applied sciences

MDPI

Article

Design and Acceleration of Field Programmable Gate Array-Based Deep Learning for Empty-Dish Recycling Robots

Zhichen Wang, Hengyi Li, Xuebin Yue and Lin Meng *

College of Science and Engineering, Ritsumeikan University, 1-1-1, Nojihigashi, Kusatsu 525-8577, Shiga, Japan; gr0415hp@ed.ritsumei.ac.jp (Z.W.); gr0468kx@ed.ritsumei.ac.jp (H.L.); gr0468xp@ed.ritsumei.ac.jp (X.Y.)
* Correspondence: menglin@fc.ritsumei.ac.jp

Abstract: As the proportion of the working population decreases worldwide, robots with artificial intelligence have been a good choice to help humans. At the same time, field programmable gate array (FPGA) is generally used on edge devices including robots, and it greatly accelerates the inference process of deep learning tasks, including object detection tasks. In this paper, we build a unique object detection dataset of 16 common kinds of dishes and use this dataset for training a YOLOv3 object detection model. Then, we propose a formalized process of deploying a YOLOv3 model on the FPGA platform, which consists of training and pruning the model on a software platform, and deploying the pruned model on a hardware platform (such as FPGA) through Vitis AI. According to the experimental results, we successfully realize acceleration of the dish detection using a YOLOv3 model based on FPGA. By applying different sparse training and pruning methods, we test the pruned model in 18 different situations on the ZCU102 evaluation board. In order to improve detection speed as much as possible while ensuring detection accuracy, for the pruned model with the highest comprehensive performance, compared to the original model, the comparison results are as follows: the model size is reduced from 62 MB to 12 MB, which is only 19% of the origin; the number of parameters is reduced from 61,657,117 to 9,900,539, which is only 16% of the origin; the running time is reduced from 14.411 s to 6.828 s, which is only less than half of the origin, while the detection accuracy is decreased from 97% to 94.1%, which is only less than 3%.

Keywords: empty-dish recycling robot; deep learning; object detection; FPGA; model pruning; Vitis AI

Citation: Wang, Z.; Li, H.; Yue, X.; Meng, L. Design and Acceleration of Field Programmable Gate Array-Based Deep Learning for Empty-Dish Recycling Robots. *Appl. Sci.* **2022**, *12*, 7337. https://doi.org/10.3390/app12147337

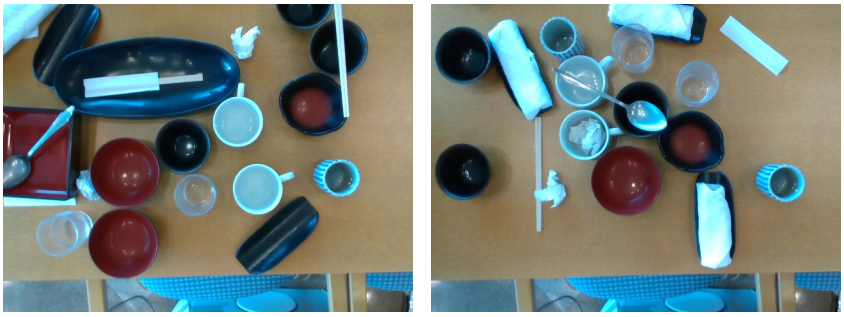
Academic Editor: Vincent A. Cicirello

Received: 30 June 2022
Accepted: 19 July 2022
Published: 21 July 2022

1. Introduction

In recent years, the phenomenon of population aging has become more and more apparent all over the world as the society develops. The aging of the population means a larger proportion of the population aged over 60 years, and also indicates a decrease in the percentage of the working population. In this situation, robots that can help humans to do some work will effectively alleviate the problem of labor shortage. This paper proposes the detection system for empty-dish recycling robots to automatically sort and recycle used dishes for the subsequent process. It has wide application scenarios, including restaurants, cafeterias, and other places with many people.

The key for the proposal is the detection of the dishes. In fact, there are many different types of dishes for each scenario as shown in Figure 1, including bowls, plates, cups of different colors and sizes, and even chopsticks and spoons. It is hard work to correctly detect and identify these different types of dishes for recycling. At the same time, in order to apply the empty-dish recycling robot in real life, the time for detection should not be too long, otherwise it would seriously affect the subsequent recycling work. Therefore, our aim is to increase the speed of the detection process as much as possible while ensuring the accuracy of the detection.



(a) (b)

Figure 1. Examples of dish photos waiting to be detected. (**a**) first example, (**b**) second example.

For the detection of dishes, a deep learning based technique is adopted. Nowadays, the technique has been widely used in various fields of our life, including automatic driving [1], environmental protection [2], cultural heritage protection [3,4], etc. In addition to images, deep learning can also be applied to speech recognition [5], machine translation [6], vulnerability detection [7], and other fields [8,9]. In terms of computer vision, the tasks could be simply classified into two purposes: image classification and object detection. Image classification is mainly to associate one or more recognized labels with the given image. As for object detection, it is to detect the instances of semantic objects in images or videos. The task requires the neural network to recognize various sub-images and locations for drawing bounding boxes around each recognized sub-image.

