water

MDPI

Article

Underwater Fish Detection and Counting Using Mask Regional Convolutional Neural Network

Teh Hong Khai [1], Siti Norul Huda Sheikh Abdullah [1,*], Mohammad Kamrul Hasan [1,*] and Ahmad Tarmizi [2]

[1] Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia; hongkhaiteh1996@gmail.com
[2] Mahjung Aquabest Hatchery, Lot 9569 Kampung Batu 4, Segari 32200, Malaysia; atarmiziphashim@gmail.com
* Correspondence: snhsabdullah@ukm.edu.my (S.N.H.S.A.); mkhasan@ukm.edu.my or hasankamrul@ieee.org (M.K.H.)

Abstract: Fish production has become a roadblock to the development of fish farming, and one of the issues encountered throughout the hatching process is the counting procedure. Previous research has mainly depended on the use of non-machine learning-based and machine learning-based counting methods and so was unable to provide precise results. In this work, we used a robotic eye camera to capture shrimp photos on a shrimp farm to train the model. The image data were classified into three categories based on the density of shrimps: low density, medium density, and high density. We used the parameter calibration strategy to discover the appropriate parameters and provided an improved Mask Regional Convolutional Neural Network (Mask R-CNN) model. As a result, the enhanced Mask R-CNN model can reach an accuracy rate of up to 97.48%.

Keywords: deep learning; counting; shrimp detection; underwater fish; machine learning

Citation: Hong Khai, T.; Abdullah, S.N.H.S.; Hasan, M.K.; Tarmizi, A. Underwater Fish Detection and Counting Using Mask Regional Convolutional Neural Network. *Water* **2022**, *14*, 222. https://doi.org/10.3390/w14020222

Academic Editors: Celestine Iwendi and Thippa Reddy Gadekallu

Received: 30 November 2021
Accepted: 4 January 2022
Published: 12 January 2022

1. Introduction

Shrimp counting is essential for farmers to estimate and manage hatching. The domain of importance in the agricultural field can help farmers improve their resources' productivity. An accurate automated shrimp detection and counting algorithm can enable farmers to optimize and streamline their hatching period. However, counting shrimp from images is a challenging task for several reasons, including the small size of shrimp and their transparent color, which we cannot easily see. An additional challenge to shrimp counting that is not present in shrimp detection is distinguishing multiple overlapping shrimps.

Shrimp farmers who use ponds for production rely on cast netting shrimp and then relating the amount caught in the surface area of the cast net to the surface area of the entire pond (Figure 1a,b). While this has been useful for estimating growth, it is not a reliable predictor of survival. The farmer must drain the tank, collect the shrimp in a mesh bag and weigh them out of the water to accurately count the shrimps in a tank culture system. Holding shrimp in water causes stress due to a lack of dissolved oxygen and the fact that they are highly concentrated when packed for weighing, which can take up to two minutes. This increase in stress and potential exoskeleton damage increases mortality and hastens the spread of black spot occurrences (Figure 1c). The unresolved challenges are when the shrimp's photographs can be visualized in three difficulties, as shown in Figure 1c for low, Figure 1d for intermediate and Figure 1e for shrimps with high density is captured from the very low- or very high-contrast images. Furthermore, the amount of water adhering to the shrimp must be considered in this method.

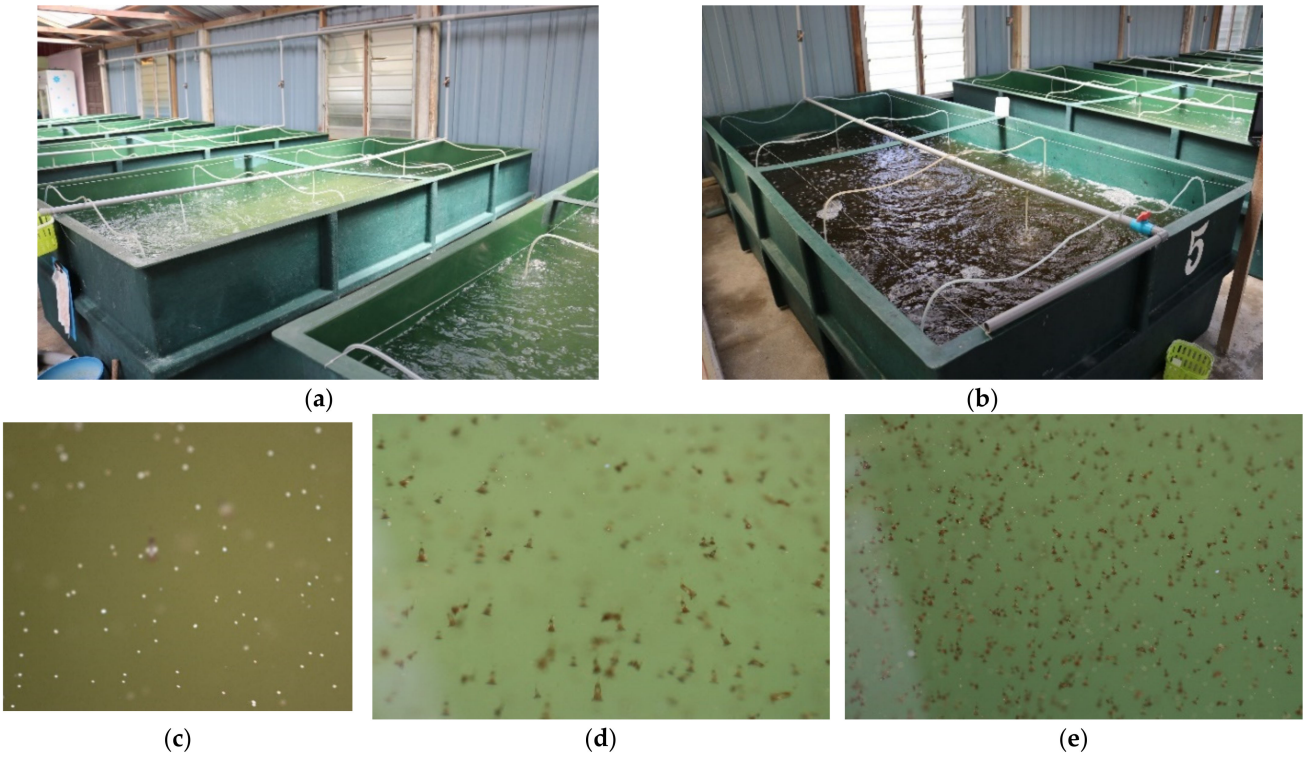


Figure 1. The challenges of shrimp detection using CNN with various illumination types: (**a**) low and (**b**) high contrast, from the side view; (**c**) low, (**d**) intermediate, and (**e**) high density of very low-contrast images due to its transparent color from the top view.

Deep learning is an obvious choice for dealing with cluttered scenes where conventional vision analytics machine methods struggle with semantic segmentation. This paper presents a novel pipeline that accurately estimates shrimp counting. The board steps of the pipelines are:

- Capture a set of shrimps images;
- Label the shrimp in each of the images captured;
- Train a ResNet101 backbone by using a default parameter to perform image segmentation;
- Train a ResNet101 backbone by performing image segmentation with the parameter calibration; and
- Use a parameter calibration to train a convolutional network to take the segmented image and output an intermediate estimate of shrimp counting.

This study aims to provide a data-driven shrimp counting methodology that a farmer can implement in cluttered farm settings. As a result, the proposed method focuses on labeling and training in speed and accuracy. The major contributions in this paper are:

- The application of a region-based convolutional network to accurately detect the shrimp; and
- The application of a region-based convolutional network to accurately count the number of shrimp.

The rest of the paper is organized as follows: (1) a summary of related work on counting approaches, (2) methods used with the introduction of Mask R-CNN and improved Mask-RCNN, (3) experimental results and analysis, and (4) conclusion.

2. Related Work

Detection with an accurate counting approach can apply non-machine learning, machine learning, and counting based on deep learning.

Figure 2 shows a conceptual diagram of underwater fish detection and recognition. The factors related to the technological factors are hardware availability, cost, and software

modularity. On the other hand, organizational factors such as knowledge sharing and management support are essential to maintain the shrimp counting system. Finally, other factors contributed to the resource factors such as dataset, deep learning skills, and fishery farm availability. These three aspects have been established to impact the researchers' motivation to study underwater fish detection and recognition.

Figure 2. Conceptual diagram of this study.

The literature review and the findings of the qualitative interviews highlighted the current problem, which is a counting process with a long period, and the results are also inaccurate. This pitfall is attributable to the weakness of a farmer's limited human resources in the counting process. Our study focused on the counting process using the deep learning-based algorithm in underwater fish detection and recognition.

At the cutting edge of the 4th Industrial Revolutionary and COVID-19 pandemic, many easy and intermediate human tasks are systematically transforming into computerized decision tools to reduce production costs and increase production goods while maintaining social distancing among workers. Likewise, the Internet of Things is widely booming and applied across sectors to keep up with the industrial needs and better quality of life. Hence, computer vision has become vital in place of many manual inspection systems. They were beginning from hyperspectral images until nano images such as a satellite in a study by Anahita et al. [1], cell detection by Yazan et al. [2], optical character recognition by Tarik et al. [3], vehicle counting by Abbas et al. [4], and rice diseases diagnosis by Abdullah et al. [5]. Furthermore, machine learning enhancement that can reverse feature engineering, namely deep learning, has attracted many researchers to explore the computer vision research area in place of handcrafted feature engineering. This extraordinary capability of imitating nature activity using a convolutional neural network ignites the implementation of shrimp counting to maximize productivity at the various growth stages.

