machines

MDPI

Article

YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots

Xuebin Yue [1], Hengyi Li [1], Masao Shimizu [2], Sadao Kawamura [3] and Lin Meng [1,*]

[1] Department of Electronic and Computer Engineering, Ritsumeikan University, Kusatsu 525-8577, Japan; gr0468xp@ed.ritsumei.ac.jp (X.Y.); gr0468kx@ed.ritsumei.ac.jp (H.L.)
[2] Research Organization of Science and Technology, Ritsumeikan University, Kusatsu 525-8577, Japan; mst21786@fc.ritsumei.ac.jp
[3] Department of Robotics, Ritsumeikan University, Kusatsu 525-8577, Japan; kawamura@se.ritsumei.ac.jp
* Correspondence: menglin@fc.ritsumei.ac.jp

Abstract: Due to the workforce shortage caused by the declining birth rate and aging population, robotics is one of the solutions to replace humans and overcome this urgent problem. This paper introduces a deep learning-based object detection algorithm for empty-dish recycling robots to automatically recycle dishes in restaurants and canteens, etc. In detail, a lightweight object detection model YOLO-GD (Ghost Net and Depthwise convolution) is proposed for detecting dishes in images such as cups, chopsticks, bowls, towels, etc., and an image processing-based catch point calculation is designed for extracting the catch point coordinates of the different-type dishes. The coordinates are used to recycle the target dishes by controlling the robot arm. Jetson Nano is equipped on the robot as a computer module, and the YOLO-GD model is also quantized by TensorRT for improving the performance. The experimental results demonstrate that the YOLO-GD model is only 1/5 size of the state-of-the-art model YOLOv4, and the *mAP* of YOLO-GD achieves 97.38%, 3.41% higher than YOLOv4. After quantization, the YOLO-GD model decreases the inference time per image from 207.92 ms to 32.75 ms, and the *mAP* is 97.42%, which is slightly higher than the model without quantization. Through the proposed image processing method, the catch points of various types of dishes are effectively extracted. The functions of empty-dish recycling are realized and will lead to further development toward practical use.

Keywords: empty-dish recycling robot; deep learning; YOLO-GD; model quantification; catch points extraction; hough transform

check for updates

Citation: Yue, X.; Li, H.; Shimizu, M.; Kawamura, S.; Meng, L. YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots. *Machines* **2022**, *10*, 294. https://doi.org/10.3390/machines10050294

Academic Editor: Marcelo H. Ang Jr.

Received: 29 March 2022
Accepted: 20 April 2022
Published: 22 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Currently, a workforce shortage is accelerating due to the declining birth rate and aging population of the world, which have brought heavy pressure on economic and social development. We target the problem and design an automatic empty-dish recycling robot for collecting empty dishes such as cups, bowls, chopsticks, towels et al., after breakfast, lunch, or dinner in a restaurant, canteen, cafeteria, etc.

The global robot industry has entered a period of rapid development, and robots have been widely used in various fields, such as factory automation [1], medical services [2], search-and-rescue [3], automated kitchen [4,5], etc., gradually replacing humans to overcome the problem of the workforce shortage. Yin et al. present a table cleaning and inspection method using a Human Support Robot (HSR), which can be operated in a typical food court setting. In addition, a lightweight deep convolution neural network (DCNN) is proposed to recognize the food litter on top of the table [6]. The commercial feeding assistant robot acquires food without feedback and moves to a pre-programmed location to deliver the food. Candeias et al. use visual feedback to determine whether the food is captured; thus, the food is effectively brought to the user's mouth rather than to a pre-programmed feeding location [7]. However, robots are less involved in the food

service industry, especially in empty-dish recycling, which is still blank. Applying robots to empty-dish recycling mainly faces the following challenges, for example, the complex restaurant working environment and a wide variety of dishes randomly distributed on the table.

With the rapid development of Artificial Intelligence (AI) technology, deep learning technology has been widely used in various fields with excellent performance; the application fields include those not only used in traditional fields such as automatic driving [8] and agriculture harvest [9], but also in special fields such as cultural heritage protection [10,11]. The important step in the dish recycling process is detection. AI technology can achieve a high level of accuracy for detection. This makes the object detection of variety and random distribution of dishes on the table a reality.

Object detection algorithms require a large amount of computational overhead and memory, making them difficult to deploy on embedded mobile devices. The optimal object detection algorithm is to achieve the best trade-off between accuracy and speed. Therefore, the trend in object detection algorithms is towards portable and efficient network architectures that provide an acceptable performance for mobile devices.

Due to a large number of redundant operations in deep learning models [12,13], there has been continuous research on lots of CNN compacting methods, which are proposed recently, such as network pruning [14], model quantization [15], knowledge distillation [16], and lightweight neural networks. Efficient neural network architecture design has a high potential to build with fewer parameters and computational effort. Furthermore, many classical and efficient lightweight convolutional neural networks have emerged, such as MobileNet [17], ShuffleNet [18], GhostNet [19], etc.

We design a deep learning-based object detection algorithm YOLO-GD for empty-dish recycling robots to detect the object dishes. Different image processing techniques are chosen to compute the catch points for different-type dishes. Catch points are the place of dishes, which are caught by robotic fingers for recycling the dishes. The detection network is named YOLO-GD because it uses Ghost Net (G) and Depthwise (D) convolution. The YOLO-GD chooses the lightweight Ghost Net to replace the backbone structure of YOLOv4, and replaces the traditional convolution with depthwise separable convolution and pointwise convolution, which effectively reduces the computational overhead of the network. The catch points are used to control the robot arm to recycle the dishes and place them in the recycling station. Furthermore, the YOLO-GD model is quantized by TensorRT for Floating-point 16 bit ($FP16$) quantization and deployed on the robot computer module, Jetson Nano.

The four major contributions of this paper are as follows:

- We design a lightweight dish detection model YOLO-GD for empty-dish recycling robots, which significantly reduces parameter numbers and improves the detection accuracy.
- We design a dish catch point method to effectively extract the catch points of different types of dishes. The catch points are used to recycle the dishes by controlling the robot arm.
- We have realized the quantification of the lightweight dish detection model YOLO-GD without losing accuracy and deploy it on the embedded mobile device, Jetson Nano.
- This paper also creates a dish dataset named Dish-20 (http://www.ihpc.se.ritsumei.ac.jp/obidataset.html; accessed on 28 March 2022), which contains 506 images in 20 classes. It not only provides training data for object detection in this paper but also helps in the field of empty-dish recycling automation.

The rest of the paper is organized as follows, Section 2 introduces the related work about robotics applications, object detection, and model quantization deployed on embedded devices. Section 3 gives a detailed explanation of deep learning-based object detection, which is equipped with the empty-dish recycling robot, dish catch points extraction, TensorRT quantization model, and deployment on embedded mobile devices. Section 4 presents the results of the relevant model comparison, and the experimental results without and with model quantification, including detection accuracy, model weights,

inference speed, etc. A discussion and future work are provided in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

The realization of automatic empty-dish recycling can effectively replace human beings to complete the work and alleviate the problem of workforce shortage. This paper aims to design an empty-dish recycling robot for realizing automatic empty-dish recycling. However, high-accurate object detection for detecting the target dishes, and the catch points' calculation for controlling the robotic arm, are still important issues. Furthermore, compacting the object detection model to the embedded robot is also a challenge in this research. In this section, we review current related research work on robotics, object detection algorithm, model quantization, and deploy it on embedded devices.

2.1. Research of Robotics

Fukuzawa et al. proposed a robotic system consisting of a robotic manipulator with six degrees of freedom, a robotic hand capable to catch and suction operations, and a 3D camera for dish detection to perform the take-out operation of various dishes from a commercial dishwasher [4]. Zhu et al. developed an automated dish tidying-up robot mechanism for cleaning dishes in a self-service restaurant with a large number of dishes. The dishes are placed on the conveyor belt by the guest, and the robot is mainly responsible for the process of sorting and collecting the dishes [20]. Kawamura et al. designed a three-degree freedom micro-hand consisting of a thin pneumatic rubber actuator generating three degrees of freedom of motion. The micro-hand contracts in the longitudinal direction and bends in any direction by changing the applied air pressure pattern to the artificial muscles, which may be expected to be used in areas such as the flexible catch of a dish [21]. Kinugawa et al. have developed a new underactuated robotic hand for circular standard dishes (square or other types of dishes are not considered), which is an important factor for the realization of a fully automatic dishwashing system [22].

2.2. Object Detection

The object detection model based on deep learning is capable of achieving high-speed object detection and object bounding box segmentation. These models are mainly divided into two categories. One is the one-stage detection algorithm, including YOLO [23], SSD [24], Retina Net [25], etc. Another is the two-stage detection algorithm, including R-CNN [26], Fast R-CNN [27], Faster R-CNN [28], etc. YOLOv4 is widely adopted due to its high speed, high accuracy, and relatively simple design [29].

YOLO predicts multiple BBox positions and classes at once, regards detection as a regression problem, and combines the two stages of candidate area and detection, with simple structure and fast detection speed [30]. A modified Tiny YOLOv2 is proposed to recognize small objects such as the shuttlecock, and by modifying the loss function, the detection speed of small objects is improved adaptively and applied to other tasks of detecting small objects at high speed [31]. Zhang et al. proposed a state-of-the-art lightweight detector, namely, CSL-YOLO. Through a lightweight convolution method Cross-Stage Lightweight (CSL) Module, it generates redundant features from cheap operations with excellent results [32]. TRC-YOLO is proposed by pruning the convolution kernel of YOLOv4-tiny and introducing an expanded convolution layer in the residual module of the network, which improves the model's mean average precision (mAP) and real-time detection speed [33]. A lightweight three-stage detection framework is composed of a Coarse Region Proposal (CRP) module, the lightweight Railway Obstacle Detection Network (RODNet), and the post-processing stage, to identify obstacles in the single railway image [34].