In terms of the hardware platforms for a deep neural network (DNN), although image classification and object detection could be performed extremely fast using a graphics processing unit (GPU), the cost of using GPU is high, it is difficult for GPU to be applied to edge. Thus, we consider deploying a field programmable gate array (FPGA) to dish recycling robots. Compared with GPU, FPGA has the advantages of small size, high portability, and low energy consumption. In addition, FPGA has similar parallel computing features to GPU, which makes the performance of FPGA much better than a central processing unit (CPU). Therefore, in this research, we implement the dish detection system on FPGA.

The three major contributions of this paper are as follows:

- We build a unique object detection dataset of 16 common kinds of dishes, which can be used in the application of empty-dish recycling robots.
- We propose a formalized process of deploying a YOLOv3 object detection model on the FPGA platform, which is divided into two parts: software process and hardware process.
- We implement a method to reduce the size of the YOLOv3 model through pruning it on the software platform, and enable the pruned model to be deployed to the FPGA evaluation board via Vitis AI, which realizes acceleration of deep learning on the FPGA platform.

The rest of this paper is organized as follows: Section 2 introduces the research related to an FPGA-based dish detection system. Section 3 shows the robotics and dish dataset used in this research. Section 4 describes the formalized process of deploying a YOLOv3 object detection model on the FPGA platform, which is divided into software process and hardware process. Section 5 lists specific research results on both software and hardware platforms. Section 6 analyzes the the experimental results and discusses the limitations and future directions of this research. Section 7 concludes the paper and draws future research directions.

2. Related Work

This section introduces the key parts of the system, including the application fields of robotics, the deep learning based object detection, and the FPGA for accelerating deep neural networks.

Threat to a validity [10,11]: In this research, to find related work about application of robotics, we use search strings 'robot and medical', 'robot and agriculture', and 'robot and dish'. To find related work about deep learning based object detection, we use search strings 'deep learning', 'convolutional neural network', and 'object detection'. To find related work about deep learning based on FPGA, we use search strings 'deep learning and FPGA' and 'convolutional neural network and FPGA'. We mainly search related work in the database named IEEE Xplore, and also use other databases such as ACM Digital Library and SpringerLink.

2.1. Robotics

Robotics is nowadays widely used in various fields to improve the quality of people's lives. For example, it can be used in fields related to human life [12], agriculture fields, and dishwashing.

2.1.1. Human Life

In disaster recovery works, Ref. [13] develops a pneumatic humanoid robot arm that can be mounted on any type of construction machinery, and constructs a control system by software to realize remote control of construction machinery on the developed system. In the medical field, Ref. [14] shows that the C-arm system is widely used in fluoroscopic surgery to instantly monitor the patient's status during surgery. Ref. [15] delves into the relationship between humans and robots, analyzes the behavior and characteristics of various types of robots, then indicates that with the development of artificial intelligence and robotics; employment in many fields is also increasing due to the high demand for intelligent machines around the world.

2.1.2. Agriculture

In agriculture, Ref. [16] designs a special gripper for picking apples from trees in apple orchards. Ref. [17] designs a smartphone-controlled spraying robot for pepper farms, which can not only help farmers reduce agricultural labor and solve the labor shortage problem, but also reduce people's direct contact with traditional chemicals that are very harmful to their bodies.

2.1.3. Dishwashing

As for tasks dealing with dishes, Ref. [18] has developed a robotic system to perform the operation of taking out various dishes from a commercial dishwasher. As a matter of fact, robots provide efficient solutions for various application scenarios.

2.2. Deep Learning Based Objection Detection

The basic theory of deep learning was proposed back in 1980 [19]. Deep learning is the process of learning the intrinsic laws of sample data, and the information obtained from this learning process can be of great help in the interpretation of data such as text, images and sounds. Its final goal is to make machines analyze and learn like humans, and be able to recognize data such as text, images, and sounds. To achieve this goal, we usually take the action of building up a convolutional neural network (CNN) model [20–25], training this model with known data, and eventually making the model be capable of recognizing unknown data.

2.2.1. Two-Stage Object Detection Algorithms

In the field of object detection, the most popular algorithms can be divided into two categories, one is the two-stage R-CNN family (R-CNN [26], Fast R-CNN [27], and Faster R-CNN [28]), which needs to first generate potential bounding boxes in the image and second run the classifier on these proposed boxes.

2.2.2. One-Stage Object Detection Algorithms

The other class is one-stage algorithms such as SSD [29] and YOLO [30], which reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. It allows a single neural network to predict bounding boxes and class probabilities directly from full images in one evaluation. After that, improved versions of YOLO (YOLOv2 [31], YOLOv3 [32], and YOLOv4 [33]) were proposed to achieve high-performance object detection while ensuring accuracy.

2.3. Deep Learning Based on FPGA

Implementing deep learning using a field programmable gate array (FPGA) [34] can accelerate deep learning, reduce energy consumption, and deploy CNN on edge platforms and embedded systems.

2.3.1. Quantization for Acceleration

In order to realize acceleration of deep learning on FPGA, relevant research can be divided into two general directions: quantization and weight reduction. Quantization in deep learning usually refers to converting floating-point numbers into shaped data as a way to reduce memory bandwidth and storage space, improve system throughput, and reduce system latency [35].

2.3.2. Weight Reduction for Acceleration

Weight reduction refers to removing redundant parameters by methods such as pruning and structural simplification, so as to compress the size of the whole CNN model and improve the speed of deep learning [36–39].

2.3.3. Energy Consumption

As for the energy consumption of deep learning, Ref. [40] argