reef bottoms and water with nine types of fish classes in the training process and then using CNN (Convolutional Neural Network) for classification. In [3], they developed a CNN-based method for underwater fish detection for cascaded deep network systems with linked ensemble components. [4] suggested utilizing ResNet-50 and GMM (Gaussian Mixture Models)-Optical Flow combined with YOLOv3 to identify and classify fish in underwater environments using deep learning with temporal information. Furthermore, the method proposed in [5] employed GMM and Pixel-wise Posteriors to recognize fish in various environments. The approach of [6] proposed instant segmentation on underwater fish images and subsequently developed an algorithm based on Res-Net technology. Work [7] removed the background first before entering the fish image into the classification algorithm. The foreground was identified using BLOB (Binary Large Object) then background and noise were removed by threshold. And in [8], foreground extraction and a CNN-based algorithm with an SVM (Support Vector Machines) classifier were used to develop accurate underwater live fish recognition utilizing a deep architecture in various background conditions. However, several of these techniques achieve high final accuracy, i.e., > 90%, while others do not. Furthermore, certain methods utilized are complicated, and some are not yet integrated between process steps, implying that they are not yet ready-to-use solutions.

The primary idea of this study is the identification and categorization of fish in a variety of background conditions using a different, simple and integrated methodology that combines landmarking techniques with YOLO version 4 (YOLOv4). Both landmarking technique and YOLOv4 are relatively new and rarely used for fish classification. Furthermore, the accuracy results also will be compared with the conventional labelling method, which is the most commonly used.

II. MATERIAL AND METHODOLOGY

A. Dataset and Image Augmentation

This work aims to detect and classify fish in various background conditions, such as objects, colours, and others. As a result, the BYU dataset is thought to be excellent for use in this study. The Brigham Young University (BYU) robotic vision group created this dataset [9]. This dataset consists of 16 different fish species with varying shooting conditions.

Furthermore, each species' image from the image data of 16 fish species is very unbalanced. For these reasons, only four species were used in this study, with a large amount of image data, raw images (not yet processed), and varying backdrop conditions (natural and underwater backgrounds). Table I lists the four fish species as B.C. Trout, Kokanee, UT

978-1-6654-8559-3/22/$31.00 @2022 IEEE

Sucker, and Steal Head. Fig.1 shows several examples of image data from the BYU dataset.

TABLE I. BYU DATASET AND IMAGE AUGMENTATION

Fish class	No. of images	Multiplication factor	No. of augmented images	New image dataset	For training (80%)	For testing (20%)
B.C. Trout	191	0	0	191	153	38
Kokanee	60	1	60	120	96	24
UT Sucker	87	1	87	174	139	35
Steal Head	25	3	75	100	80	20
Total	363	-	222	585	468	117
Standard Deviation Value (SDV)	61.91	-	-	37.42	-	-
Average	91	-	-	146	-	-

Fig. 1. Sample Images of BYU dataset

It can be observed from Table I that the amount of image data for the three classes, namely Kokanee, UT Sucker, and Steal Head, is small and unbalanced in comparison to the overall class image data. Small data (less than 100) reduce validation, and imbalanced data prevent the algorithm from having the same opportunity during the training process. For this reason, an augmentation process is needed [7]. The augmentation techniques used in this work are flip, rotation, and translation. This approach is thought to be appropriate for fish augmentation [10]. The augmentation process can be described by equations (1)-(4).

$$I_f^C = \{I_f^{1C}, I_f^{2C}, \dots, I_f^{N_C C}\}, \quad (1)$$

$$m_f = \left\{m_f^C = \left|1 - \frac{N_T}{N_C}\right|, c \in [1,2,\dots C], N_C < N_T\right\}, \quad (2)$$

$$I_{af}^{nC} = \left\{H\left(I_f^{nC}, F_a, \theta_a, T_a,\right) \middle| a \in [1,2,\dots, m_f^C]\right\}, \quad (3)$$

and

$$I_{Nf}^C = \{I_f^{1C}, I_f^{2C}, \dots, I_f^{N_C C}, I_{af}^{1C}, I_{af}^{2C}, \dots, I_{af}^{N_C C}\}. \quad (4)$$

Initially, each class (C) of fish (f) images (I) (I_f^C) is selected with the number of images (N_C) as in equation (1). Multiplication factor (m_f^C) is obtained by comparing the number of images in each class (N_C) with the target (N_T). The target (N_T) is set at 100 for each class in this work. The value of the multiplication factor must be positive and rounded up as integers, where $N_C < N_T$. The set of multiplication factors (m_f) is obtained by repeating the procedure for each class of fish as in equation (2). The multiplication factor indicates the number of augmented images that need to be created from each image (I_f^{nC}) in each class of fish. Flip (F_a), rotation (θ_a), and translation vector (T_a) are selected randomly and generate an augmented image (I_{af}^{nC}) as in equation (3). The variances of each of the flip (F_a), rotation (θ_a), and translation vectors (T_a) were predetermined. Finally, all the augmented image sets are merged with the original image set, and a new data set (I_{Nf}^C) is formed as in equation (4).

After the augmentation procedure, the new dataset has a significant number of images, ranging from 363 to 585. In addition, the average number of images in each class increased from 91 to 146. Moreover, the dataset is now more balanced, evidenced by a decrease in the standard deviation value from 61.91 to 37.42. Following that, the images were randomly separated for the training (80%) and testing (20%) processes [11].

B. Occupancy Ratio and Landmarking Technique

The occupancy ratio of an image or image in a bounding box (OR_{bb}) is a comparison between the object area or object bounding box and the overall image area or image bounding box, which includes the background. The high occupancy ratio minimizes the deep learning algorithm capturing unwanted backgrounds during the training process. It will improve the effectiveness of deep learning in capturing features that can only be found in objects. The occupancy ratio is denoted by the following expression [7].

$$OR_{bb} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{bb}(i,j)}{MxN} \quad (5)$$

where, $\sum_{i=1}^{M}\sum_{j=1}^{N} I_{bb}(i,j)$ is the object area or object bounding box, and M × N is the overall image area or image bounding box.

The landmarking technique is a relatively new labelling technique currently utilized sparingly, especially on fish objects. This technique is employed immediately before the image is utilized for the training process. With this landmarking technique, any part of the object can be marked and become an area that is only processed during training. With this method, the occupancy ratio will become very high or may even reach one, leading to more effective and on-target feature maps. As a result, the training process will be more effective, and the recognition results will be more accurate and robust.

This landmarking technique is carried out with the CiRA-Core software developed by the College of Advanced Manufacturing Innovation (AMI-KMITL) Thailand and available on https:// git. cira-lab.com/cira/cira-core. This software was advanced introduced and used at work [12] and [13]. How to run the landmarking on this software is we put the coordinate points on the object in the image, and then the program will convert it into a polygon and form an area on the object. Only objects or parts of objects marked as functional areas will be extracted for the training process, and anything outside will be ignored. Consequently, in addition to the bounding box (bbox), centre point, colour, label, and index label information, the output of this process includes data landmark points and landmark len, which will

be used in the training process to improve its effectiveness. The concept of this landmarking technique and the illustrations on the software can be observed in Fig. 2.

Furthermore, this paper also presents fish recognition using the most commonly used conventional labelling technique. The comparison between the landmarking technique and the conventional technique can be observed in Fig. 3.

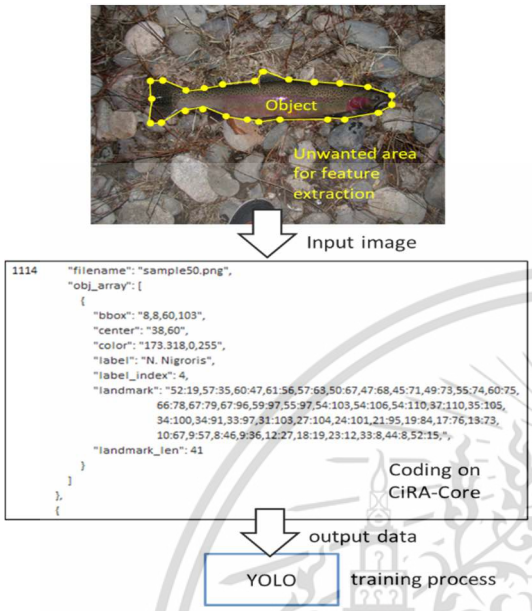
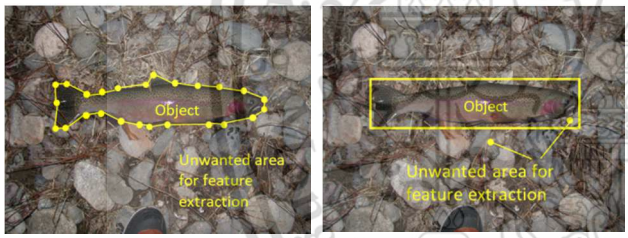


Fig. 2. Landmarking technique



(a) (a)

Fig. 3. Labelling technique (a) using the landmarking technique, (b) using the most common and conventional technique

C. YOLO v4

In this work, the detection and classification algorithm is implemented in YOLO version 4. YOLO *(You Only Look Once)* is viral object detection and classification algorithm. YOLO is extensively used and popular because of its reasonable detection rate, accuracy, and fast detection time. At the beginning of its development, YOLO was built from 24 convolution layers and two fully connected layers as shown in Fig. 4 [14]. YOLOv4 is the development of YOLOv3, which was released in 2020. The backbone of YOLOv4 is CSPDarknet53, with additional modules for spatial pyramid pooling, PANet path-aggregation neck, and YOLOv3 head. [15].

YOLOv4 was learned using training data from the BYU dataset (as shown in Table I) and achieved good results with an average loss= 0.01. Training is done in batches of 32 data points with 32 subdivisions. Data enhancement is accomplished by rotation with a threshold ranging from -180° to 180° in 90 steps and contrast with a threshold ranging from 0.4 to 1.1 in 0.2 steps.

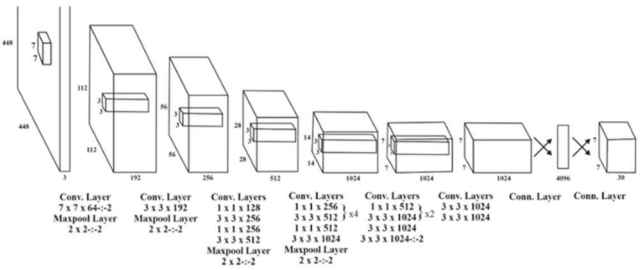


Fig. 4. YOLO architecture [14]

D. Validation Matrix

The confusion matrix is used to describe the model's performance, and the model's accuracy may be assessed using equation (6) [16].

$$Accuracy = \frac{\sum_i^N Pi}{\sum_i^N |Qi|} \times 100\% \qquad (6)$$

where $\sum_i^N Pi$ is the number of correct predictions, and $\sum_i^N |Qi|$ is the total number of predictions.

III. Experimental Results

The proposed approach was tested on the BYU testing data. As explained before, testing data consists of 20% images from every class after augmentation. The trained algorithm was tested on one by one test images and got four result conditions; *correct classification, wrong classification, not detect,* and *double classification*. *Correct classification* is when the model can classify fish correctly (as the true class). The *wrong classification* is when the model predicts incorrect fish, *not detect* when the model fails to detect the fish, and *double classification* is when the model classifies one as two different fish.

Every output or model's prediction has a classification or confident score from 0 to 100%. Moreover, the best accuracy is obtained from the 60% threshold from the experimental results. It will make the model will ignore the decision with less than 60% of confidence or classification score. The experimental result of the trained algorithm with the landmarking technique is resumed in Table II.