2.1. Handcrafted Feature Engineering

Calculating geometrical features is a suitable manual inspection method in the industry or agriculture sector. Many researchers imitate how human vision encapsulates the optical property of the continuous [2,3,5] or discontinuous [4,6] pattern or shape. Usually, continuous property involves the characteristic and structured properties of the line and circularity. However, a discontinuous feature includes an irregular edge or circularity. Another significant matter in handcrafted feature engineering is scale-invariant. Many remarkable feature engineering inventions can address scale-invariant with the employment of a non-linear function [1] as Wavelet, SIFT, BOW, SURF; however, such approaches are less tolerable or robust when dealing with low-contrast and high-contrast images. Again, their procedures entail an extensive, long processing time for the training model.

Previous authors also use a statistical approach [2–5] to indicate discontinuous geometrical properties. Yazan et al. [2] propose an iterative randomized irregular circular algorithm (IRIC) that detects irregular and overlapping cells by initiating dynamic initialization to determine the best circle candidate with a lower probability of incorrectly detecting circles. Finally, they accumulate the number of spheres to indicate the proliferation rate estimation of brain cell cancer. They also proved that their IRIC outperformed other existing algorithms. The vehicle counting system by Abbas et al. [4] strikes research attention by utilizing the exploited contour-based approach to count vehicles according to its vehicle classification emitting into a free flow entrance. Their approach has also shown remarkable performance compared to the state-of-the-art methods.

Similarly, an automatic rice disease inspection by Abdullah et al. [5] using handcrafted feature engineering can withstand only specialized irregular shape features based on their proposed bi-level thresholding. We could summarize that handcrafted feature engineering is adequate to address specified objects and static scale-invariant less. It relies on a typical environment setting upon all the above geometrical feature engineering expensive when dealing with scale-invariant, overlapping, and low- and high-contrast environments. Moreover, it would be costly when dealing with scale-invariant, overlapping, and low- and high-contrast environments.

2.2. Autocrafted Feature Engineering

Autocrafted feature engineering is also known as automated feature engineering. Automated feature engineering aims to assist data scientists with feature creation by automatically creating hundreds or thousands of new features from a dataset. Feature tools, the sole library for automated feature engineering currently available, cannot replace the data scientist. Still, they can free up the data scientist's time to focus on more valuable aspects of the machine learning pipeline, such as delivering robust models into production. Automated feature engineering has solved one problem but created another problem: too many features. Too many features can lead to poor model performance because the less valuable features drown out more important features. In image recognition [7], many feature extraction methods became obsolete with deep learning. The same for natural language processing [8], where recurrent neural networks made much feature engineering obsolete.

Furthermore, Luis et al. [9] used FETEX 2.0, a software application for automatic discriminative feature descriptor from image objects, to describe agricultural parcels and provide various information. They successfully obtained the output file of a table produced in four different formats, each containing a vector of features for each object processed. Nonetheless, with the help of FETEX 2.0, they successfully integrated additional data to advance the improvement of land use and land cover database classification and agricultural database updating processes. A novel system, "Cognito" by Udayan et al. [10], strikes research attention by performing automatic feature engineering on a given dataset for supervised learning. The users were allowed to specify the domain or data-specific choices to prioritize the exploration. They successfully presented the design and operation of Cognito and demonstrated its efficacy by using eight real datasets.

On the other hand, a computer vision system created by Jingyao et al. [11] for robotic weed management can detect agricultural plants at various stages of growth using depth-based and color-based information. Data pre-processing, vegetation pixel segmentation, plant extraction, feature extraction, feature-based localization refinement, and crop plant classification were all techniques used in the image processing pipeline. As a result, they enhanced the average segmentation scores from 76.4% to 92.4%. This study also observed that the lettuce shape is more complex to localize than the broccoli shape.

2.3. Non-Machine Learning-Based

There are several methods in counting based on non-machine learning, namely blob counting [12], counting by detecting the pixel area [13], and shape analysis [14]. The input image is segmented into blobs of moving objects, using background subtraction and shadow

elimination. Various features are extracted and normalized for each blob according to its approximate size in the actual scene. The number of objects could estimate simultaneously in each blob [15]. Subdurally et al. [16] proposed that the methods based on observation are significantly more visible and easily distinguishable than any other features.

Using image processing techniques, Toh et al. [12] propose a simple approach for counting feeder fish automatically. First, they use blobs that mark the fish's location. The image of the blobs will be filtered to remove noise and background objects. Then, the number of fish in one frame and the average number of fish in the overall fish frame are counted using the blobs' area information. They were successful in their endeavors. The fish count accuracy escalates as the median reference area, averaging the median area for higher fish test cases. Unfortunately, as the number of intersecting fish increases, the accuracy drops.

In continuation to [12], Labuguen et al. [13] offer an automated fish fry counting method that uses image processing to detect the pixel area filled by each fish silhouette. First, a school of fish fry is placed in a specially built container that goes through binarization and edge detection. Then, to acquire the overall fish count and the average number of fishes for each image frame, they add the area inside each contour for each image frame.

Fabic et al. [14] described an effective method for fish detection, counting, and species categorization from underwater video recordings using blob counting and form analysis. They also used an erasure process to help with fish detection by subtracting the coral background and recovering fish shapes with canny edge detection. Zernike shape analysis was performed to detect shape similarity for fish species identification, either Acanthuridae or Scaridae. They used blob counting to calculate the fish count after the latter had been delimited. Finally, due to the rapid frame changes, they calculated the average fish count per unit time from the counts in each frame.

With similar motivation, Subdurally [15] determining the population density of public spaces has become critical for effective public space management in surveillance applications. Their two proposed methods using blobs and contour detection are based on the observation that heads are significantly more visible than other features, facilitating their differentiation. Individuals are counted in the proposed systems that produced extremely reliable results, with an average head detection rate of 82% and 84%, respectively. However, they claimed that selecting a good threshold can increase correctness by decreasing completeness. Other challenges are that skin pixel may vary from dark to bright pixels, and the threshold value needs further calibration if the focal length changes continuously.

2.4. Machine Learning-Based

Machine learning is an algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data become available.

Meesha et al. [16] used two machine learning strategies to develop wheat classification: support vector machine and neural network. They employed a digital camera to record wheat grain images, classified using image thresholding to extract morphological features using roundness ratio, area, and volume. Machine learning methods were used to extract wheat-specific information from the photos. Based on the results, it has an accuracy range of 86.8% to 94.5%. To categorize wheat into five grades, morphological features, on the other hand, necessitate extensive parameter tweaking. Applying an image processing algorithm with a support vector machine (SVM) classifier, Abozar et al. [17] employed a top view RGB camera to monitor the pigs, and a background subtraction approach was utilized to separate the animals from their background, taking into account the perimeter and area of the convex hull and boundaries. The proposed method automatically scores the lateral and sternal lying posture in grouped pigs under commercial farm conditions with high accuracy of 94.4% for the classification and 94% for the scoring (detection) phases using two-dimensional image features and classification techniques. The localization and

classification accuracy diminishes as a result of the comparable colors to the pen floor, lying close to the feeders or the pen wall.

2.5. Deep Learning-Based

Deep learning is an artificial intelligence function that mimics the work of the human brain in data processing and decision-making patterns. Deep learning is an artificial intelligence subset of machine learning that involves a network that can learn without supervision from unstructured or unlabeled data. Deep learning can also be referred to as deep neural learning or a deep neural network.

Sethy et al. [18] used a dataset of 5932 rice field images to study four different types of rice leaf diseases. They compared CNN models, deep features, and the SVM approach for identifying rice diseases using transfer learning. They discovered that using the SVM classification model to extract deep features from resnet50 outperforms other traditional feature methods such as local binary patterns, histograms of oriented gradients, and grey level co-occurrence matrix. A small CNN called mobilenetv2 plus SVM performs comparably to the resnet50 plus SVM.

Maryam et al. [19] presented a method for estimating yield that is based on a simulated deep convolutional neural network. Accounting for the exact number of fruits and flowers available can help farmers decide on planting practices and disease-resistant plants. Their network is entirely trained on synthetic data before being tested on real-world data. They used a modified Inception-ResNet architecture to capture features on multiple scales. Their algorithm can calculate the fruit precisely tomato estimation efficiently, even if there is an overlap between fruits. Their study results showed 91% average test accuracy on real images and 93% on synthetic images. They are yet to train on the unripe fruit, focusing only on ripe and half-ripe fruit.

On the other hand, Weilu et al. [20] developed a self-learning saliency feature map-based deep learning-based pipeline for pinpointing and calculating agricultural pests in photos. To reduce overlapping detections, the proposed method combines a ZF (Zeiler and Fergus model) convolutional neural network (CNN) and a region proposal network (RPN) with non-maximum suppression (NMS) on multi-scale images. Different feature extraction networks were investigated to showcase the practical uses of their technology, including AlexNet, ResNet, and ZF Net. The precision of their proposed approach was 0.93, with a miss rate of 0.10 and a mean average precision (mAP) of 0.885. ZF-Net outperforms the Alex-Net but is comparable to ResNet-50 and ResNet-101.

