An Ultralightweight Object Detection Network for Empty-Dish Recycling Robots

Xuebin Yue, *Student Member, IEEE*, Hengyi Li, *Student Member, IEEE*, and Lin Meng, *Member, IEEE*

***Abstract*— The emergence of empty-dish recycling robots has alleviated problems, such as labor shortages, caused by an aging population. The detection and grasping of dishes play a crucial role in empty-dish recycling robots. However, due to the limited resources of edge devices, traditional object detection models require more space to store parameters and much computational overhead, limiting the development of empty-dish recycling robots. Therefore, this article proposes an ultralightweight dish detection model YOLO-GS for an empty-dish recycling robot. We use the modified CSPDarknet as the backbone structure and design an ultralightweight neck structure for efficient feature fusion. Meanwhile, we design a lightweight head structure for object classification and bounding box coordinate regression by combining ghost shuffle convolution (GSConv2D) and the anchor-free method. For the empty-dish recycling robot to grasp the dishes, we design a dish grasp point extraction algorithm using image processing. Finally, TensorRT is used to optimize and accelerate the model for efficient and intelligent detection of dishes on the NVIDIA Jetson Xavier NX. The experimental results show that YOLO-GS achieves 99.380% mean average precision (mAP) with a parameter amount of 0.606 M. The inference speed of the TensorRT-optimized YOLO-GS algorithm reaches 31.371 FPS, which meets the needs of real-time dish detection by the empty-dish recycling robot. The image of the empty-dish recycling robot demo is available at https://www.youtube.com/watch?v=pCBo1nzm3qU&t=22s.**

***Index Terms*— Empty-dish recycling robot, grasp point extraction, object detection, quantification and deployment.**

I. Introduction

With the aging of the global population and the relative reduction of the total labor force, many countries have experienced serious labor shortages. The labor shortage and the increasing labor cost are seriously restricting the development of the food service industry. With the rapid development of artificial intelligence technology, intelligent food service robots relying on those technologies emerge as the times require solving the labor shortage problem and the increasing labor cost [1]. This article aims to study the empty-dish recycling robot in intelligent food service robots.

In the food service industry, empty-dish recycling robots need to solve indoor location, dish detection, dish grasping, recycling, walk-off, and so on. Among them, the effective detection and grasp of dishes scattered on the desktop are the critical problems of the empty-dish recycling robot. With the development of convolution neural networks (CNN), computer vision-based object detection algorithms have made breakthroughs in the field of dish detection [2], [3]. Yue et al. [1] use traditional YOLOv4 to detect dishes and achieve more than 96.00% high accuracy on precision, recall, and F1 values. Wang et al. [4] use traditional YOLOv3 to detect 16 classes of dishes and achieve a mean average precision (mAP) of 96.40%. Yue et al. [5] propose a dish grasp point extraction algorithm based on image processing technology, which can extract the grasp point coordinates of dishes in a 2-D plane.

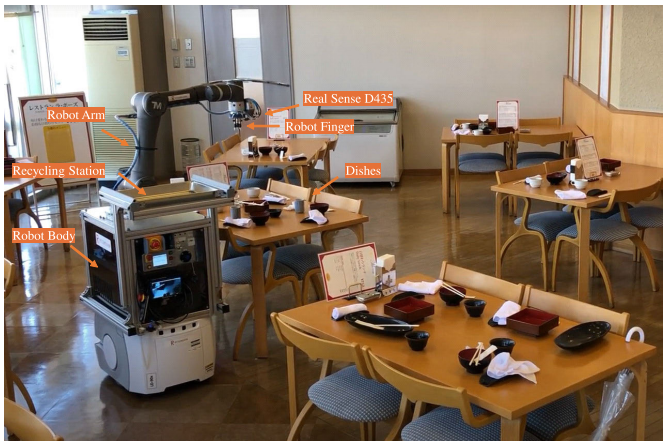


Fig. 1. Illustration of empty-dish recycling robot.

The empty-dish recycling robot needs to have high mobility. Therefore, edge devices are selected as the control and inference platform. Due to the limited resources of edge devices, traditional object detection models take a long time to load models on edge devices and require ample space to store models. Therefore, designing an object detection model suitable for running on an edge platform and meeting the requirements of real-time and accurate dish detection have become an urgent problem to be solved.

To solve the abovementioned problems, we develop an ultralightweight object detection model YOLO-GS for the empty-dish recycling robot, where GS represents the model using ghost shuffle convolution (GSConv2D). GSConv2D is to reorder the output of the Ghost module (a structure that generates a large number of feature maps with a few computations) through channel shuffle, thereby preserving the hidden connections between each channel. We choose a lightweight CSPDarknet network as the backbone structure and adjust the structure to reduce the parameters while ensuring feature

Manuscript received 16 October 2022; revised 28 December 2022; accepted 17 January 2023. Date of publication 31 January 2023; date of current version 10 February 2023. 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—through New Energy and Industrial Technology Development Organization (NEDO). The Associate Editor coordinating the review process was Jae-Ho Han. *(Corresponding author: Lin Meng.)*

The authors are with the Department of Electronic and Computer Engineering, Ritsumeikan University, Kusatsu 525-8577, Japan (e-mail: gr0468xp@ed.ritsumei.ac.jp; gr0468kx@ed.ritsumei.ac.jp; menglin@fc.ritsumei.ac.jp).

Digital Object Identifier 10.1109/TIM.2023.3241078

extraction capabilities. We propose an ultralightweight neck structure using GSConv2D to reduce the number of parameters and calculations while enhancing features' interaction ability and improving feature fusion efficiency. Simultaneously, we design object classification and bounding box coordinate regression as two parallel branches, reduce the number of parameters and floating point operations (FLOPs), and form a lightweight head structure. Ultimately, we design an image processing-based dish grasp point extraction algorithm to grasp the dishes. In addition, we use TensorRT to quantify YOLO-GS in floating-point 16-bit and deploy it on the Jetson Xavier NX, an empty-dish recycling robot control platform.

The contributions of this work are summarized as follows.

1) We design an ultralightweight neck structure for efficient feature fusion with minimal parameters and FLOPs.
2) We propose an ultralightweight dish detection model YOLO-GS for the empty-dish recycling robot. The model has only 0.606 *M* parameters, and the mean AP reaches 99.380%. The model requires very little storage space, and the load time is significantly reduced.
3) We design a dish grasp point extraction algorithm to extract the grasp points of the detected dishes through image processing and obtain the dishes' grasp point coordinate information in the 3-D space of the empty-dish recycling robot.
4) We quantify YOLO-GS with 16-bit floating-point nondestructive precision and deploy it on the control system Jetson Xavier NX of the empty-dish recycling robot.

The remainder of this article is arranged as follows. Section II reviews the previous overview of empty-dish recycling robots, lightweight object detection models, and object detection applications in the edge platform. In Section III, we describe the proposed method in detail. The experiments, discussions, and feature work are presented in Section IV. Finally, Section V concludes this article.

II. Related Work

In this section, we first summarize the recent progress of empty-dish recycling robots and then review the literature on lightweight object detection models, model quantification, and deployment on edge platforms.

A. Overview of Empty-Dish Recycling Robot

In the intelligent service robot, the empty-dish recycling robot needs to solve tasks, such as indoor location, dish detection, dish grasping, and so on, which have high complexity. Therefore, robots are mainly used for automatic food serving, table cleaning, and so on. There is little research on empty-dish recycling robots [6].

Yin et al. [7] propose a table cleaning and inspection method using a human support robot (HSR) through a lightweight deep convolutional neural network (DCNN) to recognize the food litter on top of the table and then generate cleaning paths based on the detection of food litter to perform cleaning operations. Yue et al. [1] apply YOLOv4 to the dish detection of the empty-dish recycling robot, quantify the YOLOv4 model through TensorRT, and deploy it on Jetson Nano. However, the final inference speed of the model is only 2.3 FPS. Yue et al. [5] propose a lightweight object detection model YOLO-GD, which is used to detect dishes in images, such as cups, chopsticks, bowls, towels, and so on, and based on the method of image processing, the grasp point coordinate method for extracting different types of dishes are designed. Significantly, the dish detection model has only 11.17 M parameters, and the detected mAP reaches 97.42%.

B. Lightweight Object Detection Model

Object detection algorithms based on deep learning are mainly divided into two categories. The first is the two-stage object detection algorithm based on candidate regions [8], including R-CNN [9], fast R-CNN [10], faster R-CNN [11], and so on. The other is the one-stage object detection algorithm based on regression problems, including YOLO [12], [13], SSD [14], retina net [15], and so on.

In many real-world applications, object detection must be performed in a timely and power-saving manner with computational resource constraints. Many other vision tasks have built lightweight models using methods, such as weight quantization [16], [17], network compression [18], computationally efficient architecture design [19], [20], [21], and so on. For some vision tasks, lightweight networks aim to achieve the best tradeoff between accuracy and efficiency, showing their superiority by reducing the model size and FLOPs with a little performance drop [22].

Meanwhile, several studies have proposed object detection models based on lightweight backbone structures. Guan et al. [23] propose a lightweight three-stage detection framework consisting of a coarse region proposal (CRP) module, a lightweight railway obstacle detection network (RODNet), and a postprocessing stage for recognizing obstacles in a single-railway image. Fan et al. [24] propose a lightweight meter recognition method that combines deep learning and traditional computer vision techniques for an automatic meter reading. Cai et al. [25] propose a one-stage object detection framework based on YOLOv4 for object detection in autonomous driving. At the same time, an optimization network pruning algorithm is proposed to solve the problem that the computing resources of the vehicle-mounted computing platform are limited and cannot meet the real-time performance.

C. Application of Object Detection in Edge Platform

High-level graphics processing units (GPUs) are commonly used in high-performance deep learning applications. However, building a high-performance platform is expensive in terms of cost and power consumption. In real-world application scenarios, the object detection network needs to be deployed on the edge platform. Due to the limited resources of the edge platform, quantification deployment of the object detection network becomes a key factor [26], [27].

Wang et al. [4] propose a YOLOv3-based dish detection network on an FPGA platform, and through different sparse training and pruning methods, the model size is reduced from 62 to 12 MB. Koubaa et al. [28] present a real-world

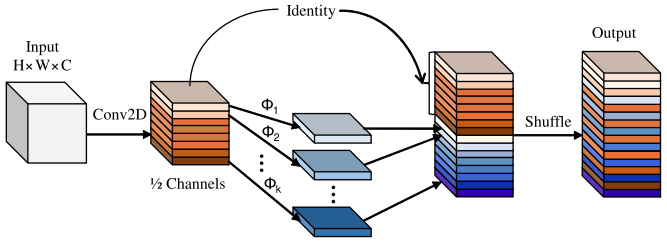


Fig. 2. Structure of the GSConv2D module.

case study of deploying a face recognition application using the MTCNN detector and FaceNet recognizer and demonstrate that TensorRT optimization provides the fastest execution on edge devices. Liu et al. [16] propose a fast and accurate power line edge intelligent detection method called RepYOLO by using the C++ language combined with TensorRT that is employed to optimize and accelerate the model on the NVIDIA Jetson Xavier NX embedded platform, fulfilling efficient power line edge intelligent detection. Tu et al. [17] propose a real-time defect detection method for tracking components based on an improved lightweight instance segmentation network, using the TensorRT inference framework to accelerate the defect detection network and realize edge platform deployment.

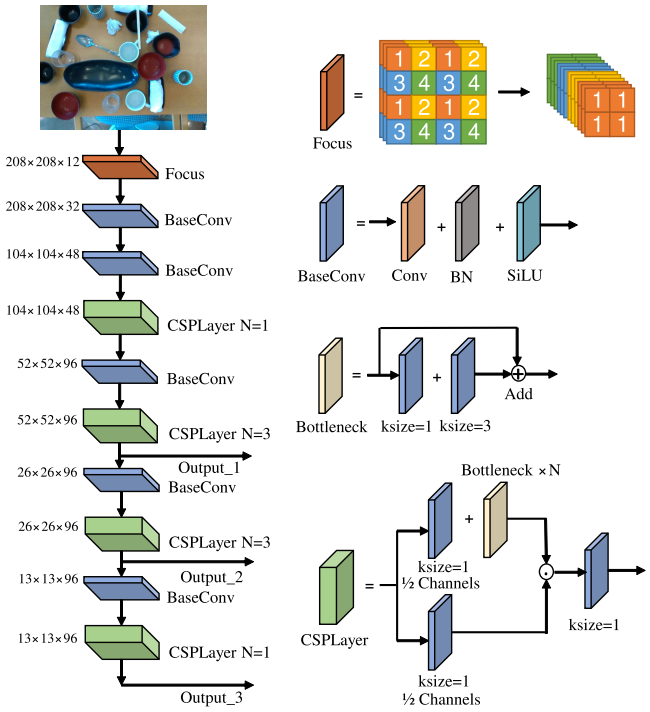


Fig. 3. Illustration of backbone network structure.

III. Methodology

A. Overview of Empty-Dish Recycling Robot

Fig. 1 shows an illustration of the empty-dish recycling robot at work. The robot consists of a robotic body, arms, fingers, and a camera that collects information about the dishes. The workflow of the empty-dish recycling robot is as follows: 1) the robot determines the table information that needs to receive dishes in the restaurant and uses the sensor and the drive system to arrive accurately at the empty-dish recycling location based on the stored location information. 2) The robot loads the ultralightweight dish detection model YOLO-GS and moves the camera (Intel RealSense D435) to the top of the table, waiting for the camera to take images. 3) Use the camera to take images, detect the type and position of dishes in the image through the loaded model, and calculate the different types of dishes' grasp points through the proposed extraction algorithm. 4) Send the obtained coordinates of the dish grasp point to the robotic arm control system, calculate the rotation angle of each joint through the inverse kinematics equation, move the robotic arm to the grasp point position, and grasp the dish with the fingers. 5) Put the dish into the recycling station and repeat steps 4 and 5 until all the dishes are recycled.

B. Overview of the YOLO-GS Framework

We aim to build an ultralightweight and efficient object detection network for the dish detection task of the empty-dish recycling robot. Therefore, we consider many factors, such as convolution method, lightweight backbone, lightweight feature fusion structure, computational efficiency, computational cost-effectiveness, and so on, and design an ultralightweight object detection model YOLO-GS.

The YOLO-GS network structure is mainly composed of three parts, namely, backbone, neck, and head. The backbone network is used for feature extraction, and the output is three effective feature layers. The neck is a feature fusion network that fuses features of different scales output by the backbone network. Head is a prediction network that predicts objects and bounding boxes on the feature map output by the neck network. Simultaneously, YOLO-GS adopts the mosaic data augmentation method to splice images through random scaling, cropping, and arrangement, which enriches the diversity of data and reduces the use of GPU memory.

1) Ghost Shuffle Convolution: In the conventional CNN model, many feature maps are similar and have more redundancy. Han et al. [20] propose the ghost module to generate a large number of feature maps with only a few computations (cheap operations). As shown in Fig. 2, in the ghost module, conventional convolution is first used to generate partial feature maps, then the cheap operation is used to generate redundant features on the generated feature maps, and finally, all the feature maps are concatenated. The dense convolution computation preserves the hidden connections between each channel, while the cheap operation severs these connections completely. Therefore, in this work, the output of the ghost module is reordered by the channel shuffle [19], which improves the flow of global information.

The computational cost of GSConv2D is only 60%–70% of standard convolution, but the contribution to the model learning ability is comparable to standard convolution [29]. Fig. 2 is a schematic of GSConv2D. Specifically, GSConv2D uses the "halved" convolution operation to retain the interaction information between channels. The features generated by convolution perform simple linear operations (cheap operations)