From Table II, we can observe that the BC Trout class has 38 images as testing data, and the model could correctly classify 35 fish images but failed to detect fish in 3 images. So we can define the accuracy is 92.11% by the equation (6) with the total prediction of 38 (35 Correct classification+ 0 Wrong classification+ 3 of Not detect+ 0 Double classification). By all correct classifications, the average classification (confident) score is 99.67%.

The model could perfectly classify 24 and 20 testing images for the Kokanee and Steelhead classes, so the accuracy is 100%. However, the UT Sucker class has wrong and double classifications. In this experiment, the final accuracy result is obtained 96.60%.

This work also sets the trained YOLOv4 algorithm model toward the test using the conventional labelling approach, which is routinely utilized on the same test data. We also did in 60% threshold. The final average accuracy result of the conventional labelling technique is 91.66%. It means that the accuracy result is 4.94% lower than the proposed method. Table III summarizes the findings of YOLOv4 detection using this conventional labelling technique.

TABLE II. EXPERIMENTAL RESULTS OF YOLOV4 WITH LANDMARKING TECHNIQUE AT 60% THRESHOLD

Threshold: 60%								
Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	0	3	0	38	92.11	99.67
Kokanee	24	24	0	0	0	24	100.00	99.93
UT Sucker	35	33	1	0	1	35	94.29	99.24
Steelhead	20	20	0	0	0	20	100.00	99.83
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	96.60	99.67

TABLE III. YOLOV4 DETECTION RESULTS WITH CONVENTIONAL LABELING TECHNIQUES AT 60% THRESHOLD

Threshold: 60%								
Class	Testing data	Correct classification	Wrong classification	Not detect	Double classification	Total prediction	Accuracy (%)	Avg. Classification score (%)
BC Trout	38	35	1	2	0	38	92.11	99.12
Kokanee	24	22	0	0	2	24	91.67	99.99
UT Sucker	35	29	0	0	6	35	82.86	99.96
Steelhead	20	20	0	0	0	20	100.00	99.87
Total	117	-	-	-	-	-	-	-
Average	-	-	-	-	-	-	91.66	99.74

Another issue that we want to report is that although the final accuracy results obtained are quite good (>95%), these results can still be improved by modifying the YOLO algorithm used or using image processing techniques that have not been carried out in this work. Modifying the YOLO algorithm cannot be carried out in this work because the tools used do not allow it, which is the biggest limitation of this research. Using image processing techniques will emphasize feature maps on the object before the training process so the captured features can be more effective and improve the final accuracy result.

IV. CONCLUSION

The purpose of this study is to provide an approach for identifying fish in a variety of background conditions. The methodology consists of combining the landmarking technique with YOLO version 4. The proposed approach was tested with four different fish species using the BYU dataset. The detection accuracy findings from the experimental tests are 96.60% at the 60% threshold. This result is 4.94% higher than the conventional labelling procedure. This trial's outcomes are considered quite good, and the method used is quite straightforward. The proposed approach does not necessitate any complicated work steps, and it is ready to be used as a ready solution.

ACKNOWLEDGEMENTS

This work was fully supported by King Mongkut's Institute of Technology Ladkrabang (KMITL). And we want to thank to AMI-KMITL for granting the CiRA-Core software license.

REFERENCES

[1] M. K. Alsmadi and I. Almarashdeh, "A survey on fish classification techniques," *Journal of King Saud University - Computer and Information Sciences,* 2020.

[2] S. Villon, D. Mouillot, M. Chaumont, E. S. Darling, G. Subsol, T. Claverie*, et al.*, "A Deep learning method for accurate and fast identification of coral reef fishes in underwater images," *Ecological Informatics,* vol. 48, pp. 238-244, 2018/11/01/ 2018.

[3] A. B. Labao and P. C. Naval, "Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild," *Ecological Informatics,* vol. 52, pp. 103-121, 2019/07/01/ 2019.

[4] A. Jalal, A. Salman, A. Mian, M. Shortis, and F. Shafait, "Fish detection and species classification in underwater environments using deep learning with temporal information," *Ecological Informatics,* vol. 57, p. 101088, 2020/05/01/ 2020.

[5] A. Salman, S. Maqbool, A. H. Khan, A. Jalal, and F. Shafait, "Real-time fish detection in complex backgrounds using probabilistic background modelling," *Ecological Informatics,* vol. 51, pp. 44-51, 2019.

[6] N. F. F. Alshdaifat, A. Z. Talib, and M. A. Osman, "Improved deep learning framework for fish segmentation in underwater videos," *Ecological Informatics,* vol. 59, p. 101121, 2020/09/01/ 2020.

[7] A. N.S, S. D, and R. K. S, "Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries," *Ecological Informatics,* vol. 61, p. 101248, 2021/03/01/ 2021.

[8] H. Qin, X. Li, J. Liang, Y. Peng, and C. Zhang, "DeepFish: Accurate underwater live fish recognition with a deep architecture," *Neurocomputing,* vol. 187, pp. 49-58, 2016.

[9] K. D. Lillywhite, Lee, D.J. (2013, 19/09/21). *Robotic vision lab, Brigham Young University, Fish dataset.* Available: http://roboticvision.groups.et.byu.net/Machine_Vision/BYUFish/BYU_Fish.html

[10] Z. Liu, X. Jia, and X. Xu, "Study of shrimp recognition methods using smart networks," *Computers and Electronics in Agriculture,* vol. 165, p. 104926, 2019/10/01/ 2019.

[11] Z. Ju and Y. Xue, "Fish species recognition using an improved AlexNet model," *Optik,* vol. 223, p. 165499, 2020/12/01/ 2020.

[12] V. Kittichai, T. Pengsakul, K. Chumchuen, Y. Samung, P. Sriwichai, N. Phatthamolrat*, et al.*, "Deep learning approaches for challenging species and gender identification of mosquito vectors," *Scientific reports,* vol. 11, pp. 1-14, 2021.

[13] V. Kittichai, M. Kaewthamasorn, S. Thanee, R. Jomtarak, K. Klanboot, K. M. Naing*, et al.*, "Classification for avian malaria parasite Plasmodium gallinaceum blood stages by using deep convolutional neural networks," *Scientific reports,* vol. 11, pp. 1-10, 2021.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016.

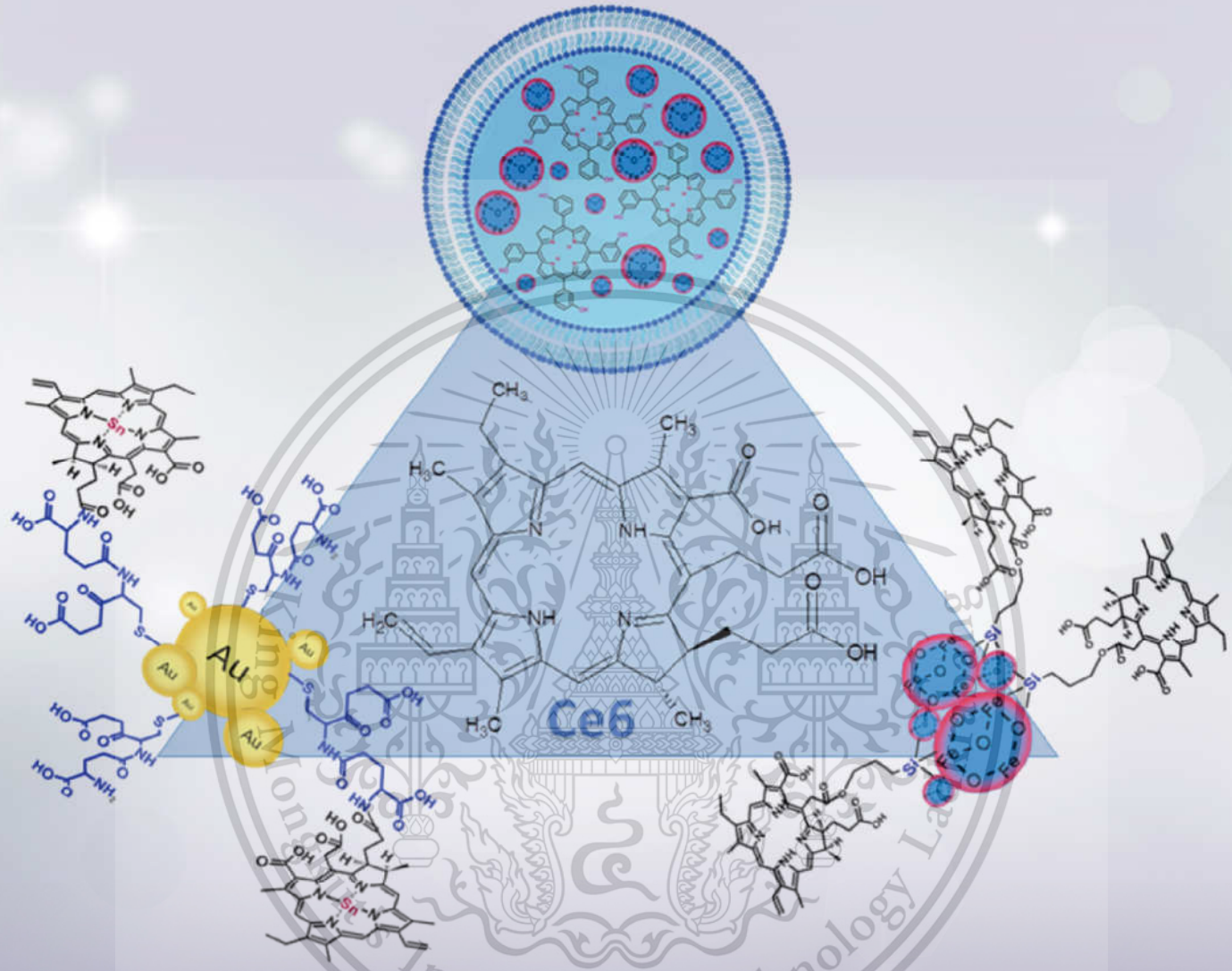
[15] A. B. a. C.-Y. W. a. H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[16] N. D. Marom, L. Rokach, and A. Shmilovici, "Using the confusion matrix for improving ensemble classifiers," in *2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, 2010, pp. 000555-000559.

applied sciences

IMPACT FACTOR 2.838

CITESCORE 3.7



Connections of Chlorin e6 and Metallic Nanoparticles for Biomedical Applications

Volume 13 · Issue 6 | March (II) 2023

MDPI

mdpi.com/journal/applsci
ISSN 2076-3417

applied sciences

an Open Access Journal by MDPI

CITESCORE 3.7

IMPACT FACTOR 2.838

CERTIFICATE OF PUBLICATION

Certificate of publication for the article titled:

Fish Detection and Classification for Automatic Sorting System with an Optimized YOLO Algorithm

Authored by:

Ari Kuswantori; Taweepol Suesut; Worapong Tangsrirat; Gerhard Schleining; Navaphattra Nunak

Published in:

Appl. Sci. 2023, Volume 13, Issue 6, 3812

MDPI Academic Open Access Publishing since 1996

Basel, March 2023

applied sciences

MDPI

Article

Fish Detection and Classification for Automatic Sorting System with an Optimized YOLO Algorithm

Ari Kuswantori [1], Taweepol Suesut [1,*], Worapong Tangsrirat [1], Gerhard Schleining [2] and Navaphattra Nunak [3]

[1] Department of Instrumentation and Control Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok 10520, Thailand; 63601247@kmitl.ac.th (A.K.); worapong.ta@kmitl.ac.th (W.T.)
[2] Department of Food Science and Technology, University of Natural Resources and Life Sciences Vienna (BOKU), 1190 Vienna, Austria; gerhard.schleining@boku.ac.at
[3] Department of Food Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok 10520, Thailand; navaphattra.nu@kmitl.ac.th
* Correspondence: taweepol.su@kmitl.ac.th

Featured Application: In the future, the application of this study is very feasible and very close to being implemented for the auto-sorting system for various fish or other objects, in the fish industry or other industries, with deep learning and machine vision technology.