In continuation, Zhenglin et al. [21] proposed a method for on-tree mango fruit detection, tracing, and summing using 10-frames-per-second videos collected of trees from a moving base alongside the inter-row at 5 km per hour using deep learning, with the Kalman filter Hungarian algorithm. MangoYOLO, a deep learning-based mango fruit detection system, was employed in each frame to detect fruit. The Hungarian method correlated fruit between neighboring frames to enable multiple-to-one assignment. To hinder repetitive summation of a single fruit that is veiled or excluded identification in a frame series, they employed a Kalman filter to anticipate the position of fruit in the subsequent frames. The suggested method recorded 2050 fruits using MangoYOLO with a bias-corrected root mean square error (RMSE) of 18.0 fruit per tree, compared to 1322 fruit with a bias-corrected RMSE 21.7 fruit per tree using the dual-view picture method. Despite its lower implementation cost, this solution does not allow tree segmentation or localization within the orchard.

Recently, detecting marine creatures has also become favorable (X. Liu [22], Merencilla et al. [23], Li et al. [24]). Shark-EYE was developed by Merencilla [23] by using the YOLOv3 algorithm to detect single shark fish, including multi-scale prediction and bounding box prediction-based logistic regression. They achieved mAP values up to 86%. However, they revised their applications with wearable devices equipped with a wide range of cameras. Liu [22] collected the marine animal images with seven classes (abalone, crab, fish, lobster, scallop, sea cucumber, and shrimp) by an underwater robot equipped with an embedded

device. A MobileNetV2 model based on convolutional neural network (CNN) and transfer learning to alleviate insufficient data and motivate feature migration has the best validation set accuracy of 92.89%. However, this work focuses on classifying individual or close objects. Similar motivation by Li et al. [24] created the world's first large-scale Marine Animal Segmentation (MAS) dataset, MAS3K, using an ECD-Net-based MAS model. Their MAS3K collection contains several photos of marine animals with high-quality annotations, covering complicated underwater environments and the camouflage properties of marine animals. They asserted that ECD-Net is an excellent deep learning-based MAS model equipped with various interactive features and cascaded Decoder Module enhancements. Extensive trials showed that ECD-Net outperforms eleven SOTA object segmentation models in MAS performance.

We describe the comparative analysis for the counting object related works based on non-machine learning, machine learning, and deep learning in Table 1.

Table 1. Summary of the comparative analysis for the counting object-related works.

	Citation and Year	Key Features of Designed Algorithms/Models (Key Objectives and Performance Metrics)	Advantages (Achieved Performance)	Limitations (Based on the Application-Specific Standard Requirements)
Non-Machine Learning-Based Algorithms	Alomari et al. [2]	Cell counting using dynamic initialization and number of iterations	Reduce false positive rate, resolve easy and medium-density object image	Unresolved high-density object image
	Abdullah et al. [4]	Car counting using adaptive blob edge analysis	Resolve medium- and low-resolution images	Unsupported to vary low-contrast image type
	Jingyao et al. [11]	Color depth fusion algorithm using Kinect V2 Sensor	High true-positive crop segmentation rates	Low localization accuracy for flower-shaped vegetation
	Y.H. Toh et al. [12]	Count fish using blob and pixel size analysis methods	Accuracy escalates as the median reference area by averaging the median area of higher fish test cases	As the number of intersecting fish increases, accuracy drops
	R. T. Labuguen et al. [13]	Count fish fry using adaptive binarization and time frame average	High accuracy with intermediate density	Accuracy drops below 75% as the number of fry fish exceeds 700
	J.N. Fabic et al. [14]	Color histogram, canny edge and Zernike shape analysis to identify fish species (Acanthuridae and Scaridae) on underwater video fish sequences	Able to estimate close to the ground truth value	Overcount of less than 10% due to background elimination
	Subdurally et al. [15]	Calculate headcount using blob and contour analysis	The acceptable head detection rate	Low-value completeness factor for the contour method, skin pixel may vary from dark to bright pixels, and the threshold value effects if the focal length changes continuously.
Machine Learning-Based Algorithms	Meesha et al. [16]	Extract wheat grading into five classes using image thresholding, morphological features, support vector machine, and neural network	SVM outperforms NN with 86.8% and 94.5% accuracy rates	Morphological features are exhaustively relying on the pixel roundness ratio, volume, and area
	Abozar et al. [17]	Pig calculation and monitoring using support vector machine	Outperforms when counting lateral and sternal lying posture of gathered pigs	Open-floor, lying close to the feeders or pen wall distracts calculation performance because of alike colors

Table 1. *Cont.*

	Citation and Year	Key Features of Designed Algorithms/Models (Key Objectives and Performance Metrics)	Advantages (Achieved Performance)	Limitations (Based on the Application-Specific Standard Requirements)
Deep Learning-Based Algorithms	Sethy et al. [18]	Classifies four types of rice leaf diseases	Extracts deep features from resnet50 and mobilenetv2, typical and small CNN models, respectively, and classify them using the SVM classification model	
	Maryam et al. [19]	Tomato estimation using three parallel layers concatenated into one that improvised the Inception-ResNet-A module	The occurrence number of ripe and half-ripe fruits can easily accumulate	Ignore green fruit counting
	Weilu et al. [20]	Overcome overlapping detections for pest counting with the introduction of CNN with the Zeiler and Fergus (ZF) model and a region proposal network (RPN) with non-maximum suppression (NMS)	Multi-scale images can reduce the error losses and decrease false positives	ZF Net + RPN is comparable to ResNet-50 and ResNet-100
	Zhenglin et al. [21]	Mango plant counting using Kalman filter, Hungarian algorithm, and YOLO	LED lighting and a camera have low implementation costs because they exclude differential global navigation satellite system	Disregards localization within the orchard or tree segmentation
	Liu et al. [22]	MobileNetV2 model based on convolutional neural network (CNN) and transfer learning	Able to classify seven marine animals using a robot camera	Classification accuracy varies according to the number of individual animals in each captured image
	Merencilla et al. [23]	Shark EYE used YOLOv3 algorithm for object detection, multiscale prediction, and logistic regression-based bounding box prediction	The system uses a large collection of great white sharks' datasets underwater for training, as sharks are hard to differentiate from other shark-like animals in an underwater environment	It needs further to be refined for a wearable device equipped with a wide-angle camera
	Li et al. [24]	Segment marine animals involving conch, fish, and crab with a complex environment, MAS3K datasets	Introduce an enhanced cascade decoder network (ECDNet) with multiple interactive feature enhancement modules (IFEMs) and cascade decoder modules (CDMs)	The single decoder and the influence of the number of CDMs need to improvise for better performance

3. Mask R-CNN

Mask R-CNN [25] aims to solve the instance segmentation problem and separate objects in an image or a video. The output will give the object bounding boxes, classes, and masks. Mask R-CNN includes two stages: generating the proposals regions with an object given the input image or video in the first stage. The second stage covers a pipeline that can anticipate the object class label, uncover the bounding box, and create an object mask at the pixel level specified in the first stage proposal region. In a nutshell, the Mask R-CNN backbone structure is subsequently associated with both stages.

Instance segmentation is an extension of object detection, in which a binary mask is associated with each bounding box. This instance segmentation allows for more refined information about the extent of the object in the bounding box. Several algorithms perform segmentation, but the one used by the Tensorflow Object Detection API is the Mask R-

CNN. Mask R-CNN detects objects through a series of steps. Firstly, the default parameter remains, and a CNN feature descriptor extracts a global feature image. Second, apply to these feature maps using the region proposal network. These steps produce ROIs representing the object proposals accompanied by their objectness scores. The ROIs and corresponding feature maps are then passed through the Region of Interest (RoI) Align layer. Mask R-CNN employs the ROI Align layer rather than the ROI Pooling layer. The RoI Align layer is intended to fix the location misalignment in the ROI pooling affected by quantization. An ROI is appropriately mapped from the actual image to the feature map without rounding up to integers. In the next step, a softmax layer is used on top of the fully connected network to predict classes. A linear regression layer is used parallelly to output bounding box coordinates for predicted classes and the softmax layer.

Faster R-CNN [26] is also a unique algorithm used for object detection. Similarly, the Faster R-CNN consists of two phases. The first phase, known as the regional proposed network (RPN), recommends a bounding box only for nominees with constrained objects. In the second stage, after extracting features from each bounding box via Region of Interest Pooling (RoIPool), Faster R-CNN executes subsequent processes involving the classification and regression for each bounding box. For a faster conclusion, the features used by both processes can be exchanged. The concept of Mask R-CNN [25] is straightforward: the Faster R-CNN mask has two outputs for each nominee object, a class label, and a bounding-box offset, and extends a third branch that generates the object mask in binary form to represent the pixels in the bounding box where the object is. The output of the extension mask is dissimilar from the output of the class. In contrast, the bounding box indulges a more acceptable spatial arrangement of the object during the extraction process.

We measure the loss function, L_{total}, for each shrimp RoI as follows:

$$\sum L_{total} = (\, w_1 \times L_{rc}\,) + (\, w_2 \times L_{rb}\,) + (\, w_3 \times L_c\,) + (w_4 \times L_b) + (w_5 \times L_m\,) \ldots . \quad (1)$$

where w is a set of weights of each loss function, L_{rc} is the region proposal network (rpn) class loss, L_{rb} is the shrimp rpn bounding box loss, L_c is the shrimp class loss, L_b is the shrimp bounding box loss, and L_m is the shrimp mask loss. It calculates L_{total} after changing a set of hyperparameters in Mask R-CNN such as maximum detection instance, maximum ground truth instance, number of thresholds, train anchors for each image, number of steps for each epoch, number of train regions of interest of each image, number of validation steps, number of steps in each epoch, and numbers of epoch, regularization, optimizers, learning rate, batch size, learning momentum, and weight decay. Undertaking the most minimum loss, we achieved calibration of those parameters manually. Hence, Figure 3 describes the improved Mask R-CNN after the parameter had changed based on the default Mask R-CNN. The steps to detect objects in the improved Mask R-CNN model will remain the same.

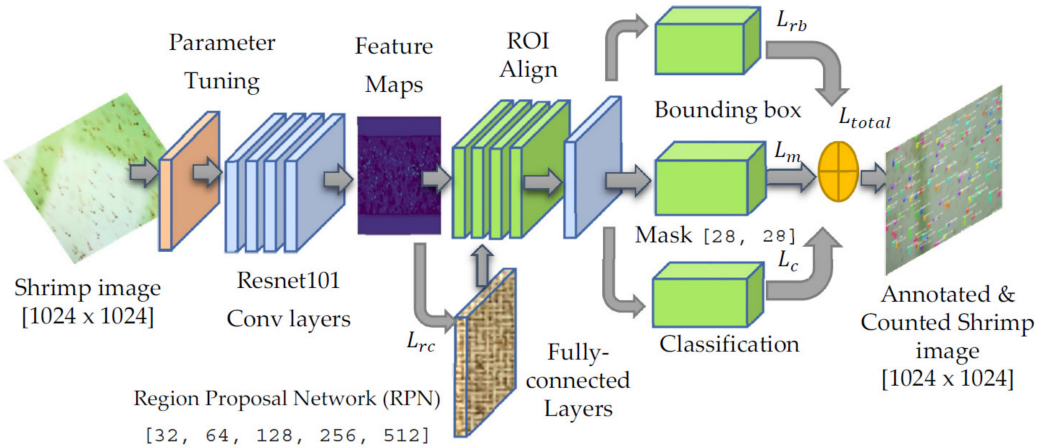


Figure 3. Algorithm for improved Mask R-CNN with parameters calibration step.

4. Experimental Results and Analysis

To save time in the training models' process and shorten the time in the labeling dataset, Mask R-CNN is used in this paper to find out the best parameter and detect the total number of shrimps in an image. We designed and sketched the experimental setup for the shrimp counting system, as shown in Figure 4, encompassing the container, box, sensor, and touch light.

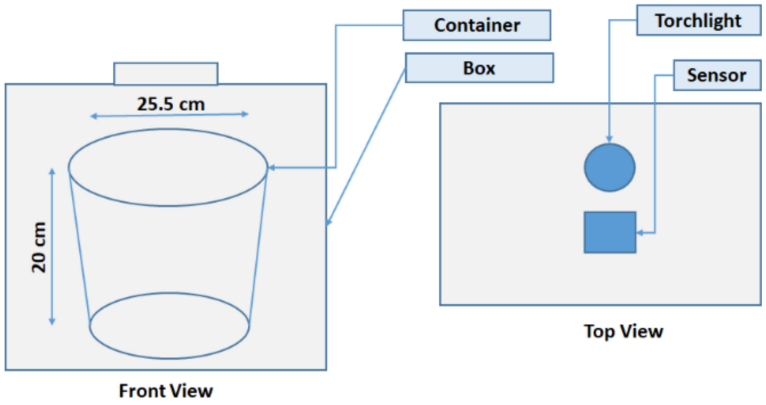


Figure 4. The experimental design to set up the shrimp counting system.

The sensor used in this experiment was a camera with a model Arducam USB camera module 8MP Sony IMX 219. The other types of equipment that were included a container, box, and torchlight. The pixels for capturing the photo were set to 640 pixels for width and 480 pixels for height. The image taken was saved in .jpg format. The height of the container was 20 cm, and the diameter was 25.5 cm. The torchlight was adjusted until all the light could cover the container's water surface. The actual view of the experimental setup for our shrimp counting system is depicted in Figure 5.

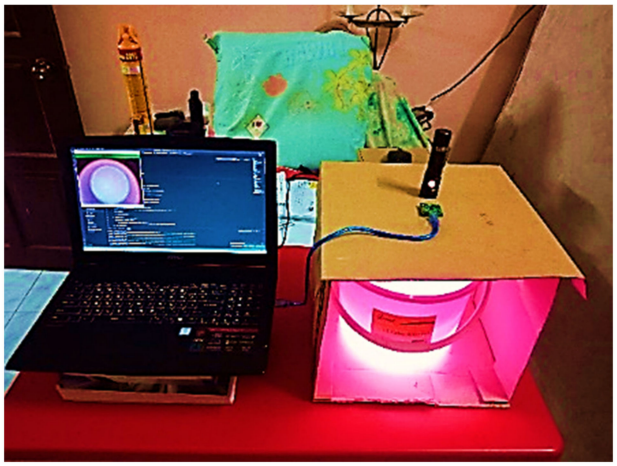
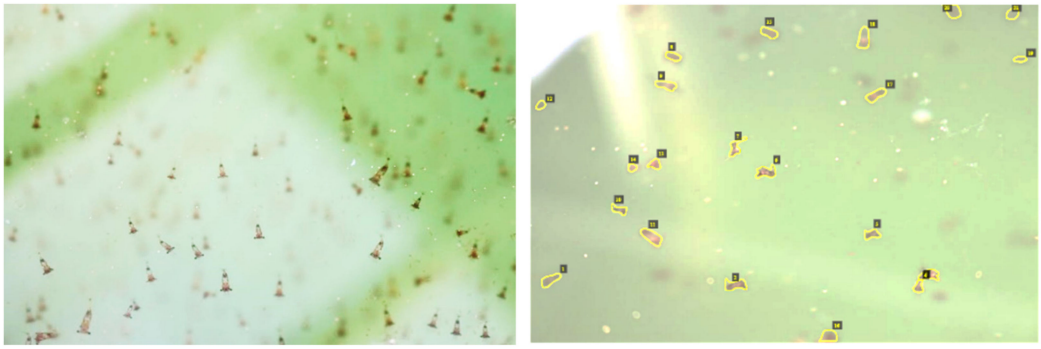


Figure 5. The real view of the experimental setup for the shrimp counting system.

4.1. Building a Dataset

The dataset used for this paper is a picture that consists of a troupe of shrimps. The picture was collected and captured using a Canon EOS 80D model DSLR camera. The aperture was f/5.6, the shutter speed was 1 per 100 s, ISO 100, the camera lens used was 18–55 mm, and the distance to take pictures was 10 cm from the top of the water level. The picture captured is shown in Figure 6a. The picture was collected with a total number of 120 images. With envision, an 80:20 supplied train and test dataset split concept was applied, consisting of 100 images for the training set and 20 images for the testing dataset.



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

Figure 6. (**a**) Underwater image detection for shrimp, and (**b**) example of a labeled picture.

Underwater cameras obtained some of the data used in the experiment for marine animals and others. The data are divided into seven categories: fish, shrimp, scallop, crab, lobster, abalone, and sea cucumber. Each category ranges from 1000 to 1400 sheets, with a total of 8455 sheets where 80% of data were used for training and 20% for testing sets. We enhanced the training set data; moreover, each original image was generated into three deformed images by three processing methods: rotation, translation, and flipping. The training set can be expanded to 27,056 pictures.

The image taken changed the image size from 6000 pixels for width and 4000 pixels for height to 960 pixels for width and 640 pixels for height to facilitate the labeling of the shrimp in the pictures. The shrimp in the picture was labeled using the VGG Image Annotator, and the labeled picture is shown in Figure 6b.

The folder of "datasets" was created in Google Drive, along with two subfolders named "train" and "test" to store the training dataset and testing dataset. After the labeled picture, the annotation information with the .json file with the same name was saved in the corresponding subfolders "train" and "test". Less dense category ranges totaled 81 images, medium dense category ranges totaled 15 images, and dense category ranges had four images. After labeling the shrimp in each image, it generated 5041 ground truth in training the model phase.

4.2. Training the Model

We chose ResNet101 in combination with FPNs from the Mask R-CNN backbone networks. The feature map was extracted from the input image by the backbone network first, and then the features were output by the backbone network. The map was given to the region proposal network (RPN) and ROIAlign (ROI) to generate the area of interest. Finally, the ROI used the convolutional and fully connected layers to forecast the target category and bounding box and the fully convolutional neural network to recognize the target region (FCN). The target's instance detection task has finally been accomplished.

The training model procedure in this paper uses 100 training images and the default and improved hyperparameters of the Mask R-CNN model, as shown in Table 2 and Appendix A. The default number of steps for each epoch is 50, but the upgraded version calls for roughly 100. In the training phase, using the above model requires a significant amount of memory resources and time. As a result, Google Collab is used in the training model process. Figure 3 depicts the method of determining the best hyperparameters.

Regularization L1 and optimizer SGD with the lowest loss function is taken to the next step, converting the learning rate. Referring to Table 2 and Figure 7, the values for the learning rate to be tested are 0.01, 0.001, and 0.0001; maintaining the lower learning rate with the lowest loss function will be maintained and taken to the next step, which is a conversion on the learning momentum. The values for learning momentum to be tested are 0, 0.5, and 0.9. The value of learning momentum with the lowest loss function will also be maintained and taken to the next step to convert the weight decay. The values for weight decay to be tested are 0.1, 0.01, and 0.001.

Table 2. Experimental parameters.