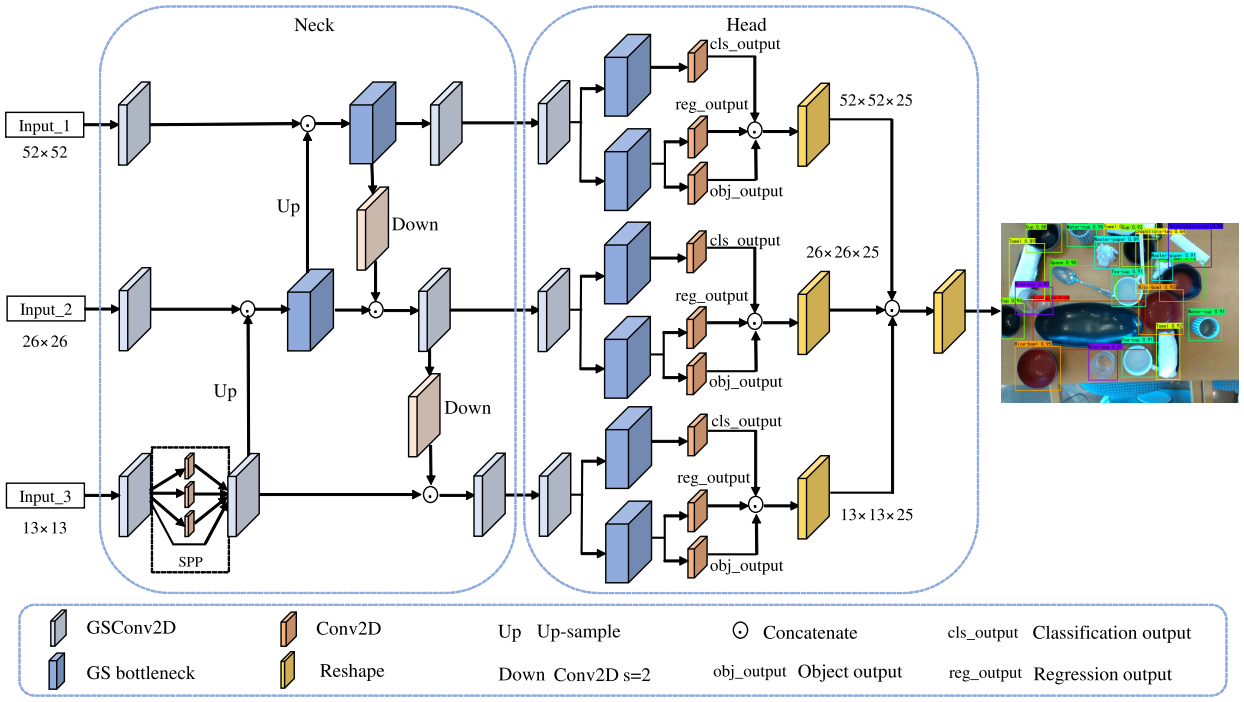


Fig. 4. Overview architecture of ultralightweight detection of neck and head models. The GS bottleneck is a stacked structure of two GSConv2D and adds the input to the output. Up-sample is to multiply the width and height of the data by using the nearest neighbor sampling method.

to generate more similar features maps, that is,

$$\text{GSConv2D} = \mathcal{S}(\mathcal{F}^{1\times1}(F) \odot \Phi(\mathcal{F}^{1\times1}(F))) \tag{1}$$

where $\odot$ represents the concatenate operation, F denotes the feature map, $\mathcal{F}^{1\times1}(\cdot)$ is the stacking structure of 1×1 convolution operation for half of the output channel, and the batch normalization (BN) operation and activation function are nonlinear operations of the sigmoid linear unit (SiLU). $\Phi(\cdot)$ is the linear operation for generating a feature map, $\mathcal{S}(\cdot)$ represents the channel shuffle operation.

2) Backbone: CSPDarknet in YOLOX-tiny [30] is an excellent feature extraction network that satisfies most feature extraction tasks for dish detection scenarios. Since the features of the dish object are relatively simple, we adjust the structure and parameters based on the CSPDarknet network to reduce the number of parameters while ensuring feature extraction capability. The structure of the backbone network is shown in Fig. 3. At the input of the backbone, the image is downsampled using focus without losing feature information. It uses slice operation to split the high-resolution feature map into multiple low-resolution feature maps. The backbone network employs the residual structure, and residual skip connections retard the gradient vanishing problem. The CSPNet [31] structure produces richer gradient combination information while requiring less calculation. The SiLU activation function is used in the backbone network's nonlinear expression. As seen (2), SiLU has no upper and lower bounds, smoothness, and nonmonotonicity, which plays an essential role in optimization and generalization

$$\text{SiLU}(x) = \frac{x}{1 + e^{-x}}. \tag{2}$$

3) Ultralightweight Detection of Neck and Head Models: The features have been described in the backbone structure, and when these feature maps reach the neck, they are already slender enough (the channel dimension reaches the maximum, and the width and height dimensions reach the minimum) and no longer need to be transformed. Therefore, using GSConv2D in the neck structure can better describe the features than in the backbone. The low level in the backbone network has less semantic information but accurate object locations. The high level has richer semantic information but coarse object locations. We aim to fuse low-level and high-level features using fewer parameters efficiently. Through the research on FPN [32] structure, PANet [33] structure, and other methods, we design an ultralightweight Neck structure, as shown in Fig. 4.

Through GSConv2D and spatial pyramid pooling (SPP), the low-level features improve the scale invariance of the image, enrich the expression ability, and expand the receptive field. SPP can be expressed as

$$\text{SPP} = \mathcal{C}(f^{5\times5}\text{MaxP}(F) \odot f^{9\times9}\text{MaxP}(F) \odot f^{13\times13}\text{MaxP}(F) \odot F). \tag{3}$$

Among them, $\odot$ represents the concatenate operation, F denotes feature map, $f^{k\times k}$ means $k \times k$ filter, MaxP means max pooling operation, $\mathcal{C}(\cdot)$ means concatenate operation.

YOLOv3 [12], v4 [13], and v5 all follow the original anchor-based method, but there are many known problems with the anchor mechanism. First, to achieve optimal detection performance, cluster analysis is required before training to determine a set of optimal anchors. Second, the anchor mechanism increases the complexity of the detection head and the number of predictions per image. The anchor-free method proposed in YOLOX [30] does not need to preset anchors but only needs to regress the object center point and the width and height of feature maps with different

Algorithm 1 Extraction of Grasp Points

Input: The object classes and coordinates. The depth image of dishes. The number of dishes in the image (Num_{dish}).

Output: Grasp point coordinates of the object in 3D space.

```
for j ← 1 to Num_dish do
    x_0, y_0, x_1, y_1 ← Coordinates
    img ← image[y_0 : y_1, x_0 : x_1]
    if type_j = Circle then
        x_c, y_c, r ← HoughCircle(img)
        x ← x_0 + x_c
        y ← y_0 + y_c − r
    else if type_j = Square then
        Lines ← HoughLinesP(img)
        for i ← 1 to len(Lines) − 1 do
            for k ← 1 to len(Lines) do
                Tht ← GetCrossAngle(Lines[i], Lines[k]) if Tht > 85 and Tht < 95 then
                    Line.append(Lines[i])
        for i ← 1 to len(Line) do
            for k ← 1 to len(Line) do
                x,y ← CrossPoint(Lines[i], Lines[k])
                Point.append([x,y])
        B_P ← boxPoint(minAreaRect(Point))
        x ← x_0 + (B_P [1] [0] + B_P [2][0])/2
        y ← y_0 + (B_P [1] [1] + B_P [2] [1])/2
    else if type_j = Polygon then
        contous, hie ← findContours(img)
        Area ← contourArea(contous)
        rect ← minAreaRect(contous(argmax(Area)))
        x ← x_0 + rect[0][0]
        y ← y_0 + rect[0] [1]
    else if type_j = Ellipse then
        contous, hie ← findContours(img)
        Area ← contourArea(contous)
        x_e, y_e, a, b, agl ← fitEllipse(contous(argmax(Area)))
        x ← x_0 + x_e − b × cos(agl)/2
        y ← y_0 + y_e − b × sin(agl)/2
    else
        Lines ← HoughLinesP(img)
        a,b,c ← Lines.shape
        for i ← 1 to a do
            Point[i][0] ← Lines[i][0][0]
            Point[i] [1] ← Lines[i][0] [1]
        polygon ← Polygon(Point).convex_hull
        coor ← polygon.centroid.coords[0]
        x ← x_0 + coor[0]
        y ← y_0 + coor [1]
    h ← Get_Distance(x,y)
    return x,y,h
```

scales, which significantly reduces the time-consuming and required computing power. Therefore, we use the anchor-free method for object classification and bounding box coordinate regression.

We adjust each position of the head into two outputs, one for predicting the classes of objects in each feature point. The other is used to predict the regression parameters of each feature point and determine whether each feature point contains an object. This method reduces the number of parameters and FLOPs, alleviates the imbalance of positive and negative samples, and avoids the adjustment of anchor parameters.