Abstract: Automatic fish recognition using deep learning and computer or machine vision is a key part of making the fish industry more productive through automation. An automatic sorting system will help to tackle the challenges of increasing food demand and the threat of food scarcity in the future due to the continuing growth of the world population and the impact of global warming and climate change. As far as the authors know, there has been no published work so far to detect and classify moving fish for the fish culture industry, especially for automatic sorting purposes based on the fish species using deep learning and machine vision. This paper proposes an approach based on the recognition algorithm YOLOv4, optimized with a unique labeling technique. The proposed method was tested with videos of real fish running on a conveyor, which were put randomly in position and order at a speed of 505.08 m/h and could obtain an accuracy of 98.15%. This study with a simple but effective method is expected to be a guide for automatically detecting, classifying, and sorting fish.

Keywords: automatic fish sorting; fish classification; fish recognition; YOLO; Computer and machine vision

Citation: Kuswantori, A.; Suesut, T.; Tangsrirat, W.; Schleining, G.; Nunak, N. Fish Detection and Classification for Automatic Sorting System with an Optimized YOLO Algorithm. *Appl. Sci.* **2023**, *13*, 3812. https://doi.org/10.3390/app13063812

Academic Editor: Antonio Fernández-Caballero

Received: 21 February 2023
Revised: 11 March 2023
Accepted: 14 March 2023
Published: 16 March 2023

1. Introduction

Automatic fish detection, recognition, and classification are popular and intriguing areas of research. Numerous researchers are engaged in its development for underwater and out-of-water environments [1–6]. Recognizing fish in underwater conditions, especially in deep ocean environments, is advantageous for fish population control and sustainability [3,7–11] and supports the development of the IoUT (Internet of Underwater Things) [12,13]. Recognizing fish in settings other than water, especially for farming fish, is useful in aquaculture, for example, for automatic sorting processes, fish quality monitoring, and other activities [4–6,14–19].

Automatic fish recognition using machine and computer vision is a key part of automating the fish industry, which is part of the food industry, to boost productivity in aquaculture. It deals with the problems of high food demand and the possibility that there won't be enough food in the future because of the growing world population and

the effects of global warming and climate change [20,21]. One of them is for the automatic sorting system, which has already been mentioned and is being worked on by many researchers [22–25]. Automatic detection and classification of moving fish is the key to automatic sorting systems in the fish culture industry, and it has some unique challenges. The main challenge is the speed, and others are the random position of the fish, the background, the deformed condition of the fish, and the fact that fish usually have a similar visual appearance [14].

For moving fish recognition, many studies have been carried out. The study in [7] employed CNN (Convolutional Neural Network) to classify fish by training them with the number of species and their environments, such as reef bottoms and water. They applied their proposed method to 116 underwater fish videos recorded using a GoPro underwater camera. The best results were achieved in classifying 9 of the 20 types of fish that appear most often in the videos. In [8], a multi-cascade object detection network with an ensemble of seven CNN components and two RPNs (Region Proposal Network) linked by sequentially jointly trained LSTMs (Long Short-Term Memory units) was performed. For training and testing, they used a set of 18 underwater fish videos that were also recorded with a GoPro underwater camera. Even though their proposed method can reliably find and count fish objects in a variety of benthic backgrounds and lighting conditions, it is only used to find fish and not to classify them. Using classic CNNs like these also has advantages when applied to other sectors, such as agriculture [26], or in other broad cases, such as detecting fine scratches [27]. The moving fish recognition in [9] utilized Optical flow, GMM (Gaussian Mixture Models), and ResNet-50, then combined the output with YOLOv3. The combination of those methods enabled the robust detection and classification of fish, which was applied to the LifeCLEF 2015 benchmark dataset from the Fish4Knowledge repository [28] and a dataset collected by the University of Western Australia (UWA) which was explained in detail by [29]. The GMM and Pixel-wise posteriors were proposed in [11], and then further developed by combining them with CNN [30]. They also used a fish dataset extracted from the Fish4Knowledge repository in their work. Similar to the work [8], the approaches proposed in their papers were only for detecting fish without classifying them.

Abinaya et al. [14] segmented the fish into three parts; head, body, and scales. Then an Alex-Net was used to classify each of these parts. Naive Bayesian Fusion (NBF) was then utilized to determine the final classification results. The accuracy obtained from this approach was quite well applied to the Fish-Pak [31] and BYU (Bringham Young University) [32] datasets. Still, the fish images used were only static, even though the narrative of this work was intended for an automatic sorting system. Mohamed et al. [16] proposed an approach for fish detection in aquaculture ponds. Image enhancement was used to improve fish detection in cloudy water conditions, and then YOLOv3 was utilized to detect the fish. However, this approach is not intended for classification but for counting and tracking fish trajectory. Xu et al. [17] applied Faster R-CNN and compared it with YOLOv3 to detect and record fish trajectory to study its behavior and relationship to ammonia levels in pond water.

However, the works reported in [7–9,11,30] recognized moving fish for underwater (ocean) environments, while some only detected fish without classifying them. Moving fish in aquaculture was discovered by [16,17], but not used for classification. The work in [14] classified fish with narration for the automatic sorting system, but the datasets used were static images. To the best of the authors' knowledge, there is no public dataset for cultured fish that run on conveyors, and there is no published work to detect and classify moving fish for the fish culture industry, especially for automatic sorting based on fish species using deep learning and computer vision. This paper will fill that gap, and it proposes a method for detecting and classifying fish and tests it on real videos of aquacultured freshwater fish moving along a conveyor belt for automatic sorting using deep learning and computer vision.

In summary, this work thus creates the following significant contributions:

1. We compiled our own dataset of eight cultivated fish species. The dataset contains not only static images, but also videos of fish running randomly at two different speeds on a conveyor belt (low and high).
2. This work employed YOLOv4, a very popular recognition algorithm, which was optimized with a unique labeling technique.
3. A trial study with several schemes was also conducted to determine their effectiveness. Among them are schemes for using data for training, versions of YOLOv4, and comparisons of labeling techniques.

By using the proposed method and using real videos of freshwater fish running on a conveyor, it is anticipated that this work will serve as a guide and provide solutions to the challenges of detecting and classifying fish for automatic sorting, which is very close to the actual condition.

2. Materials and Methods

2.1. Images Dataset and the Experimental Set-Up

The purpose of this work is to develop an approach to automatically detect and classify fish for automatic sorting systems in the fish industry. As far as the authors know, there is no publicly available dataset for cultured fish run on conveyors that can be used for this purpose. For that, we created our own dataset for this work. We took fish samples from eight types of farmed fish species called "Ben-Cak", which are generally bred, sold, and consumed in and around Thailand. Three of these fish species are endemic, originally from the Mekong and Chao Phraya rivers, which are also in Thailand. The eight fish species are:

1. Yeesok (*Labeo rohita*),
2. Nuanchan (*Cirrhinus microlepsis*),
3. Tapian (*Barbonymus gonionotus*),
4. Nai (*Cyprinus carpio*),
5. Jeen Ban (*Hypophthalmichthys molitrix*),
6. Jeen To (*Hypophthalmichthys nobilis*),
7. Nin (*Oreochromis niloticus*), and
8. Sawai (*Pangasianodon hypophthalmus*).

Sample pictures of each type of fish can be seen in Figure 1. In Thailand, these fish are bred together (mixed) in the same pond on the fish farm, and then when harvested, these fish will be brought to the fish hub for sorting and weighing for further sale to consumers. For this reason, these fish are considered very suitable for this work because the sorting process carried out at the fish hub is still done manually by humans.

(**a**) (**b**)

Figure 1. *Cont.*

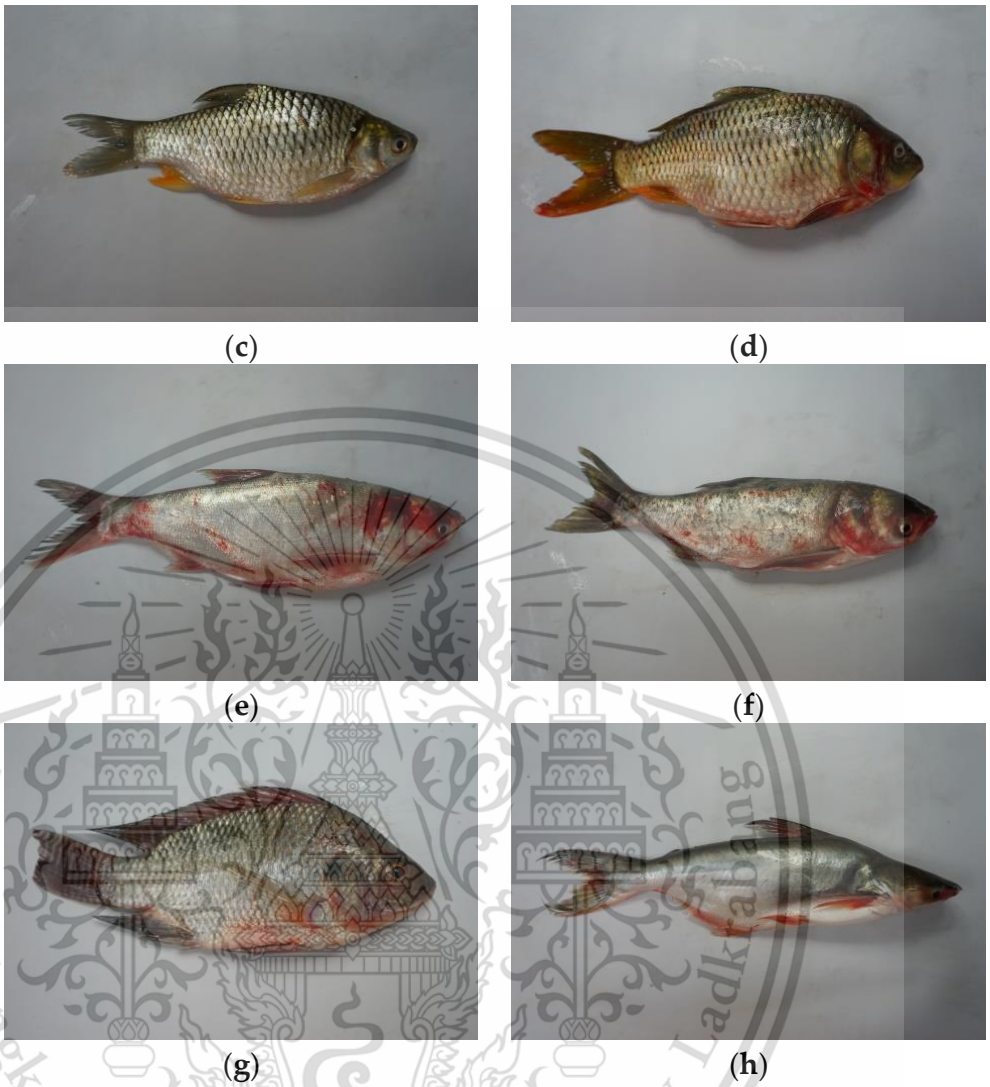


Figure 1. Sample picture of fish: (**a**) Yeesok; (**b**) Nuanchan; (**c**) Tapian; (**d**) Nai; (**e**) Jeen Ban; (**f**) Jeen To; (**g**) Nin; and (**h**) Sawai.

In creating the dataset, we did not just take static photos of each fish sample from several views, as shown in Figure 1. We ran a conveyor and randomly put the fish on it in both position and order. Then we recorded it with the overhead camera with two-speed settings; low (116.65 m/h) and high (505.08 m/h). The recordings were carried out in the Food Engineering laboratory at King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand, with room lighting conditions and additional light from an LED lamp. The measured light intensity was 846 lux/79 FC at all times during the recordings. We produced three videos; one low-speed video with a duration of 17 min and 13 s, and two high-speed videos, with a duration of 8 min and 24 s (later referred to as high-speed video 1) and 17 min and 13 s (later referred to as high-speed video 2). The camera used was a SONY Model ILCE-7 with a frame size setting of 1920 × 1080 (W × H) and a 29.97 frames/second frame rate.

In the videos produced, the background of the fish object is not entirely a conveyor, with a monotone condition. The conveyor is small and does not dominate the entire frame; it also has other objects in the background of the frame. So its condition makes this dataset more challenging. According to the authors, this is the first work that utilizes videos of aquacultured fish running on a conveyor. The experimental setup to create the dataset and the sample for capturing results can be seen in Figure 2. Tables 1 and 2 show images from the dataset that was created.

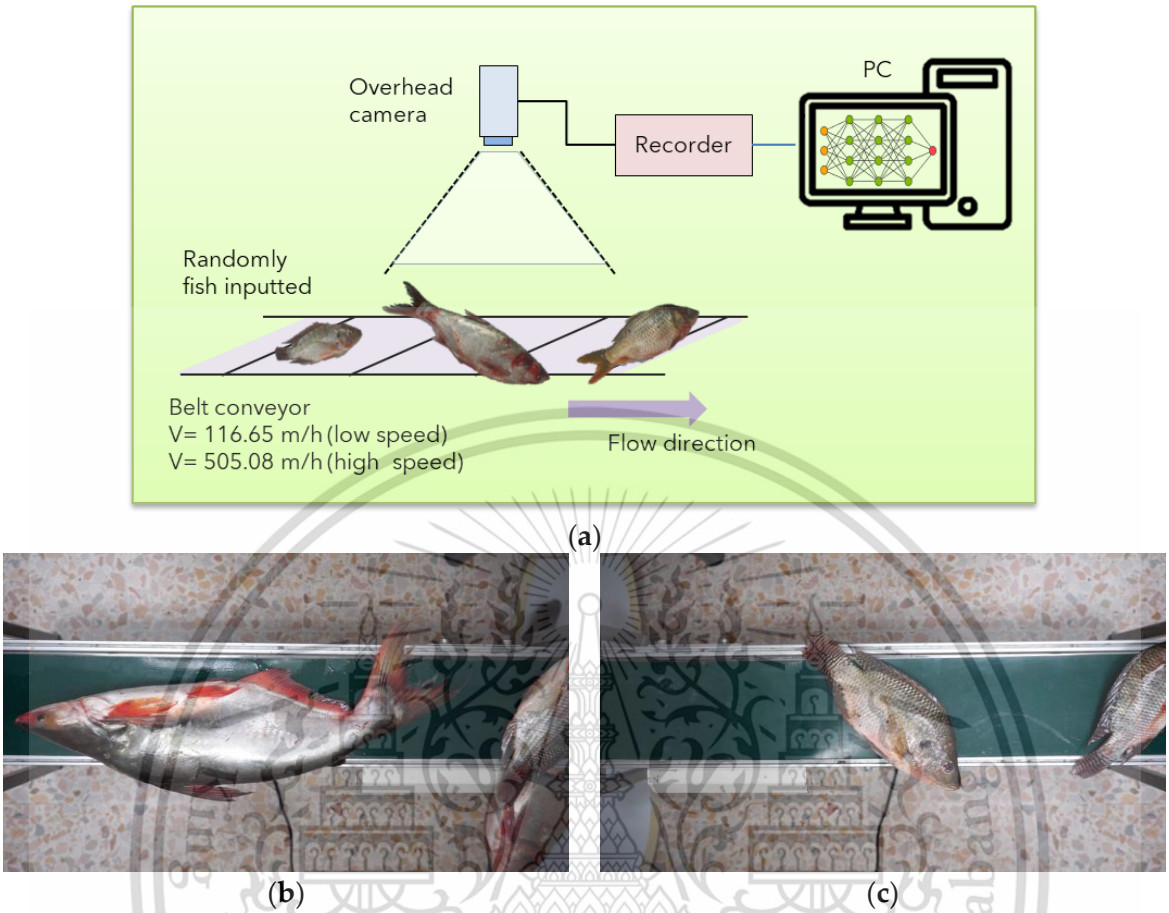


Figure 2. The experimental set-up for: (**a**) taking the videos; while (**b**,**c**) are examples of capturing video results.

Table 1. Images dataset (static pictures).

Fish Class	Yeesok	Nuanchan	Tapian	Nai	Jeen Ban	Jeen To	Nin	Sawai
No. of images	20	20	20	20	20	20	20	20
Total					160			
Average per class					20			

Table 2. Images dataset (videos).

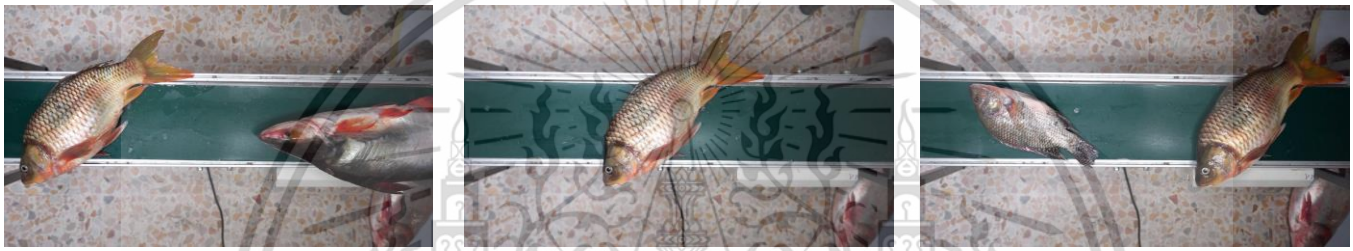
No.	Name	Conveyor Speed (m/h)	Duration	Note
1	low-speed video	116.65	17 min 13 s	later extracted for training data (scheme 2)
2	high-speed video 1	505.08	8 min 24 s	for testing data
3	high-speed video 2	505.08	17 min 13 s	for testing data

2.2. Training Images and Augmentation

The algorithm for detecting and classifying (recognition) needs to be trained, so images for training should be prepared. This work uses two schemes for using or generating training images. First, using static pictures from each fish class, and second, generating and using extracted pictures taken from one of the video recordings from the dataset. Each image prepared is then augmented in both the first and second schemas to enrich the data for a better training process [14,33,34]. The augmentation techniques used in this work

are vertical and horizontal flips, which are suitable for the case of fish objects running on conveyors [35].

Scheme 1 employed 160 static images of each fish from eight classes. The images were then augmented to enrich the data. Each augmentation technique was applied to each original image. So, every one of the 160 new images was obtained using vertical and horizontal flip techniques. All images are then combined (original and augmented) and used as training images. In scheme 2, the low-speed video was used and extracted into 188 static images. Each fish that appears in the video was extracted by taking screenshots at three positions, as shown in Figure 3: when the fish appears in its entirety (a), at the exact center position (b), and just before the fish's snout or tail hits the right-hand frame border to leave the frame (c). Then augmentation was applied to enrich the data with the same method as in scheme 1. Furthermore, training images were also obtained by combining the original images with augmentations, as in scheme 1 as well. The training images and augmentations used in this work are summarized in Table 3.



(**a**) (**b**) (**c**)

Figure 3. Example of extracting images for one fish from the low-speed video: (**a**) when the fish appears in its entirety; (**b**) at the exact center position; and (**c**) just before the fish's snout or tail hits the right-hand frame border to leave the frame.

Table 3. Images for training and augmentation.

Scheme	Original Images (8 Classes)	Averaged per Class	Augmented Images			Total Images for Training	Averaged per Class
			Flip-Vertically	Flip-Horizontally	Total		
Scheme-1 (from static pictures)	160	20	160	160	320	480	60
Scheme-2 (from extracted pictures)	188	24	188	188	376	564	71

2.3. Labeling Techniques

The labeling process is utilized during the training step. There are three types of labeling used in this work: conventional, using landmarking, and a combination of conventional and landmarking. The conventional labeling technique is the most commonly used, but it has a weakness. The object's background is included so that the algorithm's feature extraction is carried out both on the object and background. It can make the learning process less effective and impact the recognition results [14,36]. Labeling using the landmarking technique was introduced by the authors for the first time in previous works [37,38]. Table 4 provides a summary of the previously stated works. This table contains important findings regarding the effect of certain labeling techniques on the recognition accuracy of fish. The landmarking technique can optimize the recognition algorithm significantly, especially in object recognition in various background conditions. Because by using this technique, all of the object's backgrounds can be removed so that the recognition algorithm will extract only the object's features without the background. The difference between conventional labeling techniques and using landmarking can be observed in Figure 4. For the third method,

the combination labeling technique is proposed in this work. This technique combines conventional imaging with landmarking, which will be explained in detail in Section 2.4.

Table 4. Summary of recent works by authors that are related to this work.

References	Fish Object	Important Findings Related to This Work
[37]	- Static pictures. 6 classes. - Constant backgrounds. - Aquaculture fish with similar appearance and structural deformed. - Taken from the Fish-Pak dataset.	Applying YOLOv4 with conventional labeling resulted in 14.29% higher accuracy than using landmarking for 6 classes.
[38]	- Static pictures. 4 classes. - Various backgrounds. - Ocean fish images captured in various background conditions, such as rocks, water, seaweed, etc. - Taken from the BYU dataset.	Combining YOLOv4 with landmarking labeling techniques resulted in 4.94% higher accuracy than using conventional.

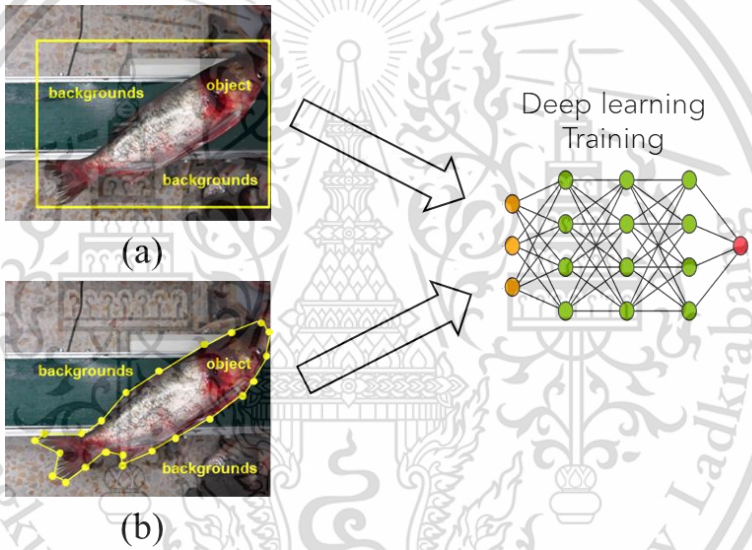


Figure 4. Labeling techniques in training process: (**a**) conventional; and (**b**) using landmarking.