2.3. Quantification and Deployment

Hirose et al. simultaneously measured the input data and the distribution of values in the middle layer during quantization with TensorRT, suppressing the deterioration of the accuracy caused by quantization [35]. Jeong et al. proposed a parallelization approach to maximize the throughput of a single deep learning application using GPUs and NPUs by exploiting various types of parallelism in TensorRT [36]. Jeong et al. proposed a TensorRT-based framework that supports various optimization parameters to accelerate deep learning applications targeting Jetson with heterogeneous processors, including multithreading, pipelining, buffer assignment, and network duplication [37]. Stäcker et al. analyzed two representative object detection networks, which are deployed on edge AI platforms, and observed a slight advantage in using TensorRT for convolutional layers and TorchScript for fully connected layers. In terms of the optimized setup selection for deployment, quantization significantly reduces the runtime while having only a small impact on the detection performance [38].

A novel neural network based on the SSD framework, including a feature extractor using the improved MobileNet and a lightweight module, is proposed for fast and low-cost high-speed railway intrusion detection. It is deployed on the Jetson TX2 with quantization by TensorRT, achieving 98.00% mAP, and has a 38.6 ms average processing time per frame [39]. The advantage of YOLO has been proven in a wide range of applications [40,41], while excellent real-time performance and fewer network parameters enable YOLO to be applied in edge detection. Yue et al. proposed a deep learning-based empty-dish recycling robot, using YOLOv4 as a detection network for the dish, $FP16$ quantization of the detection model by TensorRT, and deployment on the Jetson Nano, with more than 96.00% high accuracy on *Precision*, *Recall*, and F_1 values, and an inference speed of 0.44 s for per image were achieved. However, this method only detects dishes and does not extract catch points. The inference time of the detection model does not meet the requirements of real-time detection [42]. Wang et al. used the neural network YOLOv4 to detect dirty eggs and used TensorRT to accelerate the detection process; the system was deployed on the Jetson Nano. The method obtained an accuracy of 75.88% and achieved a speed of 2.3 frames per second (FPS) [15].

3. Object Detection System Embedded in Empty-Dish Recycling Robots

3.1. Overview of Empty-Dish Recycling Robot

Figure 1a shows the proposed empty-dish recycling robot, which consists of a robotic body, robotic arms, cameras, robot fingers, and recycling stations. Figure 1 shows the workflow of the empty-dish recycling robot. (a) Shows the initial state of the empty-dish recycling robot after arriving at the food-receiving place. The dish detection model is loaded in the process, waiting for the camera to take an image for detection. (b) Shows the process of collecting images and detecting dishes, taking images by Intel RealSense D435, detecting the dish by the proposed YOLO-GD, and calculating the catch points of the different dish types by different image processing methods. (c) Shows the process of the robot catching dishes. Through the dish category and catch points provided in (b), the embedded control system controls the robotic arm to catch the dish. (d) Shows the process of recycling the dish and putting it into the recycling station.

3.2. YOLO-GD Framework

To achieve high-speed and real-time dish detection, a lightweight detection network YOLO-GD is proposed, as shown in Figure 2. The network mainly consists of three parts: feature extraction, feature fusion, and result prediction. The purpose of network development is to realize a high-accuracy network with low computation. YOLO-GD adopts a lightweight feature extraction module; in addition, both the depthwise separable convolution and pointwise convolution are used to replace the traditional convolution operation, which effectively reduces the computational overhead of the network.

In the feature extraction stage, Ghost Net [19,43] replaces the CSPDarknet53 [44] module in the YOLOv4 network. Ghost Net aims to generate more feature maps with cheap operations. The main operation generates Ghost feature maps by applying a series of linear transformations based on original feature maps and extracting the required information from the original features at a low overhead.

Figure 3 details each module in the Ghost Net. *G_Bottleneck* is mainly composed of *G_bottleneck*, where s represents the stride size, and *SE* represents adding an SE Net [45] module; "×" represents an iterative operation. *G_bottleneck* is mainly composed of a Ghost module. As for the case where $stride = 1$, the first Ghost module is used to extend the layer and increase the number of channels. The second Ghost module reduces the number of channels to match the shortcut path. The input of the first Ghost module and the output of the second Ghost module are connected in a shortcut. After the first layer, batch normalization (BN) and $Relu$ nonlinearity are used, and after the second layer, only BN is used. As for the case where $stride = 2$, the shortcut path uses depthwise separable convolution with $stride = 2$ for downsampling, and point convolution for channel adjustment.

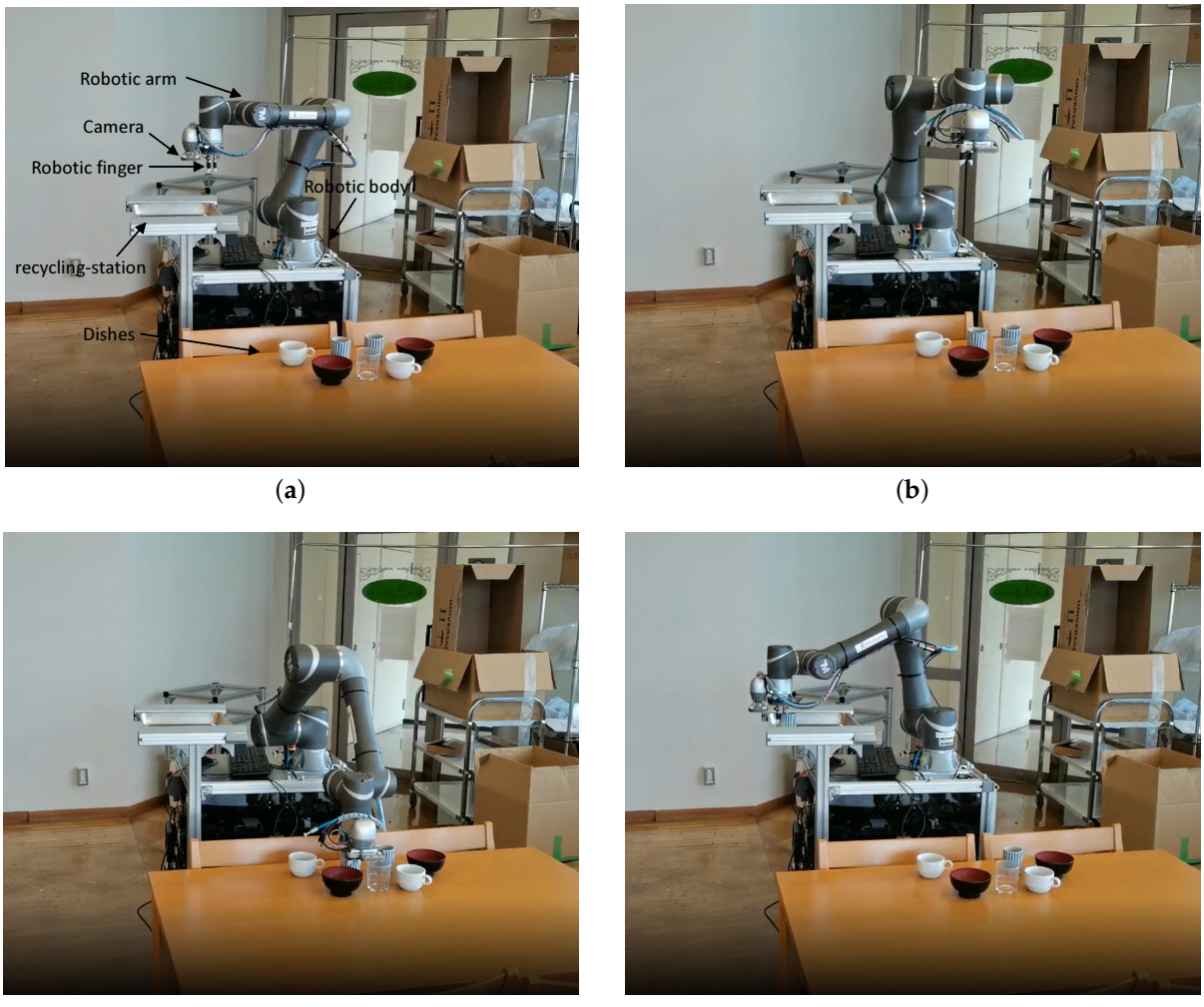


Figure 1. Overview of empty-dish recycling robot. (**a**) Initial state, (**b**) detecting state, (**c**) catching state, and (**d**) recycling state.