4) Loss Function: The loss function is the difference measurement between the predicted value and the true value. The loss of the network, like the prediction result of the network, is also composed of three parts, namely, the Cls part, the Obj part, and the Reg part, which can be formulated as

$$\text{LOSS} = \mathcal{L}_{\text{Cls}} + \mathcal{L}_{\text{Obj}} + \mathcal{L}_{\text{Reg}}. \tag{4}$$

The Cls part is the class of objects contained in the feature points, and the binary cross-entropy (BCE) loss is calculated according to the class of the real-bounding box and the class prediction result of feature points as the loss of the Cls part. The Obj part evaluates whether the feature points contain objects and calculates the BCE loss using the positive and negative samples and the prediction results of whether the feature points contain objects as the loss of the Obj part. The BCE loss is calculated as follows:

$$\mathcal{L}_{\text{BCE}} = -\Sigma_{i=1}^{n}(t_i \log P(y_i) + (1 - t_i)\log(1 - P(y_i))). \tag{5}$$

Among them, n represents the total number of samples, $t_i \in \{0, 1\}$ is the binary label, and y_i is the probability of the label value.

The Reg part is used to predict the regression parameters of the feature points and calculate the CIoU loss using the real-bounding box and the predicted bounding box as the loss of the Reg part. Reg loss is expressed as follows:

$$\mathcal{L}_{\text{Reg}} = \Sigma_{i=1}^{n}\mathcal{L}_{\text{CIoU}} \tag{6}$$

$$\text{CIoU} = \text{IoU} - \frac{\rho^2(b, b^{\text{gt}})}{c^2} - \eth\nu \tag{7}$$

$$\mathcal{L}_{\text{CIoU}} = 1 - \text{CIoU} = 1 - \text{IoU} + \frac{\rho^2(b, b^{\text{gt}})}{c^2} + \eth\nu \tag{8}$$

$$\nu = \frac{4}{\pi^2}\left(\arctan\frac{w^{\text{gt}}}{h^{\text{gt}}} - \arctan\frac{w}{h}\right)^2 \tag{9}$$

$$\eth = \frac{\nu}{(1 - \text{IoU}) + \nu}. \tag{10}$$

Among them, IoU is the intersection over union, (b, b^{gt}) represents the center point of the prediction bounding box and the real-bounding box, ρ is the Euclidean distance between the two center points, and c represents the diagonal distance of the minimum closure area that contains both the prediction bounding box and the real-bounding box. $\eth$ is the tradeoff parameter, and ν is a parameter used to measure the consistency of the aspect ratio. w^{gt}, h^{gt} represents the real width

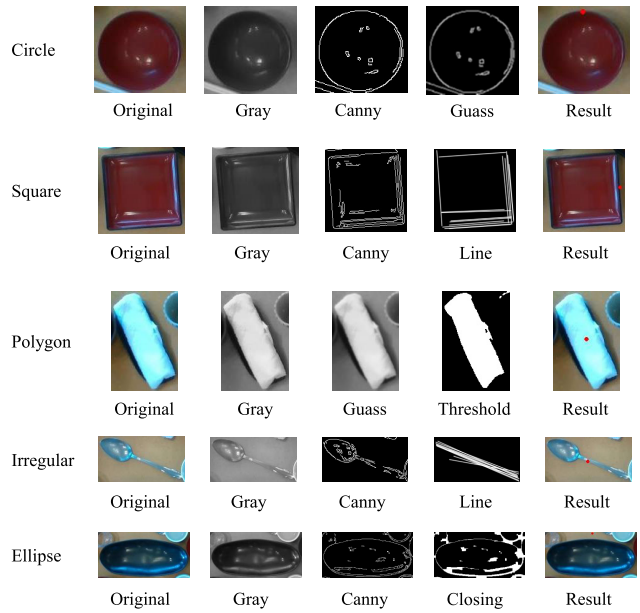


Fig. 5. Illustration of the grasp point extraction process for different classes.

and height, and w and h represent the width and height of the prediction bounding box, respectively.

C. Extraction of Grasp Points

Effective grasping of the dish by a robotic arm is a difficult task in the recycling process. When the grasp points are extracted from the whole image, mutual interference occurs between the individual dishes. By segmenting the individual dishes, we can extract the grasp points effectively. At the same time, we use different methods to extract grasp points for different types of dishes. The height information of the corresponding grasp point is obtained through the RealSense D435 sensor, and finally, the coordinate information of the grasp point in the 3-D space of the empty-dish recycling robot is determined. The extraction method of grasp points is shown in Algorithm 1.

We divide all dishes into five types: circle, ellipse, square, polygon, and irregular. The process of extracting grasp points is shown in Fig. 5, where the circle represents the round dish, the square represents square dishes, the polygon represents polygon dishes, the irregular represents irregular dishes, and the ellipse represents oval dishes. original represents the original image segmented according to the detection result, gray represents the grayscale converted image, Canny represents Canny edge detection, Guass represents Gaussian filtering, line represents line detection, threshold represents binarization processing, and result represents the grasp point extraction result.

We first segment the object dish for all detected dishes based on the detected coordinate information. Grasp-point extraction is then performed on various types of dishes. For circular dishes, we first perform the grayscale conversion and strengthen the edge information through Canny edge detection, then filter out the redundant information through Gaussian filtering. Finally, Hough circle detection is used to find the dish's contour and the grasp points. For square dishes, we use grayscale conversion and Canny edge detection to keep the lines in the image with an intersecting angle between 85° and 95°, then obtain each line's intersection points, calculate the smallest circumscribed rectangle of the intersection, and finally, obtain the four vertices' coordinates of the rectangular box to calculate the grasp points. For polygonal dishes, we use grayscale conversion, Gaussian filtering, and binarization conversion to convert the original image into a clear binarized image and directly find the largest contour in the image, and then calculate the smallest circumscribed rectangle whose center is the grasp point. The grasp point extraction of the irregular-shaped dishes is through grayscale conversion, Canny edge detection to extract the outline information of the dish, and then uses the Hough line detection to retain all the straight lines in the image and performs polygon fitting on the vertices of all straight lines, and the center of the fitting polygon is the grasp point. Through grayscale processing, Canny edge detection, and the closing operation in image processing to find the contour of the elliptical dishes, the center, the major axis, the minor axis, and the rotation angle information of the ellipse are calculated by ellipse fitting, and the grasp point is calculated.

D. Model Optimization Based on TensorRT and Deployment on Edge Platform

TensorRT is a high-performance deep learning inference SDK launched by NVIDIA, which provides low latency and high throughput for deep learning inference applications. TensorRT supports INT8, FP16, and FP32 calculations and achieves the purpose of accelerating inference by achieving an ideal tradeoff between reducing the amount of calculation and maintaining accuracy. More importantly, TensorRT reconstructs and optimizes the network structure. Fig. 6 shows the inference optimization process of TensorRT.

TensorRT eliminates useless output layers in the network to reduce computation by analyzing the network model. Through the vertical fusion of the network structure, the three layers of convolution, batch normalization, and Relu of the current mainstream neural network are integrated into one layer. Layers whose inputs are the same tensors and perform the same operations are fused together through the horizontal fusion of the network. Finally, the input of the concat layer is directly sent to the following operations, which reduces the transmission throughput and speeds up the inference process to a certain extent [16]. Moreover, quantize 32-bit floats in the network to 16-bit half floats or 8-bit integers to speed up inference.

We evaluate YOLO-GS on two edge platforms, Jetson Nano and Jetson Xavier NX, Table I shows the parameter comparison of the two devices. Jetson Nano contains a quad-core CPU and a GPU with 472 Floating-point Operations Per Second (FLOPS). Jetson Xavier NX contains a six-core CPU and a GPU with 21 Tera Operations Per Second (TOPS). All devices have the same underlying GPU architecture, so the underlying hardware instruction set remains constant and is comparable

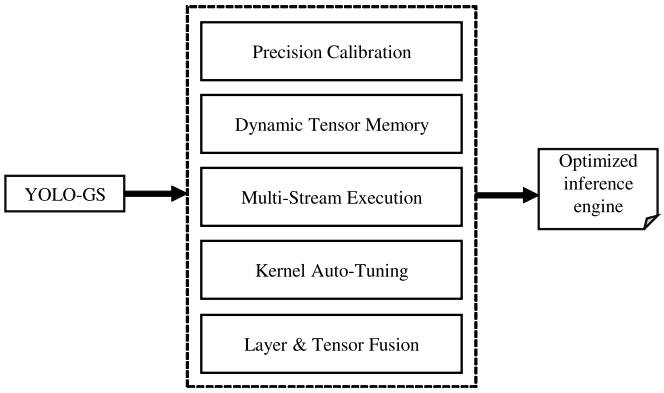


Fig. 6. Inference optimization process of TensorRT.