2.4. YOLOv4, YOLOv4-Tiny, and the Training Process

In this work, YOLO (You Only Look Once) is employed as the recognition algorithm. This algorithm is very popular and known as a real-time object detector because of its speed and accuracy [39–43]. In addition, the version used is the relatively new, YOLO version 4, which was released in April 2020. We use this version because it accommodates our resources (currently, our resources only support this version), and we suppose that this version will achieve good performance with proper optimization. YOLOv4 consists of CSPDarknet53 as the backbone, SPP (Spatial Pyramid Pooling layer) & PAN (Path Aggregation Network) as the neck, and YOLOv3 as the head. A simple architecture of YOLOv4 is shown in Figure 5, and the output of this algorithm can be represented as [44]:

$$y = (P_c, B_y, B_x, B_w, B_h, C_1, C_2, \ldots C_8) \quad (1)$$

where y is the output of the YOLO, Pc will become 1 if the algorithm detects objects (fish) and 0 if otherwise, (B_y, B_x) is the center point of the produced bounding boxes of the fish, (B_w, B_h) is the width and height of the bounding boxes, and C_1 untill C_8 represent each class for the fish [44].

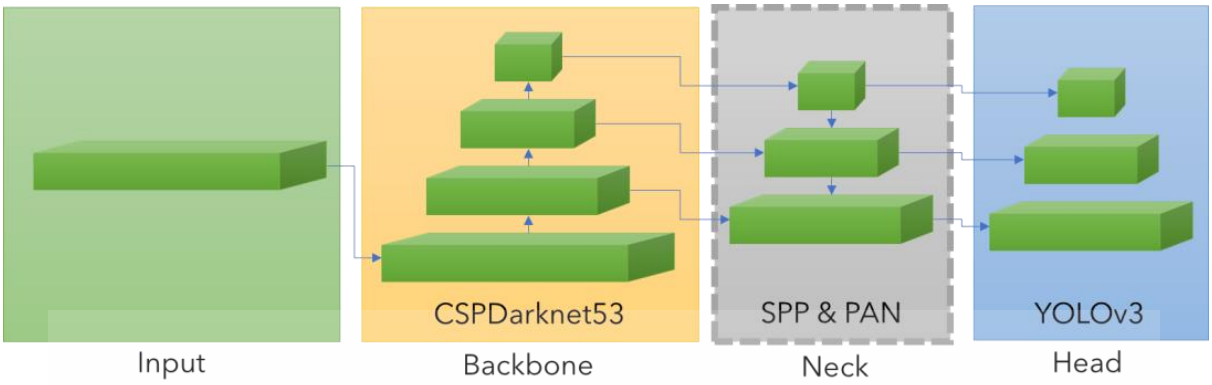


Figure 5. Simple architecture of YOLOv4 [44].

In this work, YOLOv4-Tiny is also used for study and comparison. YOLOv4-Tiny is a compressed and lite version of YOLOv4. The purpose of this compression is to reduce the computation so it can run on hardware with lower capacities, even for mobile or embedded devices. In this way, economic reasons can also be achieved. In addition, this algorithm has a higher speed but is less accurate than the full version (YOLOv4). In its research, YOLOv4-Tiny achieved 22.0% AP (average precision) or 42.0% AP50 at a speed of 443 FPS (frames per second), while YOLOv4 was able to achieve 43.5% AP or 65.7% AP50 at a real-time speed of 65 FPS for the MS COCO dataset. According to these results, YOLOv4-Tiny is approximately seven times faster than YOLOv4 but only has 2/3 of the accuracy. It makes YOLOv4-Tiny more suitable for cases that require high detection speed, unnecessarily high accuracy, applied to hardware with lower capacities (economic reasons), or for mobile or embedded devices [45]. YOLOv4-Tiny reduces the layers of some components of the original YOLOv4 to achieve a faster detection speed. First and foremost, the number of layers in the CSP backbone is reduced from 137 to only 29 pre-trained convolutional layers. In addition, YOLOv3 reduces the head from 3 to 2, and there are only a few anchor boxes for prediction [45].

These recognition algorithms were executed on CiRA-Core, a deep learning platform originally developed by Advanced Manufacturing Innovation (AMI) KMITL and first described in [46,47]. The advantages of this platform include its ease of use, user-friendliness, plethora of interface options, and provision of several automated processes, including during training, which will automatically select the most effective parameters for optimal results. In the training process, both YOLOv4 and YOLOv4-Tiny delivered good accuracy, between 0.15 and 0.03. The training was carried out with a batch size of 64 and 16 subdivisions, a learning rate of 0.001, and an adam optimizer. Data enrichment was carried out using rotation techniques, with a setting of 90 images per rotation (360°). The hardware used was a desktop PC with an Intel® 1151 Core™ i7-9700 3.0 GHz CPU (Central Processing Unit), NVIDIA GeForce RTX 3070 8 GB GDDR6 GPU (Graphical Processing Unit), and 32 GB DDR4/3200 RAM (Random Access Memory). Training time for each scheme took approximately 4–6 h.

2.5. Validation Matrix

The confusion matrix is used to evaluate the model's output in this work. This matrix is built from four blocks; TP, TN, FP, and FN. TP and TN are the basic truths, and FP and FN are the basic falsehoods. TP (True Positive) is defined as when the model can correctly detect the object (fish), TN (True Negative) is defined as when the model can correctly not detect the not-existent fish, which was not measured in this work, FP (False Positive) is defined as when the model incorrectly detects a fish, and FN (False Negative) is defined as when the model fails to detect the fish. From the confusion matrix, we can define the

accuracy to evaluate the model's output. It can be obtained from a comparison between the basic truth and the total blocks, as described by the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (2)$$

where, *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

In this work, the model's output is also categorized into three groups: correct detection, false detection (which includes wrong and double detection), and not-detect. Correct detection is defined as the detection and classification of the model that can be carried out correctly and consistently while the fish is fully visible in the frame. The wrong detection is determined if the model detects the wrong fish class more than once for more than 1 s. Double detection is specified if another fish class is also detected and appears more than once for more than 1 s. Not-detect is counted if the model cannot detect at all, can detect only momentarily (less than 1 s even though several times), or is unstable with a break of more than once for more than 1 s. The accuracy can be defined from a comparison between the number of correct detections and the total number of detections, as expressed in Equation (3). In addition to accuracy, several other parameters are also measured to evaluate the model by the confusion matrix, including precision, recall or sensitivity, specificity, and F-score. Those parameters are each obtained by Equations (4)–(7) [15].

$$Accuracy = \frac{\sum_i^N Pi}{\sum_i^N |Qi|} \times 100\% \quad (3)$$

where, $\sum_i^N Pi$ is the number of correct detections, and $\sum_i^N |Qi|$ is the total number of all detections.

$$Precission = \frac{TP}{TP + FP} \times 100\% \quad (4)$$

$$Recall/Sensitivity = \frac{TP}{TP + FN} \times 100\% \quad (5)$$

$$Specivity = \frac{TN}{TN + FP} \times 100\% \quad (6)$$

$$F - score = \frac{2 * Precission * Recall}{Precission + Recall} \times 100\% \quad (7)$$

3. Experimental Results and Discussion

After the image data for training was prepared, the algorithm was trained with various image data input schemes, labeling techniques, and network versions to determine which approach was the most effective, including the proposed method. The trained algorithm was then applied to two high-speed videos, 1 and 2, for evaluation (also referred to as video tests after). The entire flow of this work can be seen in Figure 6, while the experimental results are summarized in Tables 5 and 6, Figures 7–9. The following is the explanation and discussion for every test result's scheme.

3.1. Using Static Pictures for Training Data

In this scheme, static images were used as training data in YOLOv4. The use of static images for this training is described in Section 2.2. Then, the algorithm that had been trained was tested on the video tests. From the experimental results, the output accuracy obtained was very low. In video test 1, the model could only correctly detect 2 of a total of 71 fish/detections. This means that the accuracy obtained is only 2.82%. In video test 2, the model could correctly detect 11 of 171 fish/detections, so the accuracy obtained is only 6.43%. The final average accuracy of this model (the average accuracy of video tests 1 and 2) is only 4.62%.

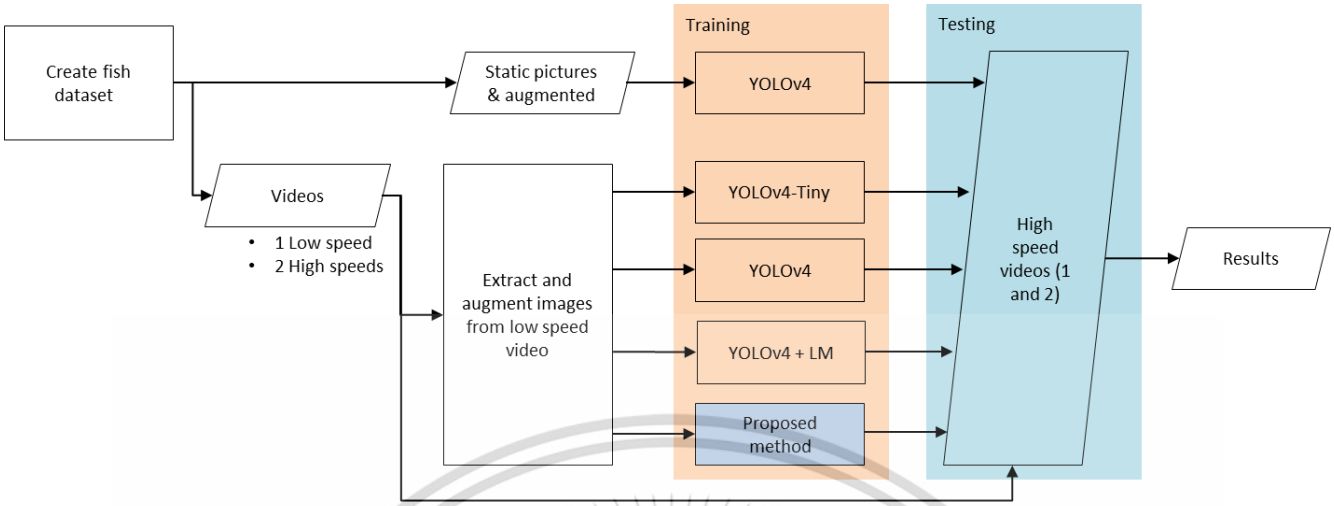


Figure 6. The work flowchart.

Certainly, the output accuracy result of this scheme is unacceptable. It indicates that using static images as training data for the algorithm, which is then used to recognize different images, is very ineffective. Although humans are able to classify fish species on static images, the learning process in the algorithm cannot capture the same thing. Static images have different sizes than the video tests used. The results of the accuracy and other performance parameters of this scheme can be seen in Figure 7.

3.2. With the Lite Version (YOLOv4-Tiny)

For the next step, the extracted pictures were used as training image data. Extracted pictures were obtained as described in Section 2.2. In this scheme, the lite version of YOLOv4 (YOLOv4-Tiny) was employed. This algorithm could be well trained and then tested on the same video tests. The output accuracy results obtained from the model were good enough. In video test 1, the model only detected four fish incorrectly out of 71 fish/detections. In video test 2, the model could correctly detect 154 out of 171 fish/detections while making 17 detection errors. This gives an accuracy score on video tests 1 and 2 of 94.37% and 90.06%, respectively, so the average score of both is 92.21%.