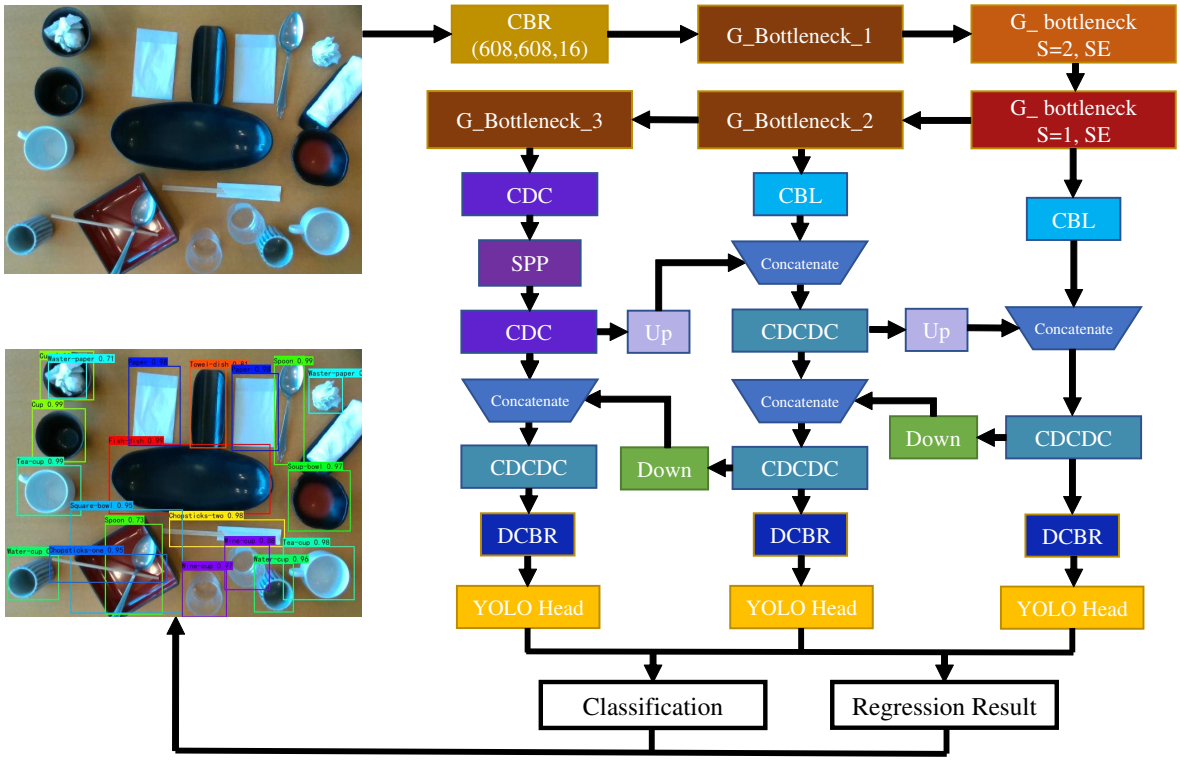


Figure 2. The framework of YOLO-GD.

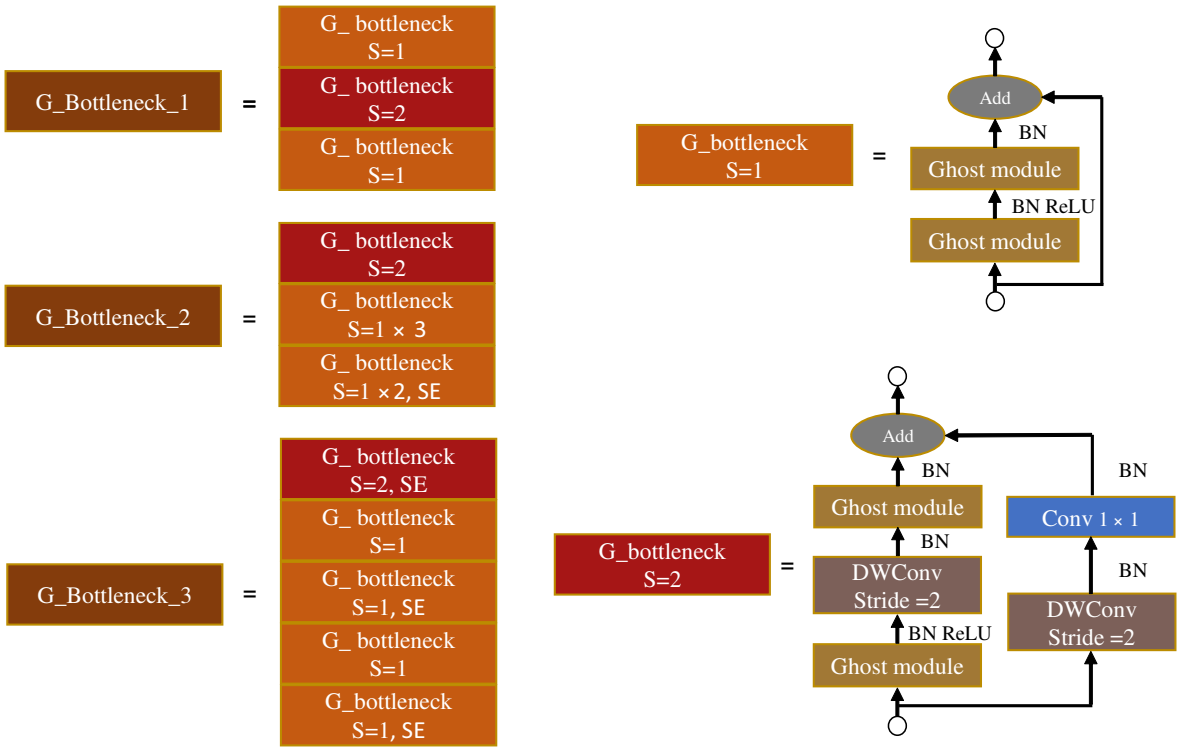


Figure 3. Detailed explanation of feature extraction module.

In the feature fusion and result prediction stages, spatial pyramid pooling (*SPP*) is inserted into the output of the network to extract the spatial feature information of different sizes and increase the receptive field information of the network. *SPP* can improve the

robustness of the model for spatial layout and object variability [46]. The calculation of the *SPP* is as follows:

$$SPP = C(f^{5\times5}MaxPool(F) + f^{9\times9}MaxPool(F) + f^{13\times13}MaxPool(F) + F). \quad (1)$$

Among them, F means feature map, C means concatenate operation, $f^{5\times5}$ means 5×5 filter, and *MaxPool* means max-pooling operation. The Path Aggregation Network (*PANet*) [47] can fuse features between three different output network layers, as shown in Figure 2. *PANet* obtains geometric detail information from the bottom network and contour information from the top network, ensuring the rich semantic information of the network and strengthening the feature extraction ability.

YOLOHead predicts the classes, confidence, and coordinate information of the dish at the same time by setting the convolution operation of the number of filters. A detailed explanation is shown in Figure 4.

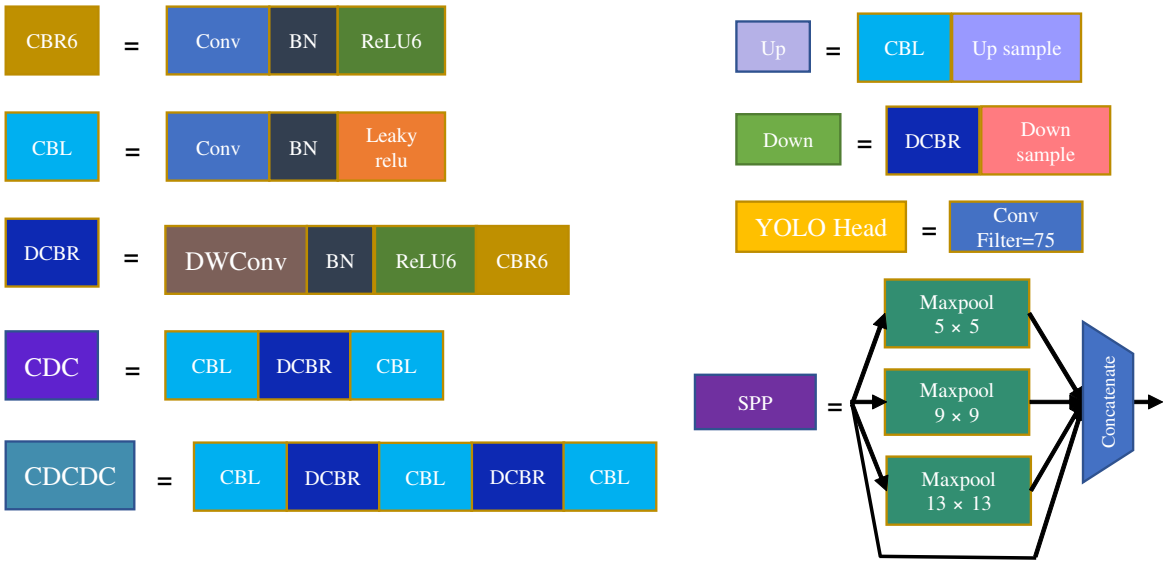


Figure 4. Detailed explanation of feature fusion and result prediction module.

To reduce the overhead of the model in the feature fusion and result prediction stages, all 3×3 convolution operations are replaced by 1×1 convolution, 3×3 depthwise separable convolution, and 1×1 convolution [30].

3.3. Extraction of Catch Points

When extracting catch points in the whole image, mutual interference exists between different classes. Therefore, we use the detection results and the coordinate information of YOLO-GD to segment the target dish and extract the catch points. For different dish types, we use different feature point extraction methods. The types are mainly divided into circle, ellipse, square, and polygon.

- Circle:
  Hough transform is used to detect the contours of the circle dish. The equation for a circle in Cartesian coordinates is shown in Equation (2).

$$(x-a)^2 + (y-b)^2 = r^2 \quad (2)$$

  where (a, b) is the center of the circle and r is the radius, which can also be expressed as Equation (3).

$$\begin{aligned} a &= x - r\cos\theta \\ b &= y - r\sin\theta \end{aligned} \tag{3}$$

In the Cartesian xy coordinate system, all points on the same circle have the same equation for the circle. They map to the same point in the abr coordinate system. In the abr coordinate system, the number of points should have the total pixels of the circle. By judging the number of points at each intersection in the abr coordinate system, points greater than a threshold are considered a circle.
For the segmented circular dish images, grayscale images, canny edge detection [48], and Gaussian filtering [49] are performed to extract the contours of the dish and reduce the interference. Through Hough transform circle detection, the center point coordinates, radius, and other information of the contour are extracted. The center point coordinates of the circle are moved up by a distance of the radius and set as the catch point [50].