TABLE I
PARAMETER COMPARISON OF EDGE PLATFORMS

Hardware	Performance	CPU	GPU	Memory
Jetson Nano	472 FLOPS	ARM Cortex-A57 (quad-core) @ 1.42 GHz	128-core NVIDIA Maxwell @ 921 MHz	4GB LPDDR4 @ 1600MHz, 25.6 GB/s
Jetson Xavier NX	21 TOPS	NVIDIA Carmel ARMv8.2 (6-core) @ 1.42 GHz (6MB L2 + 4MB L3)	384-core Volta @ 1300MHz with 48 Tensor Cores	8GB LPDDR4 @ 1600MHz, 51.2 GB/s

across devices. The differences in all measurement evaluation metrics only come from TensorRT optimization [1], [28].

IV. EVALUATION

A. Experimental Configurations

1) Implementation Details: We conduct all experiments on an i9-10900 CPU and a single NVIDIA GeForce RTX 3090Ti GPU. The operating system is Ubuntu 21.04, the CUDA version is 11.4, and the GPU acceleration library cuDNN is 8.2.4. The proposed method is implemented using the TensorFlow library.

The training of all experiments is conducted using the Adam optimizer, with parameters $\beta_1 = 0.937$, $\beta_2 = 0.999$. We decay the learning rate with a warm-up cosine annealing for each epoch as follows:

$$\eta_t = \eta^i_{\min} + \frac{1}{2}\left(\eta^i_{\max} - \eta^i_{\min}\right)\left(1 + \cos\frac{T_{\text{cur}} - W_i}{T_i - W_i}\pi\right) \quad (11)$$

$\eta^{\cdot}_{\min}$ represents the minimum learning rate. $\eta^{\cdot}_{\max}$ represents the maximum learning rate. T_{cur} is how many epochs have been trained. T_i is the total number of epochs. W_i is the epochs of warm-up. In the whole training process, $\eta^{\cdot}_{\max}$ is set to 1e-3, and $\eta^{\cdot}_{\min}$ is set to 1e-5. We train the proposed model for 300 epochs, and the batch size is set to 4. During the evaluation, confidence is set to 0.5, and IoU is set to 0.3 for nonmaximum suppression.

2) Dataset: We use the public dish dataset Dish-20,[1] which contains 506 images in 20 classes. Among them, 409 images are used for training, 46 images are used for validation, and 51 images are used for testing [5]. The image size of the dataset is resized to the YOLO-GS default input size (416 × 416) previously.

[1]http://www.ihpc.se.ritsumei.ac.jp/obidataset.html

3) Evaluation Metrics: To evaluate the effect of the object detection approach, this article mainly uses AP, mAP, parameters of the model, FLOPs, and inference speed (FPS) as evaluation metrics. AP and mAP represent the accuracy of the model. The number of parameters, FLOPs, and FPS of the model represents the computational resources required by the model [34]. The meanings of these evaluation metrics are as follows:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (12)$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (13)$$

TP represents true positives, FP represents false positives, and FN represents false negatives. P means precision, and R means recall. AP is calculated by the area under the precision–recall curve (P–R curve), expressed as

$$\text{AP} = \sum_n (R_{n+1} - R_n) P_{\max}[R_n, R_{n+1}] \quad (14)$$

among them, R_n represents the recall of the n-th value, $P_{\max}[R_n, R_{n+1}]$ represents the maximum AP value in the range of $[R_n, R_{n+1}]$

$$\text{mAP} = \frac{1}{C}\sum_j^C \text{AP}_j. \quad (15)$$

The mAP is shown in (15). C is the number of classes and AP_j is the AP of the jth class.

B. Performance Comparison

We compare the proposed YOLO-GS with 18 state-of-the-art object detection methods, including faster-RCNN [11], Efficientdet [35], SSD [14], YOLOv3 [12], YOLOv4 series [13], YOLOv5 series, YOLOX series [30], and YOLO-GD [5]. Table II shows the quantitative results. In all tables, −1.000 means no relevant data is detected. The results demonstrate that YOLO-GS achieves the same accuracy as state-of-the-art object detection methods, especially in terms of mAP, AP_{11}, and AP_{50}. For example, our method achieves comparable performance with state-of-the-art two-stage detection networks faster-RCNN and YOLOX series but significantly reduces the parameters and FLOPs. Our proposed YOLO-GS has only 0.606 M of parameters, which is three times smaller than YOLOv5-Nano (1.800 M) with the most minor parameters. Our method achieves an inference speed of 108.006 FPS, which is comparable to the inference speed of YOLOX-S and YOLOX-Tiny, but on the premise of equivalent performance, the parameters amount is only 1/8 of YOLOX-Tiny and 1/14 of YOLOX-S. Although the FPS is 1/2 of YOLOv4-Tiny, we only need 1/9 parameters of YOLOv4-Tiny, and also, our method gets a higher mAP. The FLOPs of YOLO-GS are only 2.131 G, which is smaller than other state-of-the-art models (slightly larger than 1.796 G of YOLOv5-Nano, but the parameters are only 1/3 of it). Because the number of FLOPs is related to energy consumption, YOLO-GS has the minimum FLOPs, and the complexity is the lowest. Hence, it is friendly to embedded devices with limited energy. Moreover, YOLO-GS has the highest potential to improve inference speed further.

TABLE II

PERFORMANCE COMPARISON OF YOLO-GS WITH OTHER STATE-OF-THE-ART MODELS. PARAM. MEANS THE NUMBER OF PARAMETERS. AP_{11} IS THE AVERAGE OF APS AT IOU FROM 0.5 TO 0.95 WITH A STRIDE OF 0.05

Model	Size	Param. (M)	FLOPs (G)	Speed (FPS)	mAP (%)	AP_{11}	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR	AR_s	AR_m	AR_l
Faster-RCNN (VGG16)	600	137.057	-	28.405	99.540	0.804	0.992	0.968	-1.000	0.747	0.831	0.835	-1.000	0.784	0.856
Faster-RCNN (ResNet50)	600	28.537	-	15.511	99.750	0.835	0.995	0.984	-1.000	0.774	0.856	0.865	-1.000	0.793	0.885
Efficientdet (D0)	512	3.886	4.699	38.616	96.760	0.811	0.964	0.943	-1.000	0.734	0.840	0.835	-1.000	0.751	0.862
SSD	300	26.285	64.818	86.174	94.960	0.752	0.947	0.902	-1.000	0.587	0.795	0.783	-1.000	0.606	0.824
YOLOv3	416	61.679	65.520	83.364	84.160	0.645	0.841	0.786	-1.000	0.694	0.659	0.677	-1.000	0.713	0.689
YOLOv4	416	64.106	59.851	70.622	93.480	0.801	0.977	0.951	-1.000	0.699	0.832	0.834	-1.000	0.727	0.856
YOLOv4-Tiny	416	5.924	6.829	**238.151**	98.640	0.748	0.983	0.903	-1.000	0.676	0.775	0.780	-1.000	0.702	0.804
YOLOv5-X	416	86.434	86.413	66.416	98.920	0.836	0.985	0.979	-1.000	0.779	0.851	0.862	-1.000	0.798	0.875
YOLOv5-L	416	46.301	45.720	88.102	98.970	0.834	0.987	0.964	-1.000	0.786	0.849	0.860	-1.000	0.808	0.873
YOLOv5-M	416	20.985	20.387	103.203	99.060	0.830	0.987	0.970	-1.000	0.694	0.851	0.855	-1.000	0.719	0.871
YOLOv5-S	416	7.093	6.760	130.069	98.500	0.823	0.981	0.974	-1.000	0.768	0.839	0.846	-1.000	0.798	0.860
YOLOv5-Nano	416	1.800	**1.796**	128.555	95.770	0.777	0.954	0.902	-1.000	0.687	0.801	0.800	-1.000	0.709	0.823
YOLOX-X	416	99.115	118.965	54.691	**99.910**	**0.860**	**0.997**	**0.996**	-1.000	**0.814**	**0.874**	**0.885**	-1.000	**0.844**	**0.896**
YOLOX-L	416	54.232	65.646	71.931	99.730	0.858	0.995	0.991	-1.000	0.737	0.876	0.884	-1.000	0.762	0.898
YOLOX-M	416	25.335	31.077	90.413	99.680	0.854	0.994	0.989	-1.000	0.825	0.868	0.879	-1.000	0.854	0.889
YOLOX-S	416	8.968	11.271	104.147	99.440	0.852	0.991	0.986	-1.000	0.733	0.871	0.877	-1.000	0.757	0.892
YOLOX-Tiny	416	5.056	6.410	107.208	99.650	0.843	0.993	0.988	-1.000	0.731	0.859	0.872	-1.000	0.759	0.883
YOLO-GD	416	11.166	6.609	113.368	97.660	0.748	0.973	0.929	-1.000	0.700	0.769	0.780	-1.000	0.724	0.797
Ours	416	**0.606**	2.131	108.006	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.876

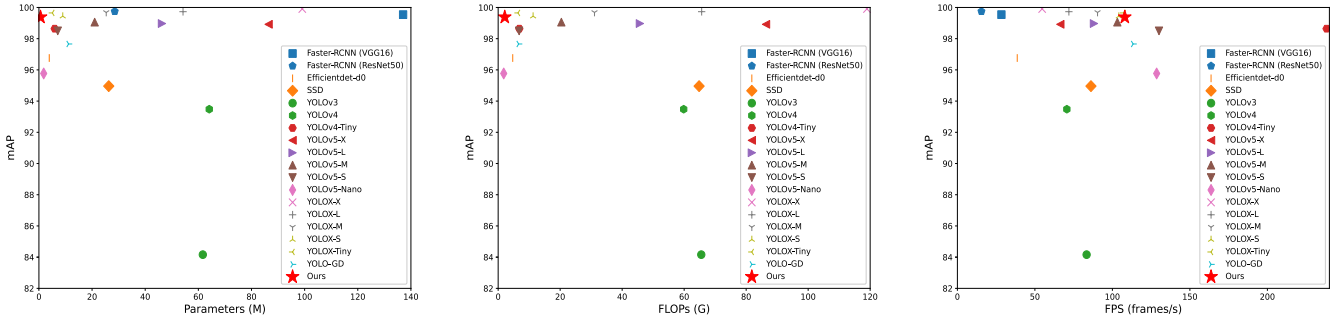


Fig. 7. Illustration of the tradeoff among mAP, the number of parameters, the number of FLOPs, and inference speed.

TABLE III

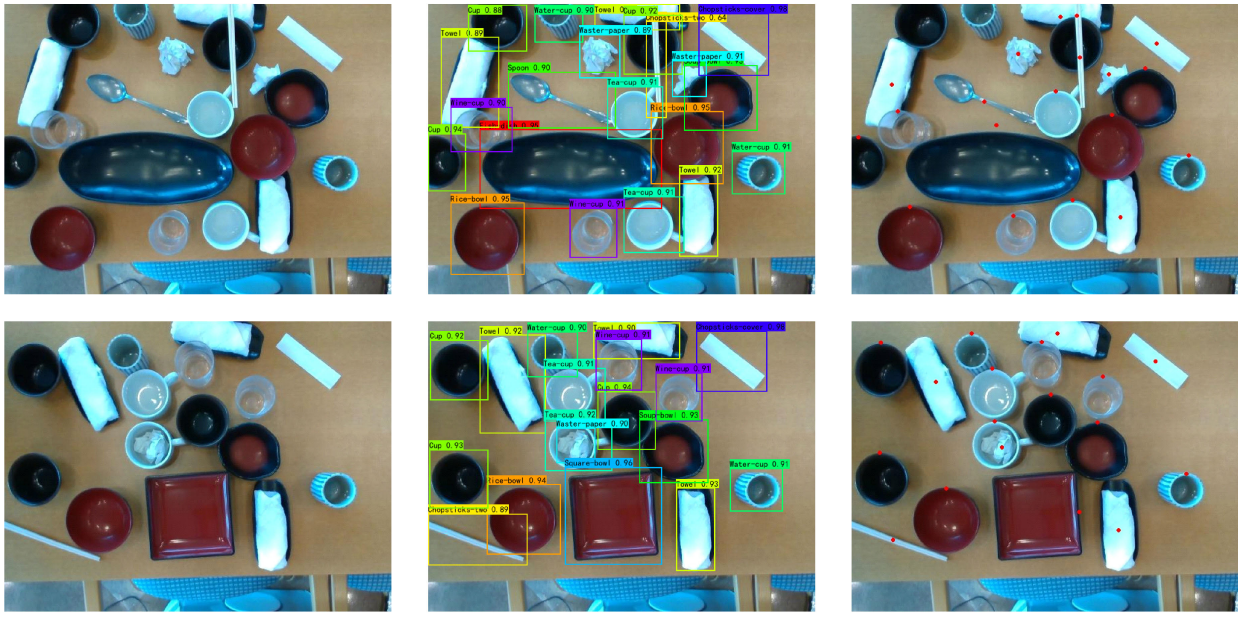
COMPARISON EXPERIMENT BETWEEN GSCONV2D AND GHOST MODULE. GSCONV2D ADDS CHANNEL SHUFFLE ON THE GHOST MODULE

Type	mAP (%)	AP_{11}	AP_{50}	AP_{75}
Ghost module	98.82	0.820	0.984	0.965
GSConv2D	**99.38**	**0.837**	**0.990**	**0.982**

To better illustrate the tradeoff between accuracy and efficiency, we present three images in Fig. 7, showing mAP against the number of parameters, the number of FLOPs, and inference speed, respectively. In the figures of mAP versus parameters and mAP versus FLOPs, YOLO-GS is in the top-left corner, which means the YOLO-GS has an ultralightweight setting and good accuracy. In the figure of mAP versus FPS, YOLO-GS is in the upper middle corner, demonstrating its good tradeoff between accuracy and inference speed. Therefore, we can conclude that YOLO-GS achieves a good tradeoff between accuracy, the number of parameters, FLOPs, and inference speed.

C. Ablation Study

Table III verifies the effectiveness of using the Ghost module and GSConv2D in the neck. We found that the results of using



Original dish images | Dish detection results | Grasp point extraction results

Fig. 8. Illustration of dish detection and grasp point extraction based on the ultralightweight object detection model YOLO-GS and grasp point extraction algorithm.

TABLE IV
COMPARE WITH THE LATEST SIGNIFICANT NECK + HEAD MODELS. OURS REPRESENTS OUR BACKBONE

Type	Param. (M)	FLOPs (G)	Speed (FPS)	mAP (%)
Ours+YOLOv3	20.915	17.802	**144.608**	94.710
Ours+YOLOv4	37.310	26.603	118.018	97.000
Ours+YOLOv5-Nano	1.185	2.423	135.502	90.370
Ours+YOLOX-Tiny	3.060	5.507	113.638	**99.650**
Ours	**0.606**	**2.131**	108.006	99.380

GSConv2D in the neck are significantly better than the Ghost module. For example, after using GSConv2D, mAP increased by 0.56%, AP_{11} increased by 1.7%, AP_{50} increased by 0.6%, and AP_{75} increased by 1.7%. We found that GSConv2D significantly improves the accuracy of the model while improving the generalization ability of the model.

Table IV shows the results of the combinatorial comparison of our proposed backbone structure with different state-of-the-art neck + head structures. Compared with Table II, it can be seen that after using our backbone in YOLOv3, the number of parameters is reduced from 61.679 to 20.915 M, the FLOPs are reduced from 65.520 to 17.802 G, and the speed is increased from 83.364 to 144.608 FPS. The mAP has increased from 84.160% to 94.710%. It is the same effect on YOLOv4. On YOLOv5-Nano, our backbone effect becomes unsatisfactory but also reduces the number of parameters and improves the inference speed. Under the same mAP, YOLOX-Tiny significantly reduces the number of parameters and FLOPs. It proves the effectiveness of our proposed backbone. At the same time, the comparison between the five models shows that our proposed neck + head has the smallest number of parameters and FLOPs. Although mAP is slightly lower than YOLOX-Tiny, our model has only 1/5 of the parameters, and the inference speed is comparable. It is proven that our proposed neck + head structure performs feature fusion with the least number of parameters.

TABLE V
PERFORMANCE EVALUATION OF YOLO-GS

Category	F_1	*Precision*	*Recall*	*AP*
Chopsticks-cover	1.00	100.00	100.00	100.00
Chopsticks-one	1.00	100.00	100.00	100.00
Chopsticks-two	1.00	100.00	100.00	100.00
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
Paper	1.00	100.00	100.00	100.00
Rice-bowl	1.00	100.00	100.00	100.00
Soup-bowl	1.00	100.00	100.00	100.00
Spoon	1.00	100.00	100.00	100.00
Tea-dish	1.00	100.00	100.00	100.00
Water-cup	1.00	100.00	100.00	100.00
Towel	0.98	96.49	100.00	100.00
Tea-cup	1.00	99.05	100.00	99.95
Fish-dish	0.96	93.10	100.00	99.87
Towel-dish	0.90	81.82	100.00	98.99
Wine-cup	0.97	96.05	98.65	98.65
Cup	0.98	97.59	98.78	97.62
Waster-paper	0.97	97.30	97.30	96.71
Square-bowl	0.98	100.00	95.83	95.83

D. Performance of YOLO-GS and Grasp Point Algorithm Evaluation

Table V shows the results of our proposed YOLO-GS on the test set. It can be seen that the AP value of 13 categories of dishes in the 20 classes is 100.00%. Among them, the "square-bowl" with the lowest recall and AP values is 95.83, the

TABLE VI
COMPARISON RESULTS OF DIFFERENT QUANTIFICATION METHODS ON DIFFERENT PLATFORMS. FP32 REPRESENTS FLOATING-POINT 32-bit QUANTIZATION, FP16 REPRESENTS FLOATING-POINT 16-bit QUANTIZATION, INT8 REPRESENTS 8-bit INTEGER QUANTIZATION, AND ORI REPRESENTS NO QUANTIZATION

Device	Type	Size	Speed (FPS)	mAP (%)	AP_{11}	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR	AR_s	AR_m	AR_l
RTX 3090Ti	FP32	416	248.283	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.876
	FP16	416	271.327	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.876
	INT8	416	274.002	82.380	0.648	0.819	0.762	-1.000	0.575	0.671	0.677	-1.000	0.593	0.699
	Ori	416	108.006	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.876
Jetson Nano	FP32	416	13.877	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.875
	FP16	416	15.359	99.380	0.837	0.990	0.982	-1.000	0.789	0.855	0.861	-1.000	0.816	0.875
	Ori	416	7.071	99.380	0.837	0.990	0.982	-1.000	0.792	0.855	0.862	-1.000	0.817	0.875
Jetson Xavier NX	FP32	416	28.067	99.380	0.837	0.990	0.981	-1.000	0.792	0.855	0.862	-1.000	0.817	0.875
	FP16	416	31.371	99.380	0.835	0.990	0.981	-1.000	0.789	0.851	0.860	-1.000	0.815	0.872
	INT8	416	32.445	71.360	0.557	0.710	0.655	-1.000	0.485	0.564	0.582	-1.000	0.499	0.590
	Ori	416	13.248	99.380	0.837	0.990	0.981	-1.000	0.792	0.855	0.862	-1.000	0.817	0.875

"fish-dish" with the lowest precision is 93.10, and the "towel-dish" with the lowest F_1 is 0.90. At the same time, the AP_{50} of YOLO-GS is 0.990, and the test accuracy meets the work requirements of the empty-dish recycling robots.

Fig. 8 shows the results of dish detection and grasp point extraction using our proposed ultralightweight object detection model YOLO-GS and grasp point extraction algorithm. In the complex desktop environment, it can be seen that YOLO-GS detects the target dish well. However, some dishes do not appear completely in the image, and the contour fitting of the dish is incomplete during the extraction of the grasp point process, resulting in the ineffective extraction of grasp point information. Our method effectively extracts the grasp points of the detected dish that appears completely in the image. As a result, our grasp-point extraction algorithm satisfies the requirements of the empty-dish recycling robot.

E. Model Optimization and Deployment on Edge Devices

The performance of our proposed YOLO-GS on different quantization methods of GPU and Jetson edge platforms is compared, as shown in Table VI. From the experimental results, on all platforms, the quantized accuracy of FP32 and FP16 is consistent with the original accuracy, and the inference speed is increased twice. On the RTX 3090Ti GPU platform, although the speed of INT8 quantization is 2.675 FPS higher than FP16, the mAP drops from 99.380 to 82.380. Similarly, in Jetson Xavier NX with similar results to RTX 3090Ti, the mAP drops from 99.380 to 71.360 after INT8 quantization. On each experimental platform (Jetson Nano does not support INT8 quantization), the inference speed of INT8 quantization is optimal, but the detection accuracy loss is large.

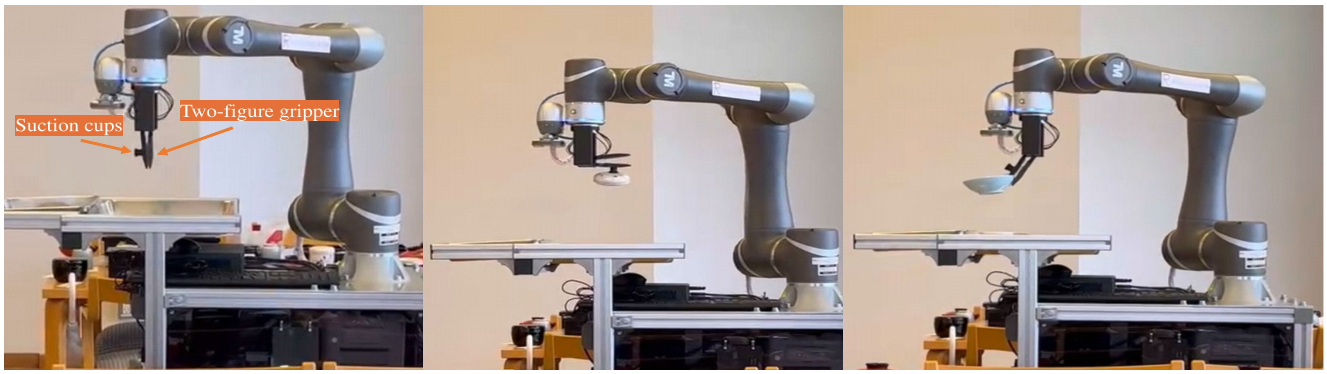
The precision after quantization by FP32 and FP16 is the same as the original precision. But the inference speed of FP16 is faster than FP32. The inference speed of YOLO-GS on Jetson Xavier NX after FP16 quantization is 31.371 FPS, which is twice of Jetson Nano (15.359 FPS). Therefore, we chose Jetson Xavier NX as the edge platform for the empty-dish recycling robot.

F. Robotic Fingers Grasp Dishes

In practical applications, we found that three pneumatic fingers cannot grasp "Chopsticks," "Paper," "Spoons," "Fish-dish," and so on very well, so we use a two-finger gripper with a suction cup to solve the problems shown in Fig. 9(a). Fig. 9(b) shows the robot using a suction cup to absorb and recycle a dish that has a smooth surface and is not easy for fingers to grasp. Fig. 9(c) shows the robot uses a two-finger gripper to grasp and recycle shallow dishes. After experimental evaluation, 20 classes of dishes can be grasped by our two-finger gripper with a suction cup. The results have proved that the extracted grasping points perfectly cooperate with the robotic fingers to grasp the dishes.

G. Discussion and Future Work

Compared with the optimal model, the detection performance of YOLOX is slightly higher than that of our proposed YOLO-GS. However, the parameters and FLOPs of the YOLOX model are too large, so it takes more time to load the model on the edge device and requires more storage space. We use related operations that reduce the number of parameters and FLOPs. For example, the computational overhead of GSConv2D is only 60%–70% off standard convolution. However, the inference time is slower than the standard convolution due to the edge devices' access quantity and memory usage limitation. Although YOLOv4-Tiny is twice as fast as YOLO-GS, the accuracy is lower than our 99.380%. When the empty-dish recycling robot works, the



(a) (b) (c)

Fig. 9. Illustration of robotic fingers grasping different types of dishes. (a) Two-finger gripper with suction cups. (b) Suction cups for dish recycling. (c) Two-finger gripper for dish recycling.

detection accuracy of dishes exceeds the inference speed. After the 16-bit quantization of the float point, YOLO-GS reaches 31.371 FPS on Jetson Xavier NX, which fully meets the detection requirements.

In practical applications, factors, such as the external environment, significantly affect the quality of the dish images. Errors occur in the feature extraction process of dish images, eventually leading to errors in the contour fitting and shifts in the position of the grasp point. For example, since the "wine-cup" is a transparent dish, it is easy to fit the bottom contour to the overall contour in the image processing stage, resulting in deviations in the grasp points extraction. However, there is a gap between the robot fingers, and the fingers can grasp the dish when the grasp point is in the fingers' gap. Hence, the robot finger works well in the case of a margin of error. In our tests, all dishes are effectively grasped. In future work, we will optimize the grasp point extraction algorithm to solve the grasp point extraction error caused by environmental factors. For all the scattered dishes on the table, we will design the optimal grasping path to solve the problem of overlapping dishes.