Of all the errors, the model made a double classification, i.e., one type of fish was detected as two different fish. It often appears between Yeesok and Nuanchan, Nuanchan and Tapian, and Jeen Ban and Jeen To. These fish are very similar to each other.

3.3. With YOLOv4 Using Conventional and Landmarking Labeling Techniques

At this stage, extracted pictures were used as training image data, and we used YOLOv4 as the algorithm. The first experiment was carried out using conventional labeling techniques. With this model, the accuracy output was better. In video test 1, an accuracy score of 97.18% was obtained, and in video test 2, it was 91.23%, so the average final accuracy of this model could reach 94.21%.

Although the final accuracy is only slightly better than with YOLOv4-Tiny, the total number of false (double) detections is much lower. From 4 to only 1 in video test 1, and from 17 to only 7 in video test 2. This means YOLOv4 has a higher classification accuracy capability than the lite version (YOLOv4-Tiny). However, another problem arose: the detection failure occurred nine times, all of which were in the Sawai class.

Table 5. Experimental results (accuracy).

Approches	Video Test-1					Video Test-2					Average (Final Accuracy) (%)
	Correct Detection	False (Double/Wrong) Detection	Not Detect	Total Detection	Accuracy (%)	Correct Detection	False (Double/Wrong) Detection	Not Detect	Total Detection	Accuracy (%)	
YOLOv4 with Static Pics	2	0	69	71	2.82	11	0	160	171	6.43	4.62
YOLOv4-Tiny	67	4	0	71	94.37	154	17	0	171	90.06	92.21
YOLOv4	69	1	1	71	97.18	156	7	8	171	91.23	94.21
YOLOv4 + LM	68	6	0	74	91.89	159	11	1	171	92.98	92.44
Proposed method	72	0	1	73	98.63	167	4	0	171	97.66	98.15

Table 6. Experimental results (other performance parameters).

Approches	Video Test 1 and 2						
	Correct Detection (TP)	Wrong/Double Detection (FP)	Not Detect (FN)	Total Detection	Precision (%)	Sensitivity (%)	F-Score (%)
YOLOv4 With Static Pics	13	0	229	242	100.00	5.37	10.20
YOLOv4-Tiny	221	21	0	242	91.32	100.0	95.46
YOLOv4	225	8	9	242	96.57	96.15	96.36
YOLOv4 + LM	227	17	1	245	93.03	99.56	96.19
Proposed Method	239	4	1	244	98.35	99.58	98.96

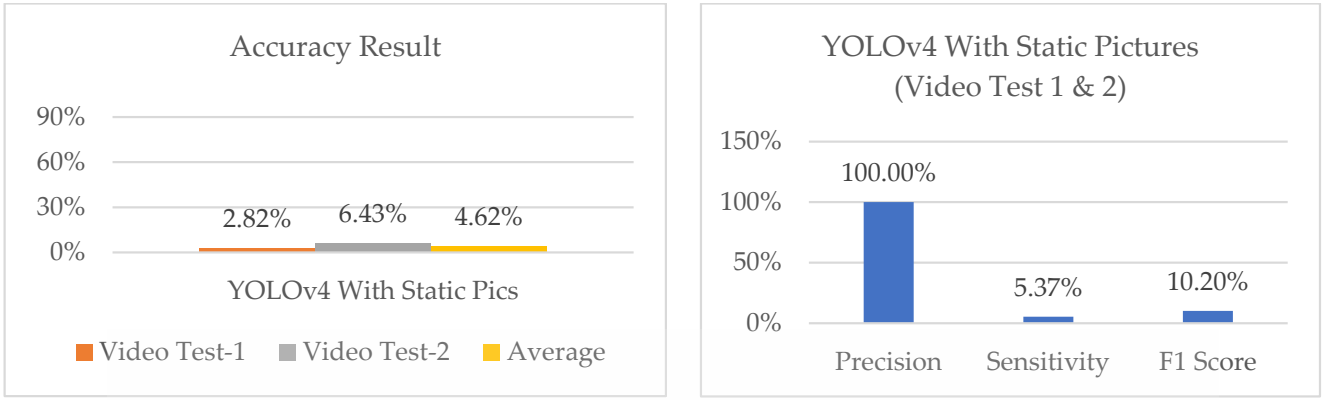


Figure 7. Test results of YOLOv4 with static pitures: (**a**) accuracy; (**b**) other parameters.

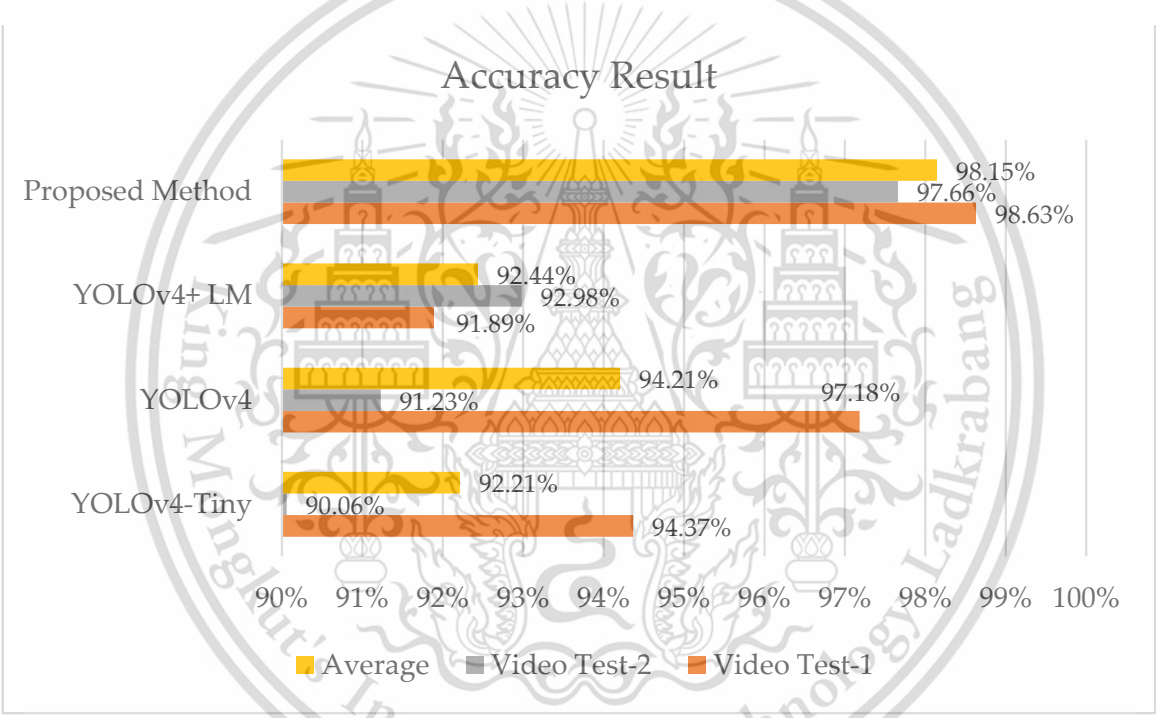


Figure 8. Experimental results (accuracy).

In the labeling process for the next scheme, YOLOv4 will be combined with the landmarking technique. With this approach, the results obtained were 91.89% accuracy for video test 1 and 92.98% accuracy for video test 2, so the average final accuracy was 92.44%. This model did many double detections, especially for the Nuanchan and Tapian, which have the same appearance of scales but different shapes. Many double detections were also carried out for the Jeen Ban-Jeen To classes. In addition, the model also detected scattered fish outside the conveyor incorrectly. The three scattered Yeesok fish were detected as Nin.

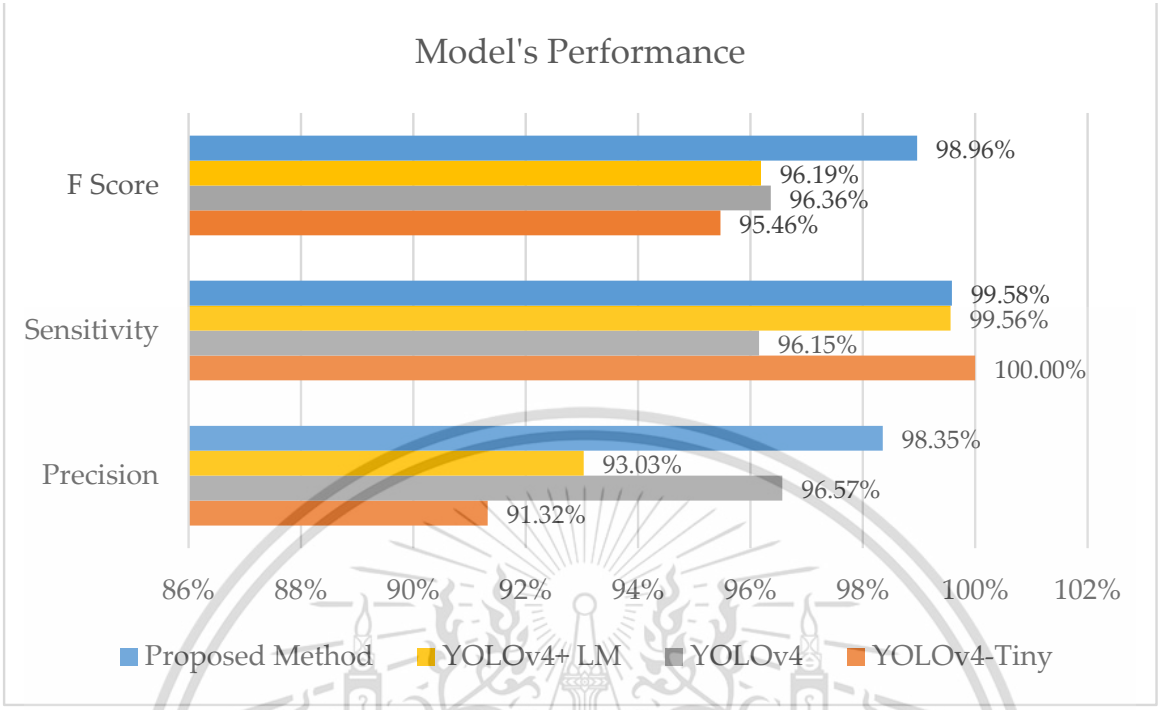


Figure 9. Experimental results (other performance parameters).

3.4. With the Prophosed Approach

Up to this step, some of the findings from various trials can be highlighted:

1. Using extracted pictures as training data provided much more effective results than static pictures.
2. YOLOv4 provided better accuracy results than its lite version (YOLOv4-Tiny).
3. Using conventional labeling techniques on YOLOv4 gave fairly accurate detection results, even for fish classes that were similar, but many of them failed to detect the Sawai class.
4. Combining YOLOv4 with the landmark labeling technique provided a fairly accurate detection result. Still, it generated many double detections for similar classes of fish, mostly for Nuanchan and Tapian, as well as Jeen Ban and Jeen To.

We can focus on the YOLOv4 algorithm with extracted pictures as image training data, and this approach produced the best accuracy. Then, we focus on the labeling technique used. The Sawai class was detected poorly using YOLOv4 and conventional labeling technique. If observed, this type of fish is the biggest. When it appears on the video, the size of this fish almost fills the entire frame. So that when extracted images of this Sawai fish are generated and used as a training image, utilizing conventional labeling techniques, many other objects in the fish's background will be included in the training process. These objects are not constant (not the same in every extracted picture), making the YOLOv4 algorithm less effective in capturing Sawai fish features because it mixes with other objects in the background [38].