- Ellipse:
In the Cartesian xy coordinate system, the maximum distance from any point to the ellipse, the point with the smallest distance is the center of the ellipse, and the smallest maximum distance is the length of the long axis of the ellipse. As shown in Equation (4).

$$\frac{((x-p)\cos\theta + (y-q)\sin\theta)^2}{a^2} + \frac{(-(x-p)\sin\theta + (y-q)\cos\theta)^2}{b^2} = 1 \tag{4}$$

where (p, q) is the center of the ellipse, a and b are the major and minor axes of the ellipse, respectively, and θ is the rotation angle.
For the elliptical dish, grayscale conversion and canny edge detection are used to extract ellipse features. The disconnected contour lines are connected and their boundaries are smoothed by the closing operation in morphological processing [10]. The contour finding method is used to find the contour points of the ellipse, and the ellipse center, long axis, short axis, and rotation angle of the ellipse are extracted by ellipse fitting in OpenCV.
In the segment elliptical dish image, the coordinates of the catch points are shown in Equation (5).

$$\begin{aligned} x &= p + \frac{1}{2}b\cos\theta \\ y &= q + \frac{1}{2}b\sin\theta \end{aligned} \tag{5}$$

- Square:
The straight-line equation is as follows:

$$x\cos\theta + y\sin\theta = \rho \quad \rho \le 0,\ 0 \le \theta \le \pi \tag{6}$$

where ρ is the distance of the straight line to the original point and θ is the angle between the straight line and the positive direction of the Cartesian coordinate x-axis. The different points on the straight line are transformed in the polar coordinate plane ρ-θ into a set of sinusoids intersecting at one point. Determine the two-dimensional statistics on the polar coordinate plane and select the peak value. The peak value is the parameter of a straight line in the image space, thus realizing the straight line detection in the Cartesian coordinate.
We consider the intersection of the two lines, L_1 and L_2, in the Cartesian coordinate, with L_1 being defined by two distinct points, (x_1, y_1) and (x_2, y_2), and L_2 being defined by two distinct points, (x_3, y_3) and (x_4, y_4).

$$\cos\alpha = \frac{(\vec{a}\cdot\vec{b})}{|\vec{a}|\cdot\left|\vec{b}\right|} \tag{7}$$

where $\vec{a}$ and $\vec{b}$ are the vectors of L_1 and L_2, respectively, and α is the intersection angle between L_1 and L_2.
The intersection P of L_1 and L_2 can be defined using determinants,

$$P_x = \frac{\begin{vmatrix}\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix}\end{vmatrix}}{\begin{vmatrix}\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}\end{vmatrix}}, \quad P_y = \frac{\begin{vmatrix}\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}\end{vmatrix}}{\begin{vmatrix}\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}\end{vmatrix}}. \tag{8}$$

The determinants are written out as:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - x_2y_1)(x_3 - x_4) - (x_3y_4 - x_4y_3)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_2)}, \right. \\ \left. \frac{(x_1y_2 - x_2y_1)(y_3 - y_4) - (x_3y_4 - x_4y_3)(y_1 - y_2)}{(x_1 - x_2)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_2)} \right) \tag{9}$$

The edge features of the square dish are highlighted by grayscale conversion and canny edge detection. The straight lines in the image are extracted using straight-line detection with Hough transform [51], and the straight lines with angles around 90°are selected by calculating the angle of all the straight lines. The intersection points are calculated for the retained straight lines, and the minimum circumscribed rectangle of all intersection points is calculated. The catch point is the midpoint of one side of the minimum circumscribed rectangle.

- Polygon:
  For the irregular dish, grayscale conversion, Gaussian filtering, and binarization conversion are performed to clarify the dish contours. The contour finding function in OpenCV is applied for finding all connected contours and taking the maximum value as the feature of the dish. All points in the contour are processed by the minimum circumscribed rectangle, and the center point is extracted as the catch point.

3.4. Model Quantification and Deployment

TensorRT is a high-performance deep learning inference optimizer that provides low-latency, high-throughput deployment inference for deep learning applications. TensorRT supports $INT8$ and $FP16$ computation for accelerating inference by achieving an ideal trade-off between reducing the computation and maintaining accuracy. TensorRT provides only forward propagation, i.e., inference, and without an in-training process [35].

Figure 5 shows the steps of TensorRT to reconstruct and optimize the network structure, which is mainly divided into two parts: model compression and hardware mapping. Model compression eliminates useless output layers in the network by parsing the network model for reducing computation. For the network vertical integration, the convolution, BN, and $Relu$ layer of the current mainstream neural network are merged into one layer. A horizontal combined network means fusing layers whose inputs are the same tensor and perform the same operation. For the concatenate layer, the input of the contact layer is directly sent to the following operation, without performing the concatenate separately and calculating the input. Furthermore, 32-bit floating-point operations are quantized to 16-bit floating-point

operations or 8-bit integer operations. In hardware mapping, the kernel selects the best pre-implemented algorithm based on different batch sizes and problem complexity and uses streaming techniques in CUDA to maximize parallel operations [36,52].

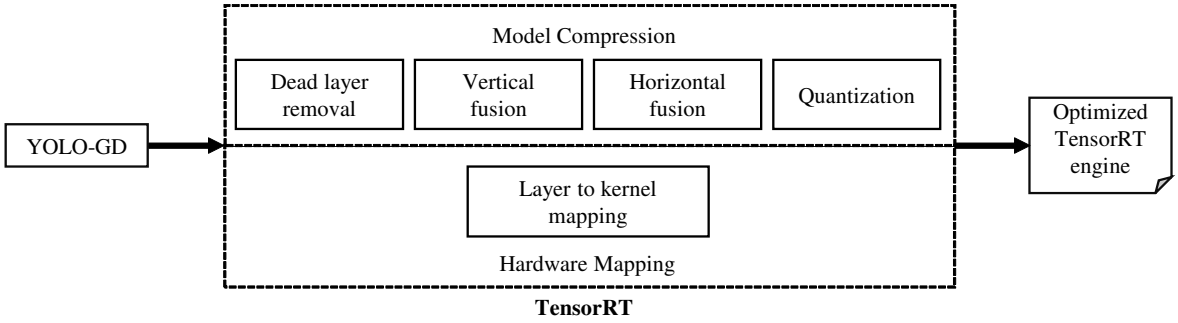


Figure 5. TensorRT optimization steps.

The robot system employs the Jetson Nano [42] as the computation model, which does not support the $INT8$ type data. Hence, the robot system uses TensorRT to quantify the object detection model as $FP16$ type data.

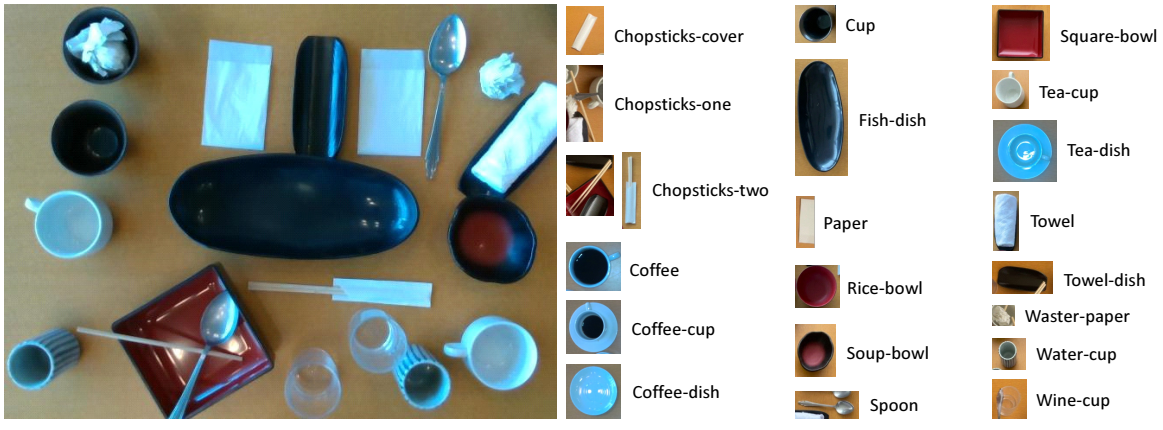
4. Evaluation

In terms of evaluation, the operating system is Ubuntu 21.04, the CPU is an Intel Core i9-10900 2.8GHz processor with 32GB RAM, the GPU is RTX 3080Ti, the CUDA version is 11.4, and the GPU acceleration library cuDNN is 8.2.4.

This paper uses transfer learning to train YOLO-GD. YOLO-GD loads the weights of YOLO's pre-trained VOC 07+12 dataset based on the Ghost Net as the backbone network. The epochs are set to 400; the first 200 epochs freeze the feature extraction part of the model to train the feature fusion and the resulting prediction layer of the model. The latter 200 epochs unfreeze the feature extraction part and train the model as a whole [53,54].

4.1. Dataset

This paper first creates a dish dataset, named Dish-20, which contains 506 images in 20 classes. In the experimentation, 409 images are used for training, 46 images are used for validation, and 51 images are used for testing. Figure 6a shows an example of a dish image, and (b) shows the definition of the 20 classes. The image size of the dataset is set as the default size of YOLO-GD (416 × 416).



(a) (b)

Figure 6. Example of dish image and definition of classes. (**a**) An example of a dish image and (**b**) is the definition of the 20 classes.