Parameter	Default Mask R-CNN	Improved Mask R-CNN
Regularization	L2	L1
Maximum Detection instance	100	400
Learning rate	0.001	0.01
Maximum ground truth instance	100	400
Name	NONE	SHRIMP
RPN threshold value	0.7	0.8
RPN train anchors per image	256	512
Number of Steps per epoch	50	100
Train Region of Interest per image	200	300
Validation steps	50	200
Weight decay	0.0001	0.001

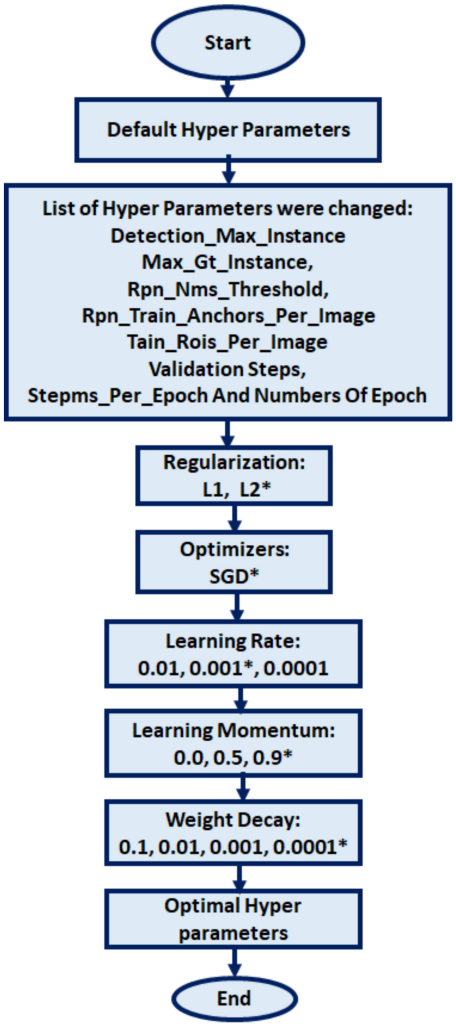


Figure 7. The process of finding the optimal hyperparameter.

In the last step, the optimal hyperparameters are selected based on their performance, and the model is used in the next phase, which is the implementation phase or the testing phase. The performance of the model is then tested in terms of accuracy.

4.3. Experimental Environment

In Table 3, we declare our experimental environment information, where we use Tensorflow version 1.3.0 [27,28] and Keras version 2.0.8 to extract important MR-CNN

libraries for the software part. We ran the code on a computer with 8GB RAM, and i-7 Processor with a CPU clock speed of 2.8 GHz, and a GPU Geforce GTX1050 running Windows 10, 64 bit.

Table 3. Experimental environmental information.

Attribute Name	Attribute Value
Tensorflow version	1.3.0
Keras Version	2.0.8
RAM	8 GB
Processor	Intel (R) Core TM i7-7700HQ CPU @ 2.80GHz
Graphics	GeForce GTX 1050
Operating system version	Windows 10 Pro, 64 bit

4.4. Evaluation Index

The evaluation index for the performance of the model is evaluated based on precisions, recall, mean average precision (mAP), accuracy based on category, and value of R^2. With 20 images as the validation set, the validation results of the improved method are compared with those of other methods. Several symbols are used in the equations, and Table 4 shows the notation table.

Table 4. List of notations.

Symbol	Meaning
TP	The number of images consisting of shrimp has been correctly localized
FP	The number of images unsuccessfully or partially localize the shrimp
FN	The number of images unsuccessfully localize the shrimp
$\sum_{q=1}^{Q} AveP(q)$	The variable q is the number of queries in the set, and $AveP(q)$ is the average accuracy average precision for a particular query
Q	A particular query
$\int_0^1 p(r)\,dr$	Mean average precision by using all the point interpolation of precision and recall
n	Number of images in the training dataset
$\sum$	Summation
x	Predicted number of shrimps
y	Number of ground truths

The test set obtains the precision following Equation (2) and the recall following Equation (3). This index is used to measure and evaluate the effect of the model on shrimps positioning. Precision is defined as the fraction of relevant instances among all retrieved instances. Recall, sometimes referred to as "sensitivity", is the fraction of retrieved samples among all appropriate instances. A perfect model or classifier has precision and recalls equal to 1.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

where TP is the number of images that correctly localize the shrimp, FP is the number of images that unsuccessfully or partially localize the shrimp, and FN is the number of images that unsuccessfully localize the shrimp.

Equation (4) shows how the mAP is calculated by q, where q is the number of queries in the set and $AveP(q)$ is the average accuracy average precision for a particular query, Q.

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{4}$$

Before testing the mAP, the value of intersection over union (IoU) needs to be specified. IoU is an intersection area between two boundaries (real object boundaries and object boundaries predicted). Equation (5) shows how IoU is calculated, and an example of IoU in the picture is shown in Figure 8.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{5}$$

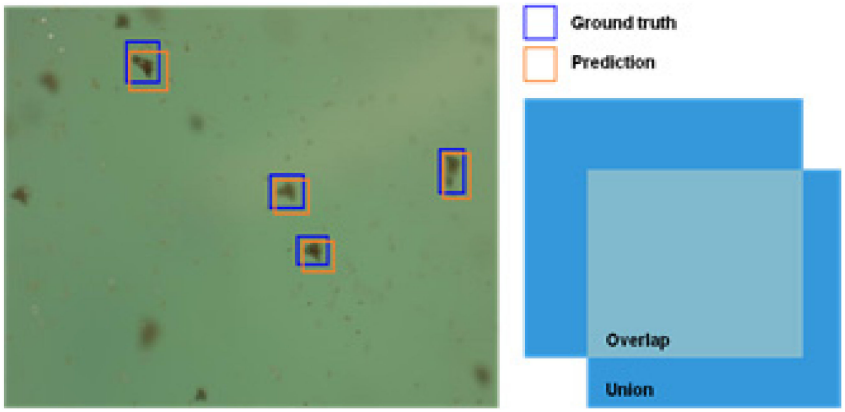


Figure 8. Example of calculated IoU.

By using the equation shown in Equation (3), the value of IoU determined in this experiment is 0.50. If the value of IoU is greater than or equal to 0.50, the object will be classified as TP, which is true positive; if the value of IoU is less than 0.5, the object will be classified as false negative (FN). Equation (6) shows how to calculate AP (average precision) with precision and recall. The value of IoU will be tested on 0.50 and 0.75.

$$AP = \int_0^1 p(r)\, dr \tag{6}$$

To calculate the accuracy based on the category is a comparison between the actual number (ground truth) and the number predicted based on the training dataset. Equation (7) shows how to calculate the accuracy rate, and Equation (8) shows how to calculate the error rate.

$$Accuracy\ Rate = \frac{No.of\ Predicted}{No.of\ Ground\ Truth}\ x\ 100\% \tag{7}$$

$$Error\ Rate = 100\% - Accuracy\ Rate \tag{8}$$

The density of the number of shrimps is divided into three categories: less dense, medium dense, and highly dense. The maximum number of actual numbers is 256 shrimps, and the minimum number is four shrimps. For the less dense category, the ground truth is between 1 to 90 shrimps, consisting of 82 images, and one of the pictures is shown in Figure 9 (left). For the category of medium dense, the ground truth is between 91 to 180 shrimps, consisting of 15 images, and one of the pictures is shown in Figure 9 (middle). For the high dense category, the ground truth is between 181 to 270 shrimps, consisting of three images and one of the pictures shown in Figure 9 (right).

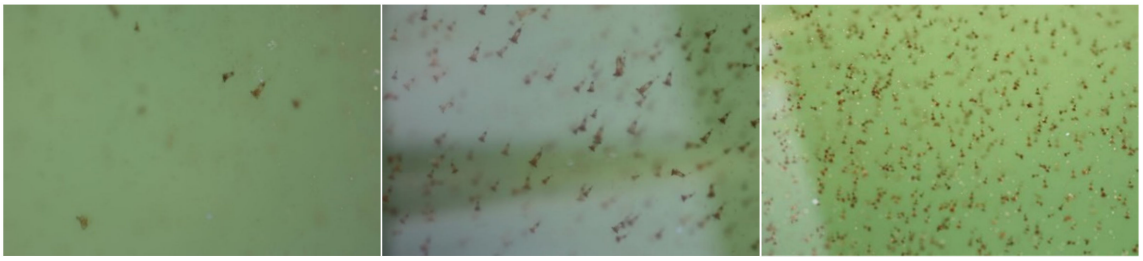


Figure 9. (**left**) Less dense category, (**middle**) medium dense category, and (**right**) high dense category.

The value of R^2 is the comparison of results between the actual number of shrimps and the predicted number of shrimps. The method performed on the actual number with the predicted number is using linear regression. In comparing the actual number and the predicted number of shrimps, the value of R^2 is evaluated. From the value of R^2, we know whether the regression line corresponds to the data used or not. Equation (9) shows how to calculate the value of R^2.

$$R^2 = \left[\frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{\left[n\sum x^2 - (\sum x)^2\right]\left[n\sum y^2 - (\sum y)^2\right]}} \right]^2, \quad (9)$$

where x is the predicted number of shrimps in each of the images and y is the actual number (ground truth). The value of R^2 is always between 0% and 100%. The value of R^2 of 0% represents a model that does not explain any variations in the response variable around its mean. The mean of the dependent variable predicts the dependent variable and the regression model. The value of R^2 of 100% represents a model that explains all the variations in the response variable around its mean. Usually, the more significant the R^2 value, the better the regression model fits the observations.

4.5. Experimental Results and Analysis

We studied the performance of the proposed improved Mask R-CNN model and compared it with the existing Mask R-CNN model using the shrimp datasets. Figure 10 shows the precision and recall results for the training dataset, and Figure 11 shows the accuracy and identification of the validation dataset. It can be seen from Figures 10 and 11 that the improved Mask R-CNN model has a significant improvement in precision and recall by comparing the Mask R-CNN model.