In contrast to other smaller fish (seven other classes) such as Nin, Nuanchan, Tapian, and even Jeen Ban and Jeen To, even though the background is involved in the training process, the background tends to be constant (almost just a conveyor background). This condition, on the other hand, has a good effect on the learning process. The model only detects fish on the conveyor, so it does not detect fish scattered outside it. The model can better extract the shape features of the fish so that it can better distinguish between Nuanchan and Tapian fish, which have a similar appearance of scales, tails, and heads but can be distinguished by their shape.

YOLOv4 combined with the landmark labeling technique, resulted in lower classifiability for similar fish such as Nuanchan-Tapian and Jeen Ban-Jeen To. Because, as previously described, with this technique, the background of the fish is completely removed so that the algorithm is not affected by the background during the training process. The advantage of this approach is that the model can better detect Sawai fish because the algorithm is not affected by the background object, which is inconsistent. However, this makes the algorithm not good at extracting shape features. Hence, the model experiences a lot of false (double) detection for fish that should be able to be distinguished from their shapes, such as Nuanchan-Tapian and Jeen Ban-Jeen To. In addition, the model also detects fish that are outside the conveyor (scattered fish).

This hypothesis can be supported by visualizing the feature maps—the features captured by the algorithm during the learning process. The visualization of these feature maps can be seen in Figure 10 for the algorithm using the conventional labeling technique and in Figure 11 when using landmarking. The example taken is the same fish for easy comparison. Figure 10 shows that the background is included in the training process and extracted by the algorithm (see the first convolution process (conv2d)). Even the background is still carried over and becomes part of the extracted features in a deeper layer; pooling 1 (max_pooling2d), convolution 2 (conv2d_1), pooling 2 (max_pooling2d_1), and convolution 3 (conv2d_2). This means that the background also becomes one of the determinants or features that are considered when the algorithm runs to detect the fish. In addition, the features of the fish's shape are better because there is a background comparison.

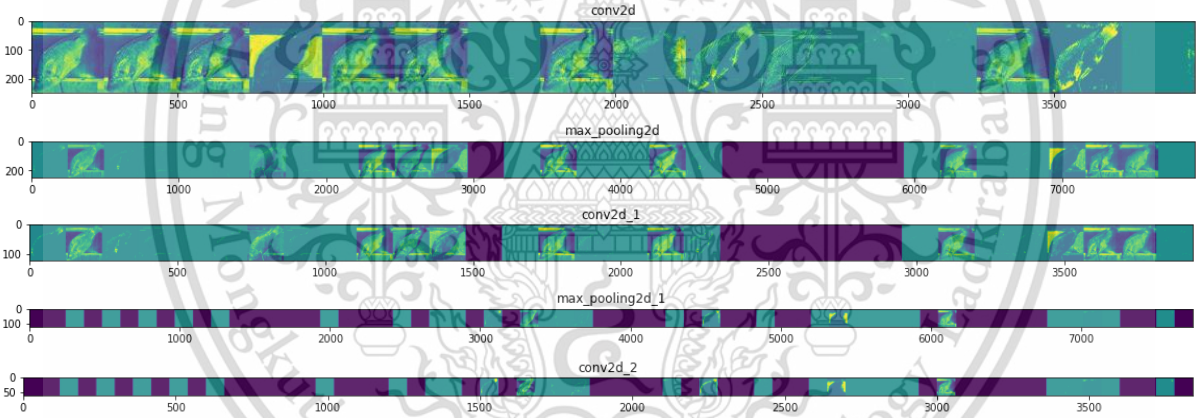


Figure 10. Feature maps visualization with conventional labeling.

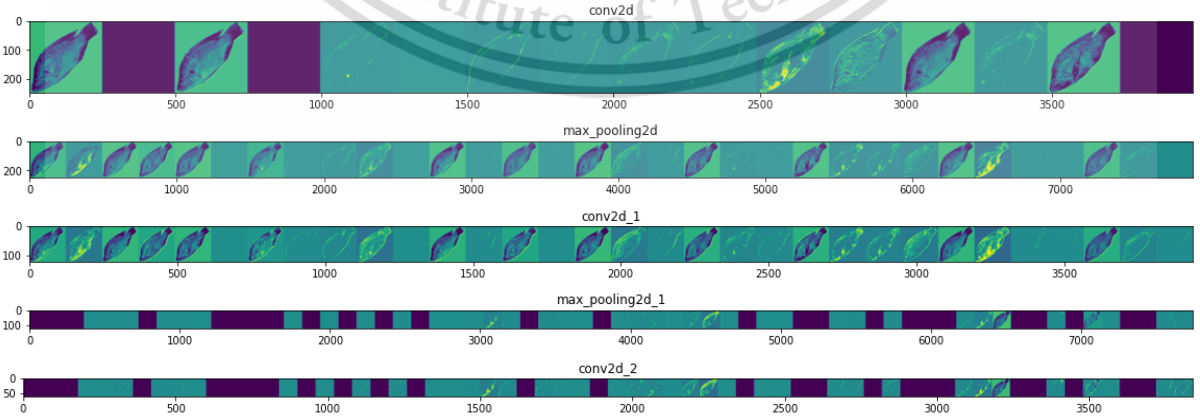


Figure 11. Feature maps visualization with landmarking labeling.

On the other hand, the extraction process only happens on fish objects on feature maps made from the learning process on YOLOv4 and the landmark labeling technique. This is because the background of the fish is completely removed with this technique. It can be seen in Figure 11. Because the learning process only focuses on the fish object, it is not disturbed by other objects. However, the extraction of shape features is less good because there is no background for comparison. For that, an approach is proposed. This approach uses extracted images as training data, YOLOv4, and a combination of labeling techniques. The combined labeling technique uses the landmarking technique for large fish (Sawai) and conventional for other classes. Then this approach is tested on the same video tests.

From the test results, the accuracy obtained was 98.63% for video test 1 and 97.66% for video test 2. This means that the average final accuracy was 98.15%. This achievement increased significantly compared to using the previous methods. There were only 4 false (double) detections, which had decreased a lot from the YOLOv4 with the landmark labeling technique (17 detections) and the conventional (8 detections). In addition, the detection failure was also significantly reduced to only one from the previous nine (using YOLOv4 with conventional labeling techniques). In other words, this approach can reach its optimum point by combining existing labeling techniques and avoiding each of their weaknesses.

3.5. Comparison with Recent State of Art

The method proposed in this work is compared to the current state of the art in order to identify its most significant contributions. Recent years have seen the development of at least six studies on recognizing swimming fish, which were also discussed in the Section 1. Table 7 provides a comparison summary. The table demonstrates that the proposed Method with simple and effective algorithms has the best average recognition results and has been tested on aquaculture fish video datasets; therefore, this work is most applicable to fish sorting systems in the fish industry.

3.6. Limitations and Future Developments

Limitations of this work also need to be reported. The biggest limitation is that the algorithm is standard and cannot be modified with the tools used in this work. So optimization studies cannot be carried out within the scope of modifications to the YOLOv4 or YOLOv4-Tiny algorithms. For this reason, this limitation can also be developed in the future. Optimization studies can be implemented by modifying them to achieve more optimal accuracy output or more efficient computations by the other tools. Or the use of a higher version (YOLO versions 5, 6, or 7) can also be considered. In addition, future developments can also be conducted using image processing techniques such as reducing glare on the appearance of fish objects or combining approaches with other classification algorithms or unsupervised learning. A tuning or inference method can be proposed to optimize the accuracy of the approach in this work. Moreover, a statistical hypothesis analysis could be conducted.

Table 7. Comparison chart of previously related works with the proposed method.

Related Work	Fish Dataset	Fish Type	No. of Fish Classes	Method/ Algorithm	Accuracy (%)	Precision (%)	Sensitivity (%)	F Score (%)	Advantage	Disadvantage
[7]	own dataset	deep ocean fish	20	CNN	94.90	-	-	-	(1) The best accuracy is achieved by recognizing nine species of fish. (2) Recognition results are accurate even though the backgrounds are various or the fish only partially appear.	(1) The results of accuracy can still be increased. (2) High accuracy is expected to apply to all classes.
[8]	own dataset	deep ocean fish	1	Multi-cascade object detection Network, 7 CNNs 2 RPNs, trained LSTMs	-	67.28	68.25	67.76	Promising to detect and count fish under various benthic backgrounds and illumination conditions.	Only detect fish, not classify them.
[9]	Fish4-Knowledge & UWA	deep ocean fish	17	Optical flow, GMM, ResNet-50, YOLOv3	91.64	-	-	95.47	Quite effective, even applied to many classes with diverse backgrounds and illumination challenges.	The results of accuracy can still be improved
[16]	own dataset	Aquacu-ltured fish	1	Image enhancement, YOLOv3	100	-	-	-	(1) Effectively detect all fish in the test images. (2) Image Enhancement can optimize the work of the algorithm significantly.	Only to detect fish and trajectory, not for classification.
[17]	own dataset	Aquacu-ltured fish	1	Faster R-CNN, YOLOv3	98.13	-	-	-	(1) High accuracy is obtained from Faster R-CNN. (2) Simple with good results	Only to detect fish and trajectory, not for classification.
[30]	Fish4-Knowledge	deep ocean fish	1	GMM, Pixel-wise posteriors, CNN	-	-	-	87.44	Increased the result fairly from the previous work.	(1) Only detect fish, not classify them. (2) The result can still be improved
Proposed method	own dataset	Aquacu-ltured fish	8	Optimized YOLOv4	98.15	98.35	99.58	98.96	(1) Simple method but delivers high results. (2) Ready to implement for aquaculture fish sorting system.	(1) Using not open access deep learning software. (2) YOLOv4 can not be modified.

-: not reported.

4. Conclusions

This work aims to propose an optimal approach for detecting and classifying fish intended for the aquaculture industry, especially for making automatic sorting processes. In this work, we created and presented a real video dataset of freshwater fish running on a conveyor, which is the first and only, as far as the authors know. The dataset includes eight types of freshwater fish that are grown and eaten most often in Thailand and nearby countries. Some of these fish are native to Thailand. This work uses YOLOv4, the most viral algorithm for object detection, and a relatively new version. Several studies were conducted to determine the level of accuracy, and finally, an approach was proposed.

This approach utilizes YOLOv4, optimized with a combination/custom labeling technique, and extracted images as training data. From the test results on the video of eight types of freshwater fish running on a conveyor with a total duration of 25 min and 37 s at a speed of 505.08 m/h, the model could produce an accuracy of 98.15%. These results are considered quite good and can even be improved in the future. By using real videos of freshwater fish running on a conveyor, this work is expected to contribute to the development of fish detection and classification, especially for the automatic sorting process in the fish industry, which is very close to real conditions.

Author Contributions: Conceptualization, A.K. and T.S.; methodology, W.T. and N.N.; software, A.K. and G.S.; validation, W.T. and G.S.; formal analysis, A.K. and N.N.; investigation, W.T. and G.S.; resources, T.S. and N.N.; data curation, A.K. and T.S.; writing—original draft preparation, A.K.; writing—review and editing, A.K., T.S., W.T., N.N. and G.S.; visualization, A.K. and N.N.; supervision, W.T. and G.S.; project administration, W.T.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: King Mongkut's Institute of Technology Ladkrabang Research Fund: KREF206314

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset created and the results videos can be found at: https://drive.google.com/drive/folders/1OrEUIYdiDWbvDyUV46WSrr6Yd83OTuwt?usp=sharing (accessed on 1 March 2023). The python codes and repository are shared at: https://github.com/AriKus77/Fish-Recognition-for-Auto-Sorting--Feature-Extraction (accessed on 1 March 2023).