At the same time, the inference speed of the model on specific hardware is not only affected by the amount of calculation but also by many factors, such as memory access, hardware characteristics, software implementation, and system environment. The proposed YOLO-GS has achieved an ultralightweight structure with fewer parameters and calculation amounts (FLOPs), and it also has the most potential for significantly improving inference speed. In future work, we focus on solving factors other than the parameters and FLOPs that slow down the inference speed, such as the amount of memory access, hardware characteristics, and so on, to improve the inference speed of our model.

V. Conclusion

Instead of only focusing on model accuracy, this article explores a new direction of object detection, namely, ultralightweight object detection networks, aiming to achieve a good tradeoff between accuracy, efficiency, parameters, and FLOPs. Therefore, we propose an ultralightweight object detection model YOLO-GS and design an algorithm to extract the grasp points of dishes. Experimental results show that YOLO-GS has only 0.606 M parameters and 2.131 G FLOPs and achieves 99.380% mAP in the Dish-20 dataset. The model is quantized with a floating-point 16-bit through TensorRT, and the inference speed of 31.371 FPS is obtained on the edge device Jetson Xavier NX under the premise of ensuring the same accuracy.

To the best of our knowledge, this is the first attempt at object detection toward an accuracy–efficiency tradeoff and ultralightweight models. We demonstrate that our proposed ultralightweight object detection model YOLO-GS effectively detects dishes and extracts the coordinates of grasp points. YOLO-GS has only 0.606 M of parameters, which is much smaller than the current object detection model and is not constrained by the storage capacity of edge devices, which has far-reaching significance for the development of empty-dish recycling robots.

References

[1] X. Yue, H. Li, M. Shimizu, S. Kawamura, and L. Meng, "Deep learning-based real-time object detection for empty-dish recycling robot," in *Proc. 13th Asian Control Conf. (ASCC)*, May 2022, pp. 2177–2182.

[2] D. He, Z. Zou, Y. Chen, B. Liu, and J. Miao, "Rail transit obstacle detection based on improved CNN," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–14, 2021.

[3] X. Lu, J. Ji, Z. Xing, and Q. Miao, "Attention and feature fusion SSD for remote sensing object detection," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.

[4] Z. Wang, H. Li, X. Yue, and L. Meng, "Design and acceleration of field programmable gate array-based deep learning for empty-dish recycling robots," *Appl. Sci.*, vol. 12, no. 14, p. 7337, Jul. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/14/7337

[5] X. Yue, H. Li, M. Shimizu, S. Kawamura, and L. Meng, "YOLO-GD: A deep learning-based object detection algorithm for empty-dish recycling robots," *Machines*, vol. 10, no. 5, p. 294, Apr. 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/5/294

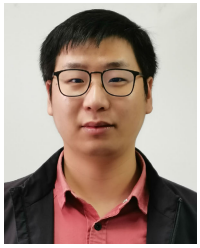
[6] Y. Fukuzawa, Z. Wang, Y. Mori, and S. Kawamura, "A robotic system capable of recognition, grasping, and suction for dishwashing automation," in *Proc. 27th Int. Conf. Mechatronics Mach. Vis. Pract. (MVIP)*, Nov. 2021, pp. 369–374.

[7] J. Yin, K. G. S. Apuroop, Y. K. Tamilselvam, R. E. Mohan, B. Ramalingam, and A. V. Le, "Table cleaning task by human support robot using deep learning technique," *Sensors*, vol. 20, no. 6, p. 1698, Mar. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/6/1698

[8] Z. Li et al., "Toward efficient safety helmet detection based on YOLOv5 with hierarchical positive sample selection and box density filtering," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.

[10] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.
[12] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[14] W. Liu et al., "SSD: Single shot MultiBox detector," in *Computer Vision*. Springer, 2016, pp. 21–37.
[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[16] M. Liu, Z. Li, Y. Li, and Y. Liu, "A fast and accurate method of power line intelligent inspection based on edge computing," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.
[17] Z. Tu, S. Wu, G. Kang, and J. Lin, "Real-time defect detection of track components: Considering class imbalance and subtle difference between classes," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.
[18] H. Li et al., "Optimizing the deep neural networks by layer-wise refined pruning and the acceleration on FPGA," *Comput. Intell. Neurosci.*, vol. 2022, Jun. 2022, doi: 10.1155/2022/8039281.
[19] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6848–6856.
[20] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1580–1589.
[21] H. Li, Z. Wang, X. Yue, W. Wang, H. Tomiyama, and L. Meng, "An architecture-level analysis on deep learning models for low-impact computations," *Artif. Intell. Rev.*, Jun. 2022, doi: 10.1007/s10462-022-10221-5.
[22] Y. Liu, Y.-C. Gu, X.-Y. Zhang, W. Wang, and M.-M. Cheng, "Lightweight salient object detection via hierarchical visual perception learning," *IEEE Trans. Cybern.*, vol. 51, no. 9, pp. 4439–4449, Sep. 2021.
[23] L. Guan, L. Jia, Z. Xie, and C. Yin, "A lightweight framework for obstacle detection in the railway image based on fast region proposal and improved YOLO-tiny network," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–16, 2022.
[24] Z. Fan, L. Shi, C. Xi, H. Wang, S. Wang, and G. Wu, "Real time power equipment meter recognition based on deep learning," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, 2022.
[25] Y. Cai et al., "YOLOv4-5D: An effective and efficient object detector for autonomous driving," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.
[26] A. A. Süzen, B. Duman, and B. Sen, "Benchmark analysis of Jetson TX2, Jetson nano and raspberry PI using deep-CNN," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robotic Appl. (HORA)*, 2020, pp. 1–5.
[27] H. Li, Z. Wang, X. Yue, W. Wang, T. Hiroyuki, and L. Meng, "A comprehensive analysis of low-impact computations in deep learning workloads," in *Proc. Great Lakes Symp. VLSI*, New York, NY, USA, Jun. 2021, pp. 385–390, doi: 10.1145/3453688.3461747.
[28] A. Koubaa, A. Ammar, A. Kanhouch, and Y. AlHabashi, "Cloud versus edge deployment strategies of real-time face recognition inference," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 143–160, Jan. 2022.
[29] H. Li, J. Li, H. Wei, Z. Liu, Z. Zhan, and Q. Ren, "Slim-neck by GSConv: A better design paradigm of detector architectures for autonomous vehicles," 2022, *arXiv:2206.02424*.
[30] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.
[31] C. Wang, H. Liao, Y. Wu, P. Chen, J. Hsieh, and I. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2020, pp. 390–391.
[32] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.
[33] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 8759–8768.
[34] X. Yan et al., "Multitargets joint training lightweight model for object detection of substation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 25, 2022, doi: 10.1109/TNNLS.2022.3190139.
[35] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10781–10790.



Xuebin Yue (Student Member, IEEE) is currently pursuing the Ph.D. degree from the College of Science and Engineering, Ritsumeikan University, Kusatsu, Japan.

His research interests include image recognition, artificial intelligence, and the applications of cultural heritage preservation and protection.

Hengyi Li (Student Member, IEEE) is currently pursuing the Ph.D. degree with the College of Science and Engineering, Ritsumeikan University, Kusatsu, Japan.

His research interests include the high-performance computing of artificial intelligence (AI), FPGA-based accelerator design for AI, and edge computing.

Lin Meng (Member, IEEE) received the Ph.D. degree from the Graduate School of Science and Engineering, Ritsumeikan University, Kusatsu, Japan, in 2012.

He was a Research Associate, an Assistant Professor, and a Lecture with the Department of Electronic and Computer Engineering, Ritsumeikan University, from 2011 to 2012, from 2013 to 2017, and in 2018, respectively. In 2015, he was a Visiting Scholar with the Department of Computer Science and Engineering, University of Minnesota at Twin Cities, Minneapolis, MN, USA. He is currently an Associate Professor with the College of Science and Engineering, Ritsumeikan University. His research interests include computer architecture, parallel processing, the Internet of Things, and artificial intelligence.

Dr. Meng is a member of the ACM, IPSJ, IEICE, and IEE.