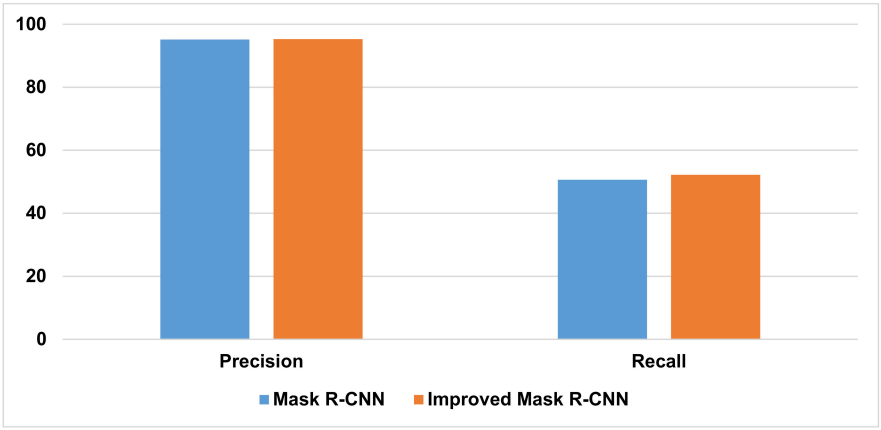


Figure 10. Results of precision and recall for the training dataset for default and improved Mask R-CNN.

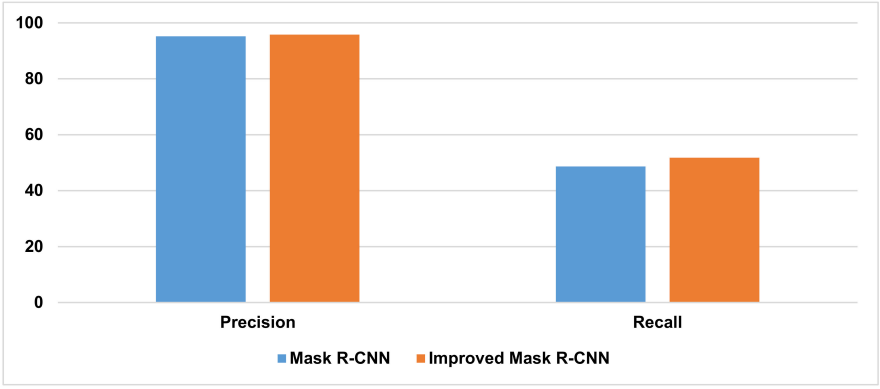


Figure 11. Results of precision and recall for the testing dataset for default and improved Mask R-CNN.

Table 5 displays the results of precision and recall from this experiment. The precision in the training dataset increased from 95.18% to 95.79%. For the recall in the training phase, the dataset significantly increased to 51.77%, whereas the precision in the validation dataset slightly increased to 95.30%. For the recall in the validation, the dataset increased from 50.63% to 52.20%.

Table 5. Results of precision and recall on the training and testing datasets.

	Train		**Test**	
	Precision	**Recall**	**Precision**	**Recall**
Mask R-CNN	95.18%	48.65%	95.15%	50.63%
Improved Mask R-CNN	95.79%	51.77%	95.30%	52.20%

As can be seen from Table 6, the AP value 0.50 of the improved Mask R-CNN in the training dataset is 99%, which is 8.77% higher than the existing detection model Mask R-CNN. The AP value 0.75 of the improved Mask R-CNN is 96.35%, which is 30.5% higher than the existing detection model Mask R-CNN. On the other hand, the improved Mask R-CNN has an AP value of 0.50 in the validation dataset, which is 3.87% higher than the existing detection model Mask R-CNN, as shown in Table 6. The improved Mask R-CNN has an AP value of 0.75, which is 25.73% higher than the existing Mask R-CNN detection model.

Table 6. Results of mean average precision.

	Train		**Test**	
	mAP ($AP_{0.50}$)	**$AP_{0.75}$**	**mAP ($AP_{0.50}$)**	**$AP_{0.75}$**
Mask R-CNN	90.23%	65.85%	95.83%	72.77%
Improved Mask R-CNN	99.00%	96.35%	99.70%	98.50%

Performance measurement of mAP considers the value of 0.50 and 0.75 for the training dataset shown in Figure 12. The proposed improved Mask R-CNN has shown significant performance compared to the existing Mask R-CNN (Figures 12 and 13) using the validates the dataset shown in Figure 14.

Notably, the accuracy drops as the density increases, as tabulated in Table 7. It also suggests that in the less dense category with the number of ground truths of 2682, the

proposed model can obtain the predicted number of shrimps of 2671 and achieve the value for an accuracy rate of 99.59% and an error rate of 0.41%.

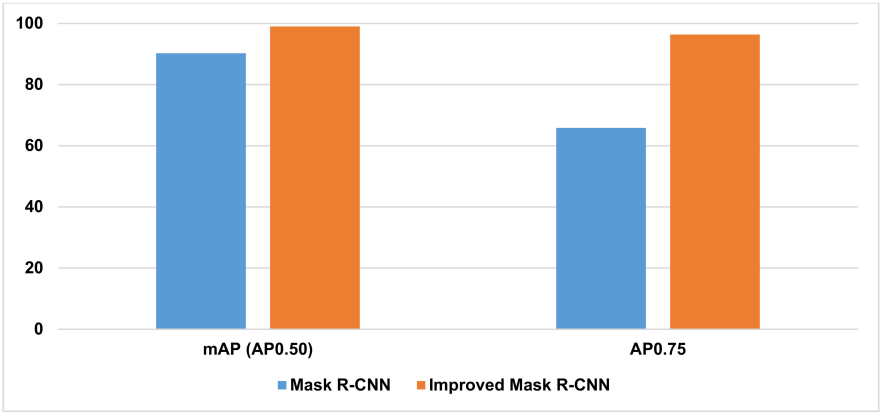


Figure 12. Results of mAP when AP = 0.50 and AP 0.75 for the training dataset for default and improved Mask R-CNN.

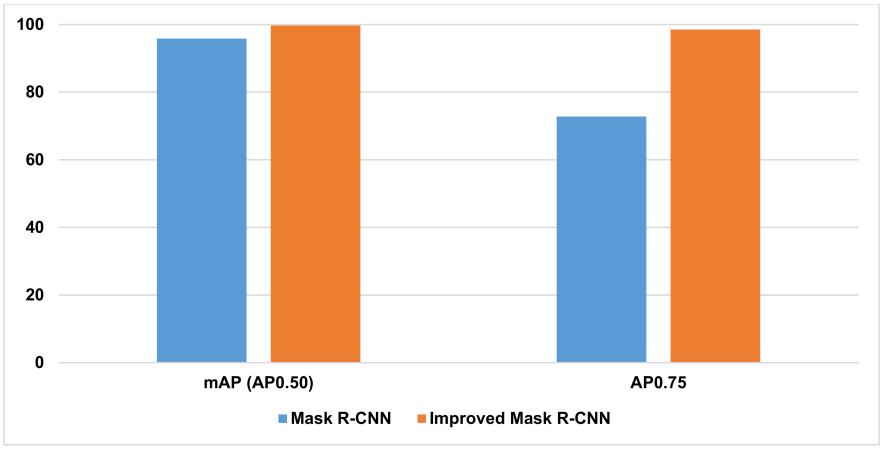


Figure 13. Results of mAP when AP = 0.50 and AP = 0.75 for the testing dataset after applying default and improved Mask R-CNN.

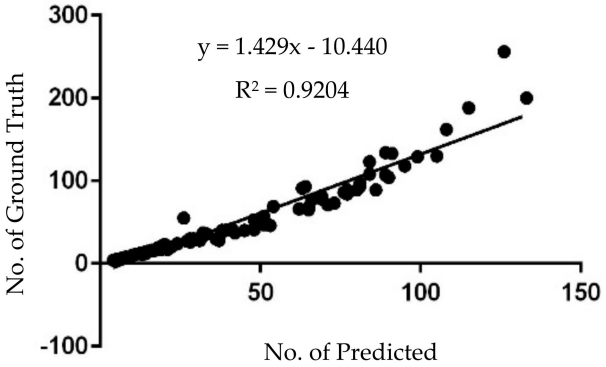


Figure 14. Linear regression between the number of ground truths and the predicted number of shrimps for 100 images in the training dataset for the default Mask R-CNN.

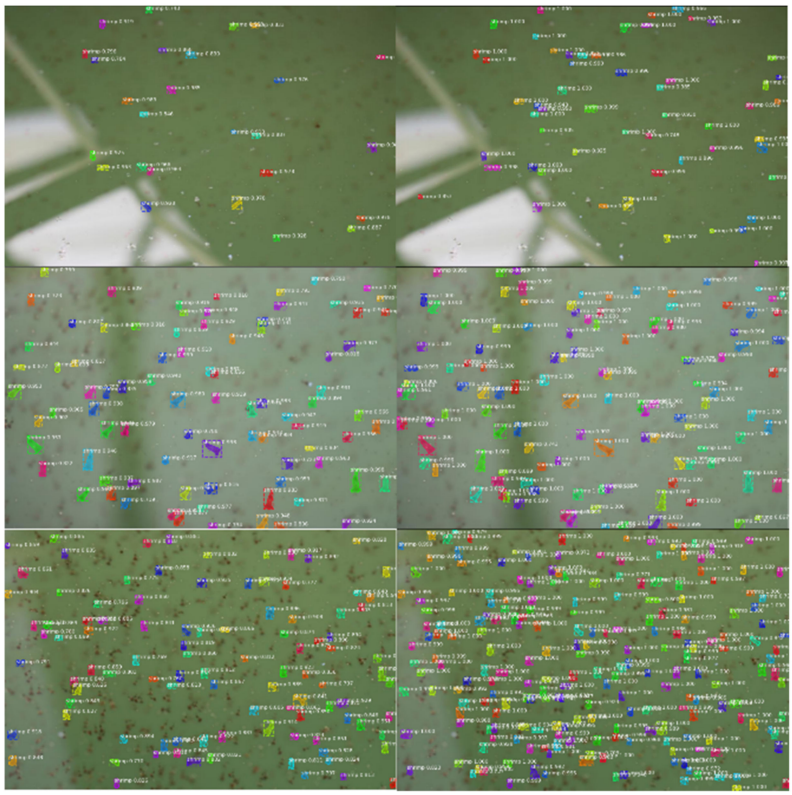
In the medium dense category with the ground truth number of 1715, the proposed model can achieve the predicted number of 1679 shrimps, 97.90% accuracy, and 2.10% error rate. Meanwhile, if the number of ground truths is 644, the proposed model still predicts 564 shrimps and 87.58% accuracy, and a 12.42% error rate. Therefore, the analysis

recommends that the overall accuracy rate for the proposed model on the training dataset reached 97.48%, which is 4914 shrimps out of 5041 shrimps.