4.2. Performance Indexes

Precision, *Recall*, F_1, and mAP, have been used to evaluate and compare the dish detection performance, which are listed in Equations (10)–(14), respectively.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

TP represents a positive sample correctly classified; FP represents a positive sample of the misclassification; FN represents a negative sample of the misclassification [55]. *Precision* and *Recall* respectively represent the proportion of the number of correctly predicted samples to the total number of positive class predictions, and the positive class is predicted as the number of positive class samples in the total number of positive samples.

$$F_1 = \frac{2 \times Precision \times Recall}{Presicion + Recall} \tag{12}$$

The F_1 score is obtained by taking the harmonic mean of *Precision* and *Recall*, namely the reciprocal of the average of the reciprocal of *Precision* and the reciprocal of *Recall* [56].

$$AP = \sum_{n} (R_{n+1} - R_n) P_{max}(R_{n+1}) \tag{13}$$

The Precision–Recall curve ($P - Rcurve$) is derived from the relationship between *Precision* and *Recall*. The Average Precision (AP) of all classes is the area of the region surrounded by the curve and the axes. In practical applications, we do not directly calculate the $P - Rcurve$ but smooth the $P - Rcurve$. That is, for each point on the $P - Rcurve$, the value of *Precision* takes the value of the largest *Precision* on the right side of the point, as shown in Equation (13). Among them, R_n represents the *Recall* of the n-th value, and P_{max} represents the largest *Precision* value on the right side of the *Recall* value [57].

The mAP is obtained by averaging the AP of all classes (C) in the dataset, as shown in Equation (14).

$$mAP = \frac{1}{C} \sum_{j}^{C} AP_j \tag{14}$$

We also use the detection and evaluation indicators in COCO API, where the calculation of AP is different from that mentioned above. For a certain classification, the *Recall* is equidistantly divided into eleven values [0.1, . . . , 0.9, 1], and the maximum *Precision* is calculated for each *Recall* value, and then the average of these eleven *Precision* values is *AveragePrecision* [53,57], as shown in Equation (15).

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0.1, \ldots, 0.9, 1\}} P_{max}(R) \tag{15}$$

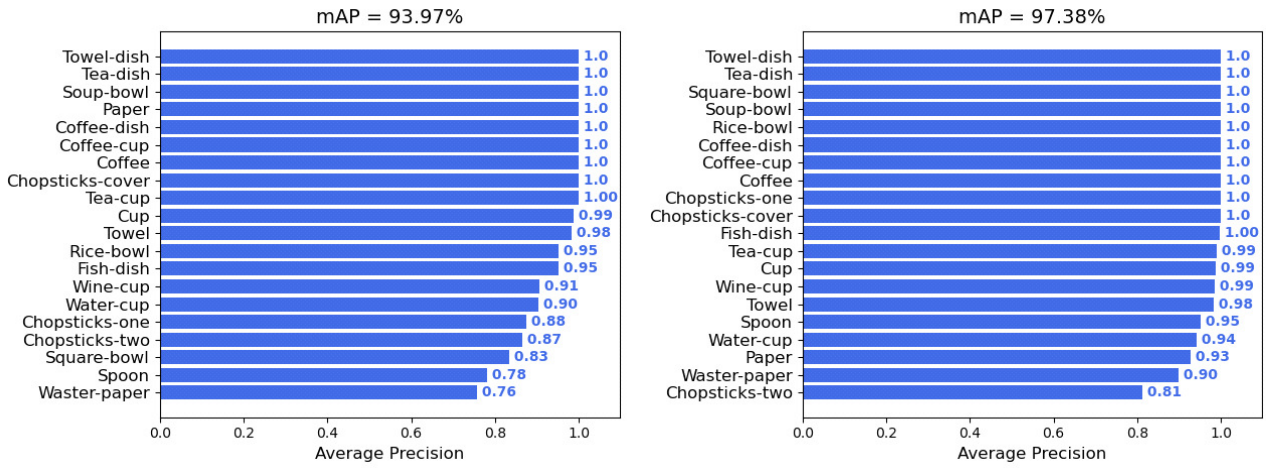
4.3. Experimental Results

To verify the performance of YOLO-GD, we analyze the test results of YOLOv4 and YOLO-GD. In addition, the parameters, weights, and Floating-Point Operations ($FLOPs$) of the two models are compared. Inference times and test results are also compared with and without YOLO-GD quantization on the Jetson Nano.

4.3.1. Performance Validation of the YOLO-GD Model

AP and mAP are important indicators that reflect the detection accuracy of object detection models. The larger the value of AP or mAP, the higher the accuracy of the model. Figure 7a shows that the mAP of YOLOv4 is 93.97%, and (b) shows that the mAP

of YOLO-GD achieves 97.38%, which is 3.41% higher than YOLOv4. It is also proven that the overall performance of YOLO-GD is better than YOLOv4.



(a) YOLOv4 (b) YOLO-GD

Figure 7. AP and *mAP* of YOLOv4 and YOLO-GD. (**a**) Is the detection result of YOLOv4 and (**b**) is the detection result of YOLO-GD.

Tables 1 and 2 show the experimental results of YOLOv4 and YOLO-GD, respectively. The results of YOLO-GD are significantly better than YOLOv4. Some classes of results have improved the accuracy with few errors; for example, the F_1 value of "Chopsticks-one" is increased from 0.93 to 1.00, the *Recall* is increased from 87.50% to 100%, and the *AP* is increased from 87.50% to 100%. The F_1 value of "Fish-dish" is increased from 0.96 to 0.98, the *Recall* is increased from 96.30% to 100%, the *Precision* is increased from 96.30% to 96.43%, and the *AP* is increased from 95.06% to 99.74%. The F_1 value of "Rice-bowl" is increased from 0.98 to 1.00, *Recall* is increased from 95.24% to 100%, and *AP* is increased from 95.24% to 100%. The F_1 value of "Spoon" is increased from 0.84 to 0.98, *Recall* is increased from 78.57% to 95.24%, *Precision* is increased from 89.19% to 100%, and *AP* is increased from 78.06% to 95.24%. The F_1 value of "Square-bowl" is increased from 0.91 to 0.96, *Recall* is increased from 83.33% to 100%, *Precision* is decreased from 100% to 92.31%, *AP* is increased from 83.33% to 100%. The F_1 value of "Waster-paper" is increased from 0.78 to 0.94, the *Recall* is increased from 78.38% to 91.89%, the *Precision* is increased from 78.38% to 97.14%, and the *AP* is increased from 75.65% to 89.88%. The F_1 value of "Water-cup" is increased from 0.95 to 0.97, *Recall* is increased from 91.18% to 94.12%, *Precision* is increased from 98.41% to 100%, and *AP* is increased from 90.45% to 94.12%. The F_1 value of "Wine-cup" is increased from 0.95 to 0.99, and *Recall* is increased from 90.54% to 98.65%. *AP* is increased from 90.54% to 98.65%.

However, the accuracy of some classes is decreased: the F_1 value of "Chopsticks-two" is decreased from 0.91 to 0.89, the *Recall* is decreased from 87.04% to 81.48%, the *Precision* is increased from 95.92% to 97.78%, and the *AP* is decreased from 86.57% to 81.28%. The F_1 value of "Paper" is decreased from 1.00 to 0.96, the *Recall* is decreased from 100% to 92.86%, and the *AP* is decreased from 100% to 92.86%.

Among the twenty classes, eight classes improve, two classes decrease, and the remaining classes are unchanged. The *Precision* of "Chopsticks-two" increased by 1.86%, but the *Recall* decreased by 5.56%, indicating that some of "Chopsticks-two" are predicted to other classes. This leads to a decrease in F_1 and *AP* values. There are various shapes of "Paper", forming a single class with multiple shapes, which leads to a decrease in the accuracy of recognition. The results demonstrate that YOLO-GD is better than YOLOv4 in the detection of the dish.

Table 1. Test results of different dish classes in YOLOv4.

Category	F_1	Recall	Precision	AP
Chopsticks-cover	1.00	100.00%	100.00%	100.00%
Chopsticks-one	0.93	87.50%	100.00%	87.50%
Chopsticks-two	0.91	87.04%	95.92%	86.57%
Coffee	1.00	100.00%	100.00%	100.00%
Coffee-cup	1.00	100.00%	100.00%	100.00%
Coffee-dish	1.00	100.00%	100.00%	100.00%
Cup	0.99	98.78%	98.78%	98.72%
Fish-dish	0.96	96.30%	96.30%	95.06%
Paper	1.00	100.00%	100.00%	100.00%
Rice-bowl	0.98	95.24%	100.00%	95.24%
Soup-bowl	1.00	100.00%	100.00%	100.00%
Spoon	0.84	78.57%	89.19%	78.06%
Square-bowl	0.91	83.33%	100.00%	83.33%
Tea-cup	1.00	100.00%	99.05%	99.99%
Tea-dish	1.00	100.00%	100.00%	100.00%
Towel	0.99	98.18%	100.00%	98.18%
Towel-dish	1.00	100.00%	100.00%	100.00%
Waster-paper	0.78	78.38%	78.38%	75.65%
Water-cup	0.95	91.18%	98.41%	90.45%
Wine-cup	0.95	90.54%	100.00%	90.54%

Table 2. Test results of different dish classes in YOLO-GD.