Acknowledgments: We want to thank AMI (Advanced Manufacturing Innovation)-KMITL, Bangkok, Thailand, for granting the CiRA-Core software license, and declare that this work was fully supported by King Mongkut's Institute of Technology Ladkrabang (KMITL).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, D.; Wang, Q.; Li, X.; Niu, M.; Wang, H.; Liu, C. Recent advances of machine vision technology in fish classification. *ICES J. Mar. Sci.* **2022**, *79*, 263–284. [CrossRef]
2. Alsmadi, M.K.; Almarashdeh, I. A survey on fish classification techniques. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *34*, 1625–1638. [CrossRef]
3. Saleh, A.; Sheaves, M.; Azghadi, M.R. Computer vision and deep learning for fish classification in underwater habitats: A survey. *Fish Fish.* **2022**, *23*, 977–999. [CrossRef]
4. Zhao, S.; Zhang, S.; Liu, J.; Wang, H.; Zhu, J.; Li, D.; Zhao, R. Application of machine learning in intelligent fish aquaculture: A review. *Aquaculture* **2021**, *540*, 736724. [CrossRef]
5. Li, D.; Du, L. Recent advances of deep learning algorithms for aquacultural machine vision systems with emphasis on fish. *Artif. Intell. Rev.* **2022**, *55*, 4077–4116. [CrossRef]
6. Yang, X.; Zhang, S.; Liu, J.; Gao, Q.; Dong, S.; Zhou, C. Deep learning for smart fish farming: Applications, opportunities and challenges. *Rev. Aquac.* **2021**, *13*, 66–90. [CrossRef]
7. Villon, S.; Mouillot, D.; Chaumont, M.; Darling, E.S.; Subsol, G.; Claverie, T.; Villéger, S. A deep learning method for accurate and fast identification of coral reef fishes in underwater images. *Ecol. Inform.* **2018**, *48*, 238–244. [CrossRef]
8. Labao, A.B.; Naval, P.C. Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild. *Ecol. Inform.* **2019**, *52*, 103–121. [CrossRef]

9. Jalal, A.; Salman, A.; Mian, A.; Shortis, M.; Shafait, F. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecol. Inform.* **2020**, *57*, 101088. [CrossRef]
10. Villon, S.; Iovan, C.; Mangeas, M.; Claverie, T.; Mouillot, D.; Villéger, S.; Vigliola, L. Automatic underwater fish species classification with limited data using few-shot learning. *Ecol. Inform.* **2021**, *63*, 101320. [CrossRef]
11. Salman, A.; Maqbool, S.; Khan, A.H.; Jalal, A.; Shafait, F. Real-time fish detection in complex backgrounds using probabilistic background modelling. *Ecol. Inform.* **2019**, *51*, 44–51. [CrossRef]
12. Victor, N.; Alazab, M.; Bhattacharya, S.; Magnusson, S.; Maddikunta, P.K.R.; Ramana, K.; Gadekallu, T.R. Federated learning for IoUT: Concepts, applications, challenges and opportunities. *arXiv* **2022**, arXiv:2207.13976. [CrossRef]
13. Bhattacharya, S.; Victor, N.; Chengoden, R.; Ramalingam, M.; Selvi, G.C.; Maddikunta, P.K.R.; Donta, P.K.; Dustdar, S.; Jhaveri, R.H.; Gadekallu, T.R. Blockchain for internet of underwater things: State-of-the-art, applications, challenges, and future directions. *Sustainability* **2022**, *14*, 15659. [CrossRef]
14. Abinaya, N.S.; Susan, D.; Kumar, R. Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries. *Ecol. Inform.* **2021**, *61*, 101248. [CrossRef]
15. Ahmed, M.S.; Aurpa, T.T.; Azad, M.A.K. Fish disease detection using image based machine learning technique in aquaculture. *J. King Saud Univ.-Comput. Inf. Sci.* **2021**, *34*, 5170–5182. [CrossRef]
16. Mohamed, H.E.-D.; Fadl, A.; Anas, O.; Wageeh, Y.; ElMasry, N.; Nabil, A.; Atia, A. MSR-YOLO: Method to enhance fish detection and tracking in fish farms. *Procedia Comput. Sci.* **2020**, *170*, 539–546. [CrossRef]
17. Xu, W.; Zhu, Z.; Ge, F.; Han, Z.; Li, J. Analysis of behavior trajectory based on deep learning in ammonia environment for fish. *Sensors* **2020**, *20*, 4425. [CrossRef]
18. Waleed, A.; Medhat, H.; Esmail, M.; Osama, K.; Samy, R.; Ghanim, T.M. Automatic recognition of fish diseases in fish farms. In Proceedings of the 2019 14th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 17 December 2019; IEEE: Manhattan, NY, USA, 2020. [CrossRef]
19. Ubina, N.; Cheng, S.-C.; Chang, C.-C.; Chen, H.-Y. Evaluating fish feeding intensity in aquaculture with convolutional neural networks. *Aquac. Eng.* **2021**, *94*, 102178. [CrossRef]
20. Bader, F.; Rahimifard, S. Challenges for industrial robot applications in food manufacturing. In Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, Stockholm, Sweden, 21–23 September 2018; Association for Computing Machinery (ACM): New York, NY, USA, 2018. [CrossRef]
21. Goncharuk, A. Food business and food security challenges in research. *J. Appl. Manag. Invest.* **2015**, *4*, 223–230.
22. Vo, T.T.E.; Ko, H.; Huh, J.-H.; Kim, Y. Overview of smart aquaculture system: Focusing on applications of machine learning and computer vision. *Electronics* **2021**, *10*, 2882. [CrossRef]
23. Gladju, J.; Kamalam, B.S.; Kanagaraj, A. Applications of data mining and machine learning framework in aquaculture and fisheries: A review. *Smart Agric. Technol.* **2022**, *2*, 100061. [CrossRef]
24. Wu, Y.; Zhuang, R.; Cui, Z. Automatic sorting system of large yellow croaker based on machine vision. In Proceedings of the 2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), Shenzhen, China, 9–11 May 2019; IEEE: Manhattan, NY, USA, 2019. [CrossRef]
25. Tappi, S.; Rocculi, P.; Ciampa, A.; Romani, S.; Balestra, F.; Capozzi, F.; Dalla Rosa, M. Computer vision system (CVS): A powerful non-destructive technique for the assessment of red mullet (*Mullus barbatus*) freshness. *Eur. Food Res. Technol.* **2017**, *243*, 2225–2233. [CrossRef]
26. Li, C.; Zhen, T.; Li, Z. Image classification of pests with residual neural network based on transfer learning. *Appl. Sci.* **2022**, *12*, 4356. [CrossRef]
27. Li, W.; Zhang, L.; Wu, C.; Cui, Z.; Niu, C. A new lightweight deep neural network for surface scratch detection. *Int. J. Adv. Manuf. Technol.* **2022**, *123*, 1999–2015. [CrossRef]
28. Fisher, R.B.; Chen-Burger, Y.-H.; Giordano, D.; Hardman, L.; Lin, F.-P. *Fish4Knowledge: Collecting and Analyzing Massive Coral Reef Fish Video Data*; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
29. Siddiqui, S.A.; Salman, A.; Malik, M.I.; Shafait, F.; Mian, A.; Shortis, M.R.; Harvey, E.S. Automatic fish species classification in underwater videos: Exploiting pre-trained deep neural network models to compensate for limited labelled data. *ICES J. Mar. Sci.* **2018**, *75*, 374–389. [CrossRef]
30. Salman, A.; Siddiqui, S.A.; Shafait, F.; Mian, A.; Shortis, M.R.; Khurshid, K.; Ulges, A.; Schwanecke, U. Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system. *ICES J. Mar. Sci.* **2020**, *77*, 1295–1307. [CrossRef]
31. Shah, S.Z.H.; Rauf, H.T.; IkramUllah, M.; Khalid, M.S.; Farooq, M.; Fatima, M.; Bukhari, S.A.C. Fish-pak: Fish species dataset from Pakistan for visual features based classification. *Data Brief* **2019**, *27*, 104565. [CrossRef]
32. Lillywhite, K.D.; Lee, D.J. Robotic Vision Lab, Brigham Young University, Fish Dataset. 2013. Available online: http://roboticvision.groups.et.byu.net/Machine_Vision/BYUFish/BYU_Fish.html (accessed on 19 September 2021).
33. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
34. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv* **2017**, arXiv:1712.04621. [CrossRef]
35. Liu, Z.; Jia, X.; Xu, X. Study of shrimp recognition methods using smart networks. *Comput. Electron. Agric.* **2019**, *165*, 104926. [CrossRef]

36. Bouwmans, T.; Javed, S.; Sultana, M.; Jung, S.K. Deep neural network concepts for background subtraction: A systematic review and comparative evaluation. *Neural Netw.* **2019**, *117*, 8–66. [CrossRef]
37. Kuswantori, A.; Suesut, T.; Tangsrirat, W.; Nunak, N. Development of object detection and classification with YOLOv4 for similar and structural deformed fish. *EUREKA Phys. Eng.* **2022**, *2*, 154–165. [CrossRef]
38. Kuswantori, A.; Suesut, T.; Tangsrirat, W.; Satthamsakul, S. Fish recognition optimization in various backgrounds using landmarking technique and YOLOv4. In Proceedings of the the 37th International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Phuket, Thailand, 5–8 July 2022; IEEE: Manhattan, NY, USA, 2022. [CrossRef]
39. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A review of yolo algorithm developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. [CrossRef]
40. Diwan, T.; Anirudh, G.; Tembhurne, J.V. Object detection using YOLO: Challenges, architectural successors, datasets and applications. *Multimed. Tools Appl.* **2022**, *82*, 9243–9275. [CrossRef] [PubMed]
41. Chandana, R.; Ramachandra, A. Real time object detection system with YOLO and CNN models: A review. *arXiv* **2022**, arXiv:2208.00773. [CrossRef]
42. Zhao, Z.-Q.; Zheng, P.; Xu, S.-T.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
43. Shetty, A.K.; Saha, I.; Sanghvi, R.M.; Save, S.A.; Patel, Y.J. A review: Object detection models. In Proceedings of the 2021 6th International Conference for Convergence in Technology (I2CT), Pune, India, 2–4 April 2021; IEEE: Manhattan, NY, USA, 2021. [CrossRef]
44. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
45. Jiang, Z.; Zhao, L.; Li, S.; Jia, Y. Real-time object detection method based on improved YOLOv4-tiny. *arXiv* **2020**, arXiv:2011.04244. [CrossRef]
46. Kittichai, V.; Pengsakul, T.; Chumchuen, K.; Samung, Y.; Sriwichai, P.; Phatthamolrat, N.; Tongloy, T.; Jaksukam, K.; Chuwongin, S.; Boonsang, S. Deep learning approaches for challenging species and gender identification of mosquito vectors. *Sci. Rep.* **2021**, *11*, 4838. [CrossRef]
47. Kittichai, V.; Kaewthamasorn, M.; Thanee, S.; Jomtarak, R.; Klanboot, K.; Naing, K.M.; Tongloy, T.; Chuwongin, S.; Boonsang, S. Classification for avian malaria parasite plasmodium gallinaceum blood stages by using deep convolutional neural networks. *Sci. Rep.* **2021**, *11*, 16919. [CrossRef]