Table 7. Results of accuracy rate and error rate based on category.

Category	No. of Ground Truths	No. of Predicted Shrimps	Accuracy Rate	Error Rate
Less dense	2682	2671	99.59%	0.41%
Medium dense	1715	1679	97.90%	2.10%
Dense	644	564	87.58%	12.42%
Total	5041	4914	97.48%	2.52%

Figure 14 shows the result of the pictures based on each category using the (i) default Mask R-CNN model and (ii) the improved Mask R-CNN model. Figure 15 depicts three shrimp images by density category, where the (a) less dense category has predicted 26 shrimps (GT = 26) in (a) and 55 (GT = 55) in (b); the (b) medium dense category has predicted about 83 shrimps (GT = 84) in (a) and 97 (GT = 99) in (b); finally, the (c) high dense category has predicted 100 shrimps (GT = 104) in (a) and 213 (GT = 256) in (b).



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

Figure 15. Comparison of the results of shrimps by using the (**a**) default Mask R-CNN model and (**b**) improved Mask R-CNN model. The images of shrimp by density: row 1, less dense with the predicted number of shrimps of 26 (GT = 26) in (**a**) and 55 (GT = 55) in (**b**); row 2, medium dense with the predicted number of shrimps of 83 (GT = 84) in (**a**) and 97 (GT = 99) in (b); row 3, high dense with the predicted number of shrimps of 100 (GT = 104) in (**a**) and 213 (GT = 256) in (**b**).

One of the ideas in counting is to calculate the object indirectly by estimating the density map. The first step is to prepare a practice sample dataset with an appropriate density map for each picture. Density maps are created by performing a convolution with a Gaussian kernel and normalizing it so that integrating it yields the number of objects.

The main objective is to train the convolutional network to plot an image to a density map that can accumulate the number of object occurrences. The density maps according to the shrimp density are shown in Figure 16.

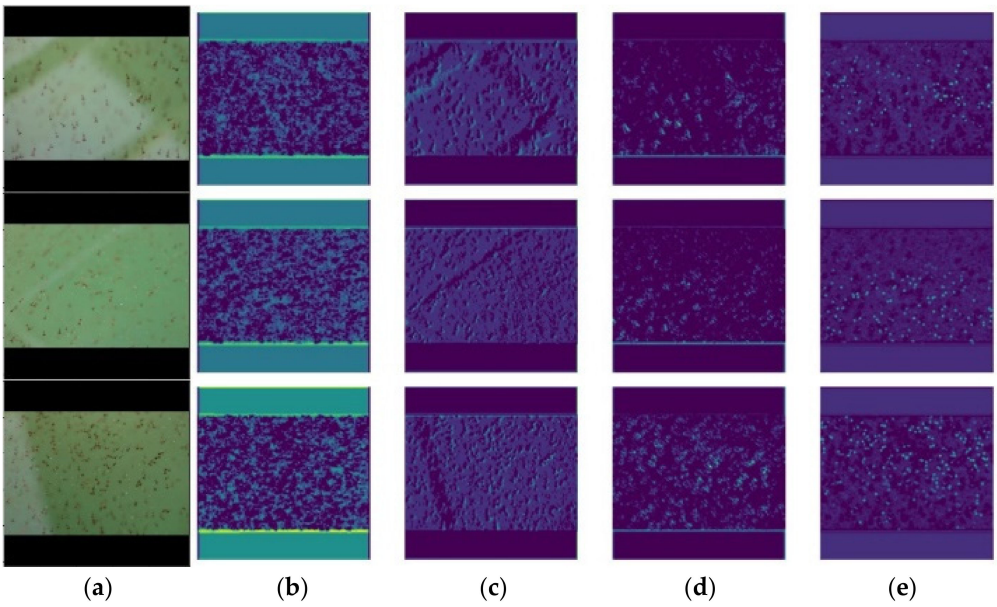


Figure 16. (**a**) Image of shrimp by density: row 1, less dense with the actual number of shrimps of 46 and predicted number of shrimps of 46; row 2, medium dense with the actual number of 118 and predicted number of 118; row 3, dense with the actual number of 188 and predicted a number of 170. (**b**) Layer of Resnet101 res2c_out, (**c**) layer of Resnet101 res3c_out, (**d**) layer of Resnet101 res4w_out and (**e**) predicted number of shrimps.

Linear regression for the existing Mask R-CNN model and improved Mask R-CNN was performed between the actual and the predicted number of shrimps. Linear regression for the existing Mask R-CNN model for the detection is shown in Figure 17. The value of R^2 of 0.9204 in Figure 14 suggests that the regression line does not fit well over the data, which means the predicted number of shrimps is not similar to the actual number of shrimps.

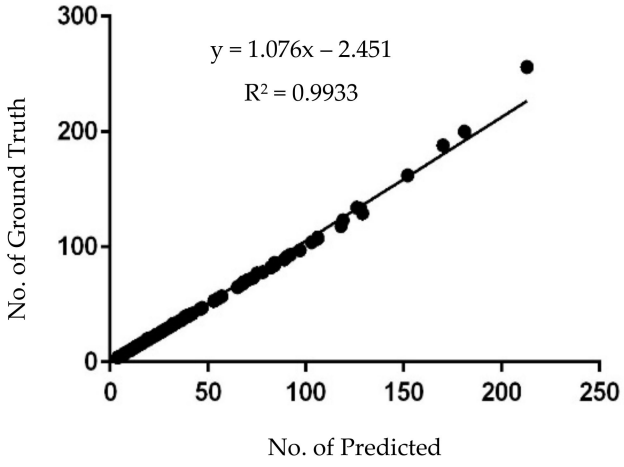


Figure 17. Linear regression between the number of ground truths and the predicted number of shrimps for 100 images in the training dataset for the improved Mask R-CNN.

Linear regression for the improved Mask R-CNN model is shown in Figure 17. The value of R^2 of 0.9933 in Figure 17 suggests that the regression line fits nicely over the

data, which means the predicted number of the shrimps is similar to the actual number of shrimps.

Despite these shortcomings, no studies have investigated shrimp counting using computer vision with deep learning. Hence, this work offers several significant contributions:

i The shrimp images were recorded from the top view with the assumption of equal size due to similar shrimp age kept in the container.
ii It can automatically estimate the number of shrimps using computer vision and deep learning.
iii Default Mask R-CNN can be manipulated to effectively segment and count tiny shrimps or objects.
iv The shrimp counting accuracy depreciates as the shrimp density increases or intensifies.
v The shrimp estimation efficacy has a linear proportion when the hyperparameters such as maximum detection instance, learning rate, maximum ground truth instance, RPN threshold value, RPN train anchors per image, the number of steps per epoch, train region of interest per image, validation steps, and weight decay are increasing.
vi The linear regression shows that R^2 increases with better precision after performing hyperparameter manipulation over the default Mask R-CNN.
vii This application can reduce shrimp death risk compared to practicing manual counting.

Moreover, this work can be extended using IoT [29–31] with explainable future communication technologies for further data collection and processing, and further validation will be carried out on a large scale of implementation.

5. Conclusions

This study successfully developed a detection and recognition model based on a deep learning-based Mask R-CNN model for underwater shrimp counting. After testing and improvement, the proposed method improved the mAP, precision, and recall. The critical parameters that influence this advancement for the proposed method are maximum detection instance, maximum ground truth instance, number of thresholds, train anchors for each image, number of steps for each epoch, number of train regions of interest of each image, number of validation steps, number of steps in each epoch, and numbers of epochs, regularization, optimizers, learning rate, batch size, learning momentum, and weight decay. After intensive manual changing of these parameters, we identified the best Mask R-CNN parameters. The backbone used in this paper is ResNet-101, either with default or parameter tuning to verify the precision and efficiency of shrimp recognition. The training dataset and validation dataset results show that the improved Mask R-CNN model can detect and locate the shrimp accurately with a value of 97.48% compared to the existing method, which is more accurate than existing methods. The contribution of this study is evident as the resulting outcomes can be capitalized as guidelines for the agriculture industry when estimating the number of underwater animals using computer vision versus manual counting. In fact, the current study contributes to our underwater computer vision knowledge by addressing three critical issues: reducing underwater animal death risk despite manual counting, Mask R-CNN configuration, and highlighting the pitfalls and advantages in terms of efficacy when dealing with different densities of small animals.