Category	F_1	Recall	Precision	AP
Chopsticks-cover	1.00	100.00%	100.00%	100.00%
Chopsticks-one	1.00	100.00%	100.00%	100.00%
Chopsticks-two	0.89	81.48%	97.78%	81.28%
Coffee	1.00	100.00%	100.00%	100.00%
Coffee-cup	1.00	100.00%	100.00%	100.00%
Coffee-dish	1.00	100.00%	100.00%	100.00%
Cup	0.99	98.78%	98.78%	98.69%
Fish-dish	0.98	100.00%	96.43%	99.74%
Paper	0.96	92.86%	100.00%	92.86%
Rice-bowl	1.00	100.00%	100.00%	100.00%
Soup-bowl	1.00	100.00%	100.00%	100.00%
Spoon	0.98	95.24%	100.00%	95.24%
Square-bowl	0.96	100.00%	92.31%	100.00%
Tea-cup	0.99	99.04%	99.04%	99.04%
Tea-dish	1.00	100.00%	100.00%	100.00%
Towel	0.99	98.18%	100.00%	98.18%
Towel-dish	1.00	100.00%	100.00%	100.00%
Waster-paper	0.94	91.89%	97.14%	89.88%
Water-cup	0.97	94.12%	100.00%	94.12%
Wine-cup	0.99	98.65%	100.00%	98.65%

The COCO API is employed to evaluate the performance of the training model. The performance of YOLO-GD is tested by calculating AP_{11} (Average Precision) and AR (Average Recall) based on different IoU values, area sizes, and the number of objects contained in the image. The AP_{11} is averaged according to the 10 IoU thresholds of 0.50 to 0.95 (the step is 0.50), and the AP calculation is performed when $IoU = 0.50$ and $IoU = 0.75$, respectively. AP_{11} and AR are calculated by different detection areas (*Small* or *Medium*, or *Large*) of the object. AR is calculated by the different maximum number of objects detected in each image (1, 10, and 20).

Tables 3 and 4 show that for *Small* detection areas, the AP_{11} and AR values are -1.000, which means the relevant dish is not detected in the *Small* detection area. The AP_{11} and AR of the *Large* detection area are higher than those of the *Medium* detection area. The

dish works well in the *Large* detection area, indicating that the anchor box of YOLO-GD should be adjusted to increase the detection area of a small area. As the maximum number of detected objects increases from 1 to 10, the AP_{11} and AR values increase significantly, but when the maximum number of detected objects increases from 10 to 20, the AP_{11} and AR values remain unchanged. This demonstrates that when the model detects each image, the maximum number of detections for each dish does not exceed 10. The values of YOLO-GD are higher than those of YOLOv4. These evaluation indicators validate that the performance of the YOLO-GD model is better.

A relevant parameters comparison of YOLOv4 and YOLO-GD is shown in Table 5. The weights, parameters, and *FLOPs* of YOLO-GD are significantly lower than YOLOv4. The weight of YOLO-GD is 45.80 MB, which is 82.12% lower than YOLOv4 (256.20 MB); the number of parameters is 11.17 M, which is 82.50% lower than YOLOv4 (63.84 M); the *FLOPs* is 6.61 G, which is 88.69% lower than YOLOv4 (58.43 G). It is proven that the YOLO-GD model is only 1/5 the size of the YOLOv4 model, which is more lightweight.

Table 3. Results on YOLOv4 using COCO API.

IoU	Area	maxDets	AP_{11}	AR
0.50:0.95	All	20	0.726	-
0.50	All	20	0.936	-
0.75	All	20	0.884	-
0.50:0.95	Small	20	−1.000	-
0.50:0.95	Medium	20	0.706	-
0.50:0.95	Large	20	0.753	-
0.50:0.95	All	1	-	0.566
0.50:0.95	All	10	-	0.762
0.50:0.95	All	20	-	0.762
0.50:0.95	Small	20	-	−1.000
0.50:0.95	Medium	20	-	0.732
0.50:0.95	Large	20	-	0.787

Table 4. Results on YOLO-GD using COCO API.

IoU	Area	maxDets	AP_{11}	AR
0.50:0.95	All	20	0.753	-
0.50	All	20	0.970	-
0.75	All	20	0.907	-
0.50:0.95	Small	20	−1.000	-
0.50:0.95	Medium	20	0.709	-
0.50:0.95	Large	20	0.766	-
0.50:0.95	All	1	-	0.588
0.50:0.95	All	10	-	0.788
0.50:0.95	All	20	-	0.788
0.50:0.95	Small	20	-	−1.000
0.50:0.95	Medium	20	-	0.734
0.50:0.95	Large	20	-	0.795

Table 5. Comparison of parameters between YOLOv4 and YOLO-GD.

Model	Weights	Parameters	FLOPs
YOLOv4	256.20 MB	63.84 M	58.43 G
YOLO-GD	45.80 MB	11.17 M	6.61 G

4.3.2. YOLO-GD Quantification, Deployment, and Result Analysis

We deploy the YOLO-GD dish detection model on the robot control system, the Jetson Nano. Table 6 shows the per-image inference time and *FPS* on the Jetson Nano without and with quantization. The per-image inference time is the average inference time of

49 images. Under normal circumstances, the per-image inference time of the YOLO-GD model on the Jetson Nano is 207.92 ms and the FPS is 4.81. With $FP16$ quantization using TensorRT, the per-image inference time is 32.75 ms, and the FPS is 30.53. After quantization, the per-image inference time is decreased by 84.25%, which meets the real-time detection requirements.

Table 6. Comparison of YOLO-GD inference times without and with quantification on Jetson Nano.

Model	FPS	Inference Time per Image
Unquantized model	4.81	207.92 ms
Quantified model	30.53	32.75 ms

Figure 8 shows the quantified YOLO-GD results with an mAP of 97.42%, which is slightly higher than the model without quantization. The overall detection accuracy of YOLO-GD does not decrease with quantization, while the detection speed per image increased by 84.25%, which proves the feasibility of the quantization method.

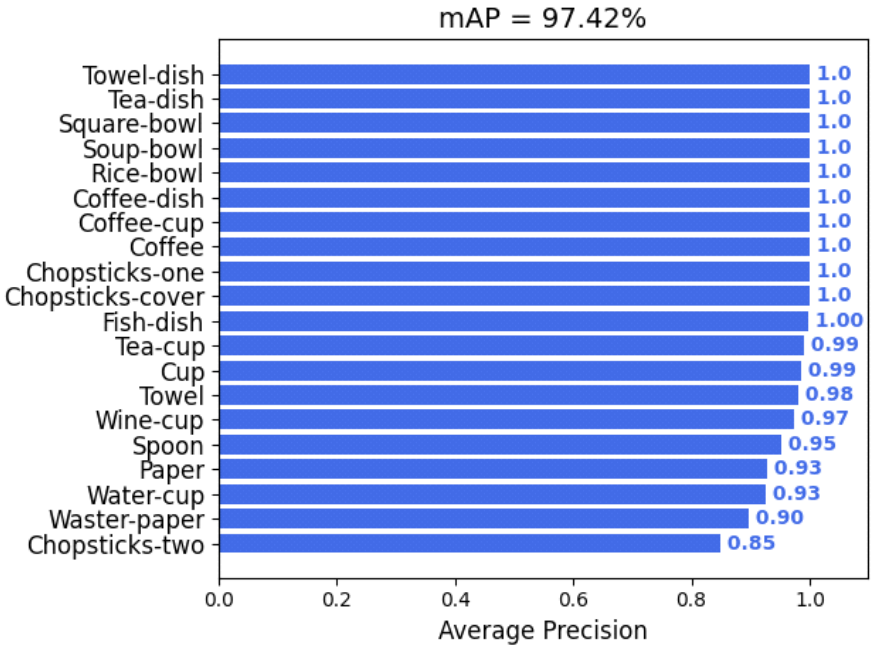


Figure 8. AP and mAP of YOLO-GD on Jetson Nano with quantification by FP16.

Tables 2 and 7 show that the results without and with quantified YOLO-GD and the results of "Chopsticks-two" are increased as a whole, but the $Precision$ and F_1 values of "Chopsticks-one" are decreased, and the AP value of "Waster-paper" is decreased by 0.07%. The $Recall$, AP, and F_1 values in "Water-cup" are also decreased, and the $Recall$ and AP values in "Wine-cup" are decreased. Figure 7 shows that the mAP value of YOLO-GD with quantization is 97.42%, which is higher than without quantization. It is proven that after YOLO-GD is quantized by TensorRT's $FP16$, the detection accuracy remains unchanged.

The results of YOLO-GD using COCO API without and with quantification are shown in Tables 4 and 8. AP_{11} value decrease by 0.006, AP_{11} decrease by 0.004 when $IoU = 0.75$, AP_{11} decrease by 0.007 when the detection area is $Large$, and AR decrease by 0.007 when maximum number of objects detected is 1. AR decreases by 0.008 when the detection area is $Large$ and the maximum number of objects detected are 10 and 20.

Table 7. Test results of YOLO-GD quantization model on different dish classes.

Category	F_1	Recall	Precision	AP
Chopsticks-cover	1.00	100.00%	100.00%	100.00%
Chopsticks-one	0.97	100.00%	94.12%	100.00%
Chopsticks-two	0.91	85.19%	97.87%	84.99%
Coffee	1.00	100.00%	100.00%	100.00%
Coffee-cup	1.00	100.00%	100.00%	100.00%
Coffee-dish	1.00	100.00%	100.00%	100.00%
Cup	0.99	98.78%	98.78%	98.68%
Fish-dish	0.98	100.00%	96.43%	99.74%
Paper	0.96	92.86%	100.00%	92.86%
Rice-bowl	1.00	100.00%	100.00%	100.00%
Soup-bowl	1.00	100.00%	100.00%	100.00%
Spoon	0.98	95.24%	100.00%	95.24%
Square-bowl	0.96	100.00%	92.31%	100.00%
Tea-cup	0.99	99.04%	99.04%	99.04%
Tea-dish	1.00	100.00%	100.00%	100.00%
Towel	0.99	98.18%	100.00%	98.18%
Towel-dish	1.00	100.00%	100.00%	100.00%
Waster-paper	0.94	91.89%	97.14%	89.81%
Water-cup	0.96	92.65%	100.00%	92.65%
Wine-cup	0.99	97.30%	100.00%	97.30%

Table 8. Results of the YOLO-GD with quantization using COCO API.

IoU	Area	maxDets	AP_{11}	AR
0.50:0.95	All	20	0.747	-
0.50	All	20	0.970	-
0.75	All	20	0.903	-
0.50:0.95	Small	20	−1.000	-
0.50:0.95	Medium	20	0.709	-
0.50:0.95	Large	20	0.759	-
0.50:0.95	All	1	-	0.581
0.50:0.95	All	10	-	0.780
0.50:0.95	All	20	-	0.780
0.50:0.95	Small	20	-	−1.000
0.50:0.95	Medium	20	-	0.735
0.50:0.95	Large	20	-	0.787

4.4. Extraction Results of Catch Points

Figure 9 shows the experimental results of the dish image extraction from the dishes' detection to the catch points calculation. (a) and (d) are the taken dish images. (b) is the detection result of (a). The result demonstrates that "Towel" has not been recognized, and "Waster-paper" in "Cup" and "Spoon" in "Square-bowl" have low detection accuracy, mainly because the two dishes put together affects the detection accuracy. (c) is the catch point extraction image of (a), the catch point of "Chopsticks-one" is not positioned at the center, and the catch point of "Towel-dish" is slightly off the edge. (e) is the detection result of (d), and all the detection results are above 95%; (f) is the catch point of (d) to extract the image, and the catch point of "Wine-cup" on the left has an error in extraction because the class is a transparent object. In the process of image processing, the lower edge of the dish is fitted into a circle, which causes the catch point to shift to the lower edge.

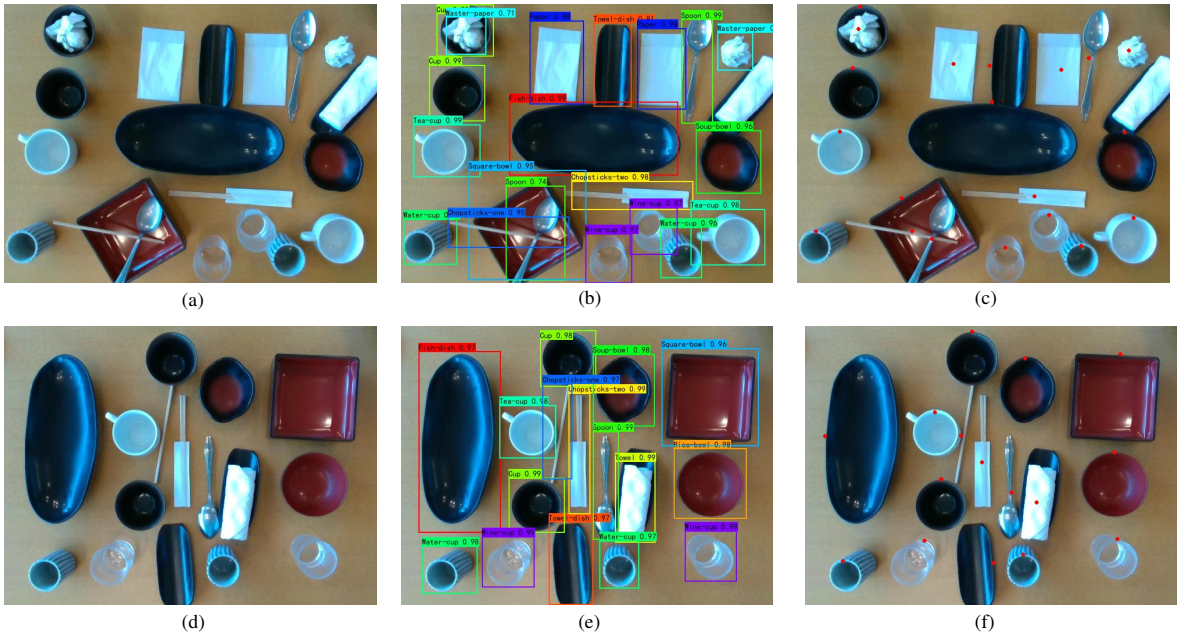


Figure 9. Example of dish detection and catch point extraction. (**a**) Is one of Dish's images, (**b**) is the detection result of (**a**), and (**c**) is the catch point extraction image of (**a**). (**d**) Is one of Dish's images, (**e**) is the detection result of (**d**), and (**f**) is the catch point extraction image of (**d**).

The robotic fingers that catch the dish are pneumatic fingers, which expand the fingers by pneumatic force to clamp the dish. The catch point does not affect the catch of the dish within a certain error. During the field testing, the method achieves high accuracy, proving the effectiveness of the catch point extraction method proposed in this paper.

5. Discussion and Future Work

When the confidence score is set to 0.5 and the IoU is set to 0.5, the mAP value of YOLO-GD is 97.38%, which is 3.41% higher than YOLOv4. The weight of YOLO-GD is only 45.80 MB, the parameter is 11.17 M, and the $FLOPs$ is 6.61 G, which is 1/5 of YOLOv4. After TensorRT's $FP16$ quantization and deployment on the Jetson Nano, the inference time per image is 32.75 ms and reaches 30.53 FPS, and the inference speed is 8.4 times higher than without quantization. Moreover, the mAP with quantization is 97.42%, which is 0.04% higher than without quantization. In addition, the power consumption of the Jetson Nano is only 5–10 W, which meets the requirements of low power consumption for robots.

In the detection process of YOLO-GD, some dishes could not be recognized effectively because of the variety of dishes and the intersection of placement. For example, "Towel" in Figure 9b is not recognized, failing to extract its catch point. In the process of extracting the catch points, the location of the dish contour and other information are misjudged in the image processing of the catch point extraction because the environment such as the light has a significant impact on the image. For example, the "Wine-cup" in Figure 9f locates the catch point at the bottom contour position.

In future work, the YOLO-GD model is further compressed using pruning techniques to make it more lightweight. The catch point extraction method is optimized to ensure a more accurate extraction of the catch points. The Intel RealSense Depth Camera D435's depth images and video images are used to capture the height information of the dish catch point and feed the detailed information of the catch point in a three-dimensional space to the robot. In addition, we will design new algorithms to optimize the order of the catch dishes.

6. Conclusions

This article introduces a deep learning-based object detection algorithm YOLO-GD for empty-dish recycling robots. The object detection model algorithm YOLO-GD is based on YOLOv4. By replacing the backbone structure with a lightweight Ghost Net, as well as replacing the traditional convolution with depthwise separable convolution and pointwise convolution in the stage of feature fusion and result prediction, a lightweight one-stage detection model YOLO-GD is formed. According to the detection results of the dish, a different image processing is performed to extract the catch points. The coordinate information of the catch point is transmitted to the robot, and the robotic arm is used to catch the dish. To improve the detection speed, TensorRT is used to quantify the object detection model YOLO-GD as $FP16$ and is deployed on the robot control system, Jetson Nano. The experimental results demonstrate that the object detection algorithm is only 1/5 of YOLOv4, and the mAP value is 97.38%, which is 3.41% higher than the 93.97% of YOLOv4. After YOLO-GD quantization, the inference time per image is decreased from 207.92 ms to 32.75 ms, and the mAP is increased from 97.38% to 97.42%. Although there is a certain error in the extraction of the catch point coordinates, it meets the error requirements of the robotic finger. In summary, the system can effectively detect the dish and extract the catch point, which has far-reaching significance for the empty-dish recycling robot.

Author Contributions: Conceptualization, X.Y. and L.M.; software, X.Y. and H.L.; validation, X.Y. and L.M.; formal analysis, X.Y. and L.M.; investigation, X.Y. and L.M.; resources, X.Y. and L.M.; data curation, X.Y. and L.M.; writing—original draft preparation, X.Y. and L.M.; writing—review and editing, X.Y. and L.M.; visualization, X.Y.; supervision, L.M.; project administration, L.M.; funding acquisition, L.M., M.S., and S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Cabinet Office (CAO) and the Cross-ministerial Strategic Innovation Promotion Program (SIP), "An intelligent knowledge processing infrastructure, integrating physical and virtual domains" (funding agency: NEDO).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: For the "Dish-20" dataset connection, please refer to http://www.ihpc.se.ritsumei.ac.jp/obidataset.html; accessed on 28 March 2022.

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

References

1. Dotoli, M.; Fay, A.; Miśkowicz, M.; Seatzu, C. An overview of current technologies and emerging trends in factory automation. *Int. J. Prod. Res.* **2019**, *57*, 5047–5067. [CrossRef]
2. Haase, C.B.; Bearman, M.; Brodersen, J.; Hoeyer, K.; Risor, T. 'You should see a doctor', said the robot: Reflections on a digital diagnostic device in a pandemic age. *Scand. J. Public Health* **2021**, *49*, 33–36. [CrossRef] [PubMed]
3. Yang, Z.; Ji, X.; Tang, X.; Li, X. Intelligent search and rescue robot design based on KANO model and TRIZ theory. In Proceedings of the 2021 2nd International Conference on Intelligent Design (ICID), Xi'an, China, 19 October 2021; pp. 371–377. [CrossRef]
4. Fukuzawa, Y.; Wang, Z.; Mori, Y.; Kawamura, S. A Robotic System Capable of Recognition, Grasping, and Suction for Dishwashing Automation. In Proceedings of the 2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Shanghai, China, 26–28 November 2021; pp. 369–374. [CrossRef]
5. Pereira, D.; Bozzato, A.; Dario, P.; Ciuti, G. Towards Foodservice Robotics: A Taxonomy of Actions of Foodservice Workers and a Critical Review of Supportive Technology. *IEEE Trans. Autom. Sci. Eng.* **2022**, 1–39. [CrossRef]
6. Yin, J.; Apuroop, K.G.S.; Tamilselvam, Y.K.; Mohan, R.E.; Ramalingam, B.; Le, A.V. Table Cleaning Task by Human Support Robot Using Deep Learning Technique. *Sensors* **2020**, *20*, 1698. [CrossRef] [PubMed]
7. Candeias, A.; Rhodes, T.; Marques, M.; ao Costeira, J.P.; Veloso, M. Vision Augmented Robot Feeding. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
8. Chen, J.; Bai, T. SAANet: Spatial adaptive alignment network for object detection in automatic driving. *Image Vis. Comput.* **2020**, *94*, 103873. [CrossRef]
9. Li, X.; Qin, Y.; Wang, F.; Guo, F.; Yeow, J.T.W. Pitaya detection in orchards using the MobileNet-YOLO model. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–30 July 2020; pp. 6274–6278. [CrossRef]

10. Yue, X.; Lyu, B.; Li, H.; Fujikawa, Y.; Meng, L. Deep Learning and Image Processing Combined Organization of Shirakawa's Hand-Notated Documents on OBI Research. In Proceedings of the 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), Xiamen, China, 3–5 December 2021; Volume 1, pp. 1–6. [CrossRef]
11. Fujikawa, Y.; Li, H.; Yue, X.; Prabhu G.A.; Meng, L. Recognition of Oracle Bone Inscriptions by using Two Deep Learning Models. *Int. J. Digit. Humanit.* **2022**. [CrossRef]
12. Li, H.; Wang, Z.; Yue, X.; Wang, W.; Tomiyama, H.; Meng, L.A. Comprehensive Analysis of Low-Impact Computations in Deep Learning Workloads. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 385–390.
13. Li, H.; Yue, X.; Wang, Z.; Wang, W.; Chai, Z.; Tomiyama, H.; Meng, L. Optimizing the deep neural networks by layer-wise refined pruning and the acceleration on FPGA. In *Computational Intelligence and Neuroscience*; Hindawi: London, UK, 2022.
14. Li, G.; Wang, J.; Shen, H.W.; Chen, K.; Shan, G.; Lu, Z. CNNPruner: Pruning Convolutional Neural Networks with Visual Analytics. *IEEE Trans. Vis. Comput. Graph.* **2021**, *27*, 1364–1373. [CrossRef]
15. Wang, X.; Yue, X.; Li, H.; Meng, L. A high-efficiency dirty-egg detection system based on YOLOv4 and TensorRT. In Proceedings of the 2021 International Conference on Advanced Mechatronic Systems (ICAMechS), Tokyo, Japan, 9–12 December 2021; pp. 75–80. [CrossRef]
16. Wang, L.; Yoon, K.J. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**. [CrossRef]
17. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 July 2018.
18. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
19. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features From Cheap Operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 13–19 June 2020.
20. Zhu, D.; Seki, H.; Tsuji, T.; Hiramitsu, T. Mechanism and Design of Tableware Tidying-up Robot for Self-Service Restaurant. In Proceedings of the 2021 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 8–11 August 2021; pp. 825–830. [CrossRef]
21. Kawamura, S.; Sudani, M.; Deng, M.; Noge, Y.; Wakimoto, S. Modeling and System Integration for a Thin Pneumatic Rubber 3-DOF Actuator. *Actuators* **2019**, *8*, 32. [CrossRef]
22. Kinugawa, J.; Suzuki, H.; Terayama, J.; Kosuge, K. Underactuated robotic hand for a fully automatic dishwasher based on grasp stability analysis. *Adv. Robot.* **2022**, *36*, 167–181. [CrossRef]
23. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2015**, arXiv:1506.02640.
24. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. *arXiv* **2015**, arXiv:1512.02325.
25. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002.
26. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv* **2013**, arXiv:1311.2524.
27. Girshick, R.B. Fast R-CNN. *arXiv* **2015**, arXiv:1504.08083.
28. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2015**, arXiv:1506.01497.
29. Bochkovskiy, A.; Wang, C.; Liao, H.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
30. Li, Y.; Li, S.; Du, H.; Chen, L.; Zhang, D.; Li, Y. YOLO-ACN: Focusing on Small Target and Occluded Object Detection. *IEEE Access* **2020**, *8*, 227288–227303. [CrossRef]
31. Cao, Z.; Liao, T.; Song, W.; Chen, Z.; Li, C. Detecting the shuttlecock for a badminton robot: A YOLO based approach. *Expert Syst. Appl.* **2021**, *164*, 113833. [CrossRef]
32. Zhang, Y.; Lee, C.; Hsieh, J.; Fan, K. CSL-YOLO: A New Lightweight Object Detection System for Edge Computing. *arXiv* **2021**, arXiv:2107.04829.
33. Wang, G.; Ding, H.; Yang, Z.; Li, B.; Wang, Y.; Bao, L. TRC-YOLO: A real-time detection method for lightweight targets based on mobile devices. *IET Comput. Vis.* **2022**, *16*, 126–142. [CrossRef]
34. Guan, L.; Jia, L.; Xie, Z.; Yin, C. A Lightweight Framework for Obstacle Detection in the Railway Image based on Fast Region Proposal and Improved YOLO-tiny Network. *IEEE Trans. Instrum. Meas.* **2022**. [CrossRef]
35. Hirose, S.; Wada, N.; Katto, J.; Sun, H. Research and examination on implementation of super-resolution models using deep learning with INT8 precision. In Proceedings of the 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Korea, 21–24 February 2022; pp. 133–137. [CrossRef]
36. Jeong, E.; Kim, J.; Tan, S.; Lee, J.; Ha, S. Deep Learning Inference Parallelization on Heterogeneous Processors With TensorRT. *IEEE Embed. Syst. Lett.* **2022**, *14*, 15–18. [CrossRef]
37. Jeong, E.; Kim, J.; Ha, S. TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards. *ACM Trans. Embed. Comput. Syst.* **2022**. [CrossRef]

38. Stäcker, L.; Fei, J.; Heidenreich, P.; Bonarens, F.; Rambach, J.; Stricker, D.; Stiller, C. Deployment of Deep Neural Networks for Object Detection on Edge AI Devices With Runtime Optimization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Virtual, 11–17 October 2021; pp. 1015–1022.
39. Wang, Y.; Yu, P. A Fast Intrusion Detection Method for High-Speed Railway Clearance Based on Low-Cost Embedded GPUs. *Sensors* **2021**, *21*, 7279. [CrossRef] [PubMed]
40. Zhang, C.; Cao, Q.; Jiang, H.; Zhang, W.; Li, J.; Yao, J. A Fast Filtering Mechanism to Improve Efficiency of Large-Scale Video Analytics. *IEEE Trans. Comput.* **2020**, *69*, 914–928. [CrossRef]
41. Shao, Z.; Wang, L.; Wang, Z.; Du, W.; Wu, W. Saliency-Aware Convolution Neural Network for Ship Detection in Surveillance Video. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 781–794. [CrossRef]
42. Yue, X.; Li, H.; Shimizu, M.; Kawamura, S.; Meng, L. Deep Learning-based Real-time Object Detection for Empty-Dish Recycling Robot. In Proceedings of the 13th Asian Control Conference (ASCC 2022), Jeju Island, Korea, 4–7 May 2022.
43. Liu, J.; Cong, W.; Li, H. Vehicle Detection Method Based on GhostNet-SSD. In Proceedings of the 2020 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), Zhangjiajie, China, 18–19 July 2020; pp. 200–203. [CrossRef]
44. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 14–19 June 2020.
45. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141. [CrossRef]
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *arXiv* **2014**, arXiv:1406.4729.
47. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. *arXiv* **2018**, arXiv:1803.01534.
48. Ding, L.; Goshtasby, A. On the Canny edge detector. *Pattern Recognit.* **2001**, *34*, 721–725. [CrossRef]
49. Fu, Y.; Zeng, H.; Ma, L.; Ni, Z.; Zhu, J.; Ma, K.K. Screen Content Image Quality Assessment Using Multi-Scale Difference of Gaussian. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 2428–2432. [CrossRef]
50. Yue, X.; Lyu, B.; Li, H.; Meng, L.; Furumoto, K. Real-time medicine packet recognition system in dispensing medicines for the elderly. *Meas. Sens.* **2021**, *18*, 100072. [CrossRef]
51. Meng, L. Recognition of Oracle Bone Inscriptions by Extracting Line Features on Image Processing. In Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods—Volume 1: ICPRAM,. INSTICC, SciTePress, Porto, Portugal, 24–26 February 2017; pp. 606–611. [CrossRef]
52. Shafi, O.; Rai, C.; Sen, R.; Ananthanarayanan, G. Demystifying TensorRT: Characterizing Neural Network Inference Engine on Nvidia Edge Devices. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*; IEEE Computer Society: Los Alamitos, CA, USA, 2021; pp. 226–237. [CrossRef]
53. Mamdouh, N.; Khattab, A. YOLO-Based Deep Learning Framework for Olive Fruit Fly Detection and Counting. *IEEE Access* **2021**, *9*, 84252–84262. [CrossRef]
54. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* **2021**, *109*, 43–76. [CrossRef]
55. Xu, Q.; Lin, R.; Yue, H.; Huang, H.; Yang, Y.; Yao, Z. Research on Small Target Detection in Driving Scenarios Based on Improved Yolo Network. *IEEE Access* **2020**, *8*, 27574–27583. [CrossRef]
56. Kumar, P.; Batchu, S.; Swamy S.N.; Kota, S.R. Real-Time Concrete Damage Detection Using Deep Learning for High Rise Structures. *IEEE Access* **2021**, *9*, 112312–112331. [CrossRef]
57. Padilla, R.; Netto, S.L.; da Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Rio de Janeiro, Brazil, 1–3 July 2020; pp. 237–242. [CrossRef]