Author Contributions: Conceptualization, T.H.K. and S.N.H.S.A.; Formal analysis, T.H.K., S.N.H.S.A. and M.K.H.; Funding acquisition, S.N.H.S.A.; Investigation, T.H.K.; Methodology, T.H.K., S.N.H.S.A. and M.K.H.; Resources, S.N.H.S.A. and Ahmad Tarmizi; Software, T.H.K. and M.K.H.; Supervision, S.N.H.S.A.; Validation, S.N.H.S.A.; Visualization, M.K.H. and A.T.; Writing—original draft, T.H.K.; Writing—review & editing, S.N.H.S.A. and M.K.H. All authors have read and agreed to the published version of the manuscript.

Funding: The Ministry of Higher Education Malaysia supported this research via a research grant award (no. FRGS/1/2019/ICT02/UKM/02/9).

Data Availability Statement: All data has been supplied with the manuscript.

Acknowledgments: Authors would like to thank the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, for supporting this research and the Ministry of Higher Education, Malaysia. The authors also extend theirs acknowledges to the industry collaborators for the technical setup and environment assistance.

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

Appendix A

Hyperparameter settings for the proposed Mask R-CNN pipeline calibration.

BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	400
DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	1
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	1024
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	800
IMAGE_MIN_SCALE	0
IMAGE_RESIZE_MODE	square
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.01
LOSS_WEIGHTS	{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	400
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(56, 56)
NAME	shrimp
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.8
RPN_TRAIN_ANCHORS_PER_IMAGE	512
STEPS_PER_EPOCH	100
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	False
TRAIN_ROIS_PER_IMAGE	300
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	200
WEIGHT_DEC	0.001

References

1. Ghazvini, A.; Abdullah, S.N.H.S.; Ayob, M. A Recent Trend in Individual Counting Approach Using Deep Network. *Int. J. Interact. Multimed. Artif. Intell.* **2019**, *5*, 7–14. [CrossRef]
2. Abdullah, S.N.H.S.; Zaini, A.S.; Yilmaz, B.; Abdullah, A.; Othman, N.S.M.; Kok, V.J. Contour Based Tracking for Driveway Entrance Counting System. *Int. J. Integr. Eng.* **2019**, *11*, 1–10. [CrossRef]
3. Abu-Ain, T.A.K.; Abdullah, S.N.H.S.; Omar, K.; Abd Rahman, S.Z. Advanced Stroke Labeling Technique Based on Directions Features for Arabic Character Segmentation. *Asia-Pac. J. Inf. Technol. Multimed.* **2019**, *8*, 97–127. [CrossRef]
4. Abdullah, S.N.H.S.; Abdullah, S.; Petrou, M.; Abdullah, S.; Fitriayanshah, A.; Razalan, M.H.A. A portable rice disease diagnosis tool based on bi-level color image thresholding. *Appl. Eng. Agric.* **2016**, *32*, 295–310. [CrossRef]
5. Alomari, Y.M.; Abdullah, S.N.H.S.; Zin, R.R.M.; Omar, K. Iterative Randomized Irregular Circular Algorithm for Proliferation Rate Estimation in Brain Tumor Ki-67 Histology Images. *Expert Syst. Appl.* **2015**, *48*, 111–129. [CrossRef]
6. Solahudin, M.; Slamet, W.; Dwi, A.S. Vaname (*Litopenaeus vannamei*) Shrimp Fry Counting Based on Image Processing Method. *IOP Conf. Ser. Earth Environ. Sci.* **2018**, *147*, 012014. [CrossRef]
7. Kumar, G.; Bhatia, P.K. A Detailed Review of Feature Extraction in Image Processing Systems. In Proceedings of the 2014 Fourth International Conference on Advanced Computing & Communication Technologies, Rohtak, India, 8–9 February 2014. [CrossRef]
8. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [CrossRef]
9. Ruiz, L.A.; Recio, J.A.; Fernández-Sarría, A.; Hermosilla, T. A feature extraction software tool for agricultural object-based image analysis. *Comput. Electron. Agric.* **2011**, *76*, 284–296. [CrossRef]
10. Khurana, U.; Turaga, D.; Samulowitz, H.; Parthasrathy, S. Cognito: Automated Feature Engineering for Supervised Learning. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 1304–1307. [CrossRef]
11. Gai, J.; Tang, L.; Steward, B.L. Automated crop plant detection based on the fusion of color and depth images for robotic weed control. *J. Field Robot.* **2019**, *37*, 35–52. [CrossRef]
12. Toh, Y.H.; Ng, T.M.; Liew, B.K. Automated Fish Counting Using Image Processing. In Proceedings of the 2009 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 11–13 December 2009. [CrossRef]
13. Labuguen, R.T.; Volante, E.J.P.; Causo, A.; Bayot, R.; Peren, G.; Macaraig, R.M.; Libatique, N.J.C.; Tangonan, G.L. Automated fish fry counting and schooling behavior analysis using computer vision. In Proceedings of the 2012 IEEE 8th International Colloquium on Signal Processing and Its Applications (CSPA 2012), Malacca, Malaysia, 23–25 March 2012. [CrossRef]
14. Fabic, J.N.; Turla, I.E.; Capacillo, J.A.; David, L.T.; Naval, P.C. Fish population estimation and species classification from underwater video sequences using blob counting and shape analysis. In Proceedings of the 2013 IEEE International Underwater Technology Symposium (UT 2013), Tokyo, Japan, 5–8 March 2013; pp. 1–6. [CrossRef]
15. Subdurally, S.; Dya, D.; Pudaruth, S. Counting People Using Blobs and Contours. *Int. J. Comput. Vis. Image Processing* **2015**, *3*, 1–16. [CrossRef]
16. Punn, M.; Bhalla, N. Classification of Wheat Grains Using Machine Algorithms. *Int. J. Sci. Res.* **2013**, *2*, 363–366.
17. Nasirahmadi, A.; Sturm, B.; Olsson, A.C.; Jeppsson, K.H.; Müller, S.; Edwards, S.; Hensel, O. Automatic scoring of lateral and sternal lying posture in grouped pigs using image processing and Support Vector Machine. *Comput. Electron. Agric.* **2019**, *156*, 475–481. [CrossRef]
18. Sethy, P.K.; Barpanda, N.K.; Rath, A.K.; Behera, S.K. Deep feature-based rice leaf disease identification using support vector machine. *Comput. Electron. Agric.* **2020**, *175*, 105527. Available online: https://www.sciencedirect.com/science/article/abs/pii/S0168169919326997 (accessed on 29 November 2021). [CrossRef]
19. Rahnemoonfar, M.; Sheppard, C. Deep Count: Fruit Counting Based on Deep Simulated Learning. *Sensors* **2017**, *17*, 905. [CrossRef] [PubMed]
20. Li, W.; Chen, P.; Wang, B.; Xie, C. Automatic Localization and Count of Agricultural Crop Pests Based on an Improved Deep Learning Pipeline. *Sci. Rep.* **2019**, *9*, 7024. [CrossRef] [PubMed]
21. Wang, Z.; Walsh, K.; Koirala, A. Mango Fruit Load Estimation Using a Video Based MangoYOLO—Kalman Filter—Hungarian Algorithm Method. *Sensors* **2019**, *19*, 2742. [CrossRef] [PubMed]
22. Liu, X.; Jia, Z.; Hou, X.; Fu, M.; Ma, L.; Sun, Q. Real-time Marine Animal Images Classification by Embedded System Based on Mobilenet and Transfer Learning. In Proceedings of the OCEANS 2019—Marseille, Marseille, France, 17–20 June 2019; pp. 1–5. [CrossRef]
23. Merencilla, N.E.; Alon, A.S.; Fernando, G.J.O.; Cepe, E.M.; Malunao, D.C. Shark-EYE: A Deep Inference Convolutional Neural Network of Shark Detection for Underwater Diving Surveillance. In Proceedings of the 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 17–18 March 2021; pp. 384–388. [CrossRef]
24. Li, L.; Dong, B.; Rigall, E.; Zhou, T.; Dong, J.; Chen, G. Marine Animal Segmentation. *IEEE Trans. Circuits Syst. Video Technol.* **2021**. [CrossRef]
25. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988. [CrossRef]

26. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef]
27. Boom, B.J.; He, J.; Palazzo, S.; Huang, P.X.; Beyan, C.; Chou, H.M.; Lin, F.P.; Spampinato, C.; Fisher, R.B. A research tool for long-term and continuous analysis of fish assemblage in coral-reefs using underwater camera footage. *Ecol. Inform.* **2014**, *23*, 83–97. [CrossRef]
28. Spampinato, C.; Chen-Burger, Y.H.; Nadarajan, G.; Fisher, R.B. Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos. *VISAPP* **2008**, *2008*, 1–6.
29. Patel, H.; Singh Rajput, D.; Thippa Reddy, G.; Iwendi, C.; Kashif Bashir, A.; Jo, O. A review on classification of imbalanced data for wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 4. [CrossRef]
30. Iwendi, C.; Maddikunta, P.K.R.; Gadekallu, T.R.; Lakshmanna, K.; Bashir, A.K.; Piran, M.J. A metaheuristic optimization approach for energy efficiency in the IoT networks. *Softw. Pract. Exp.* **2021**, *51*, 2558–2571. [CrossRef]
31. Wang, S.; Qureshi, M.A.; Miralles-Pechuaán, L.; Huynh-The, T.; Gadekallu, T.R.; Liyanage, M. Explainable AI for B5G/6G: Technical Aspects, Use Cases, and Research Challenges. *arXiv* **2021**, arXiv:2112.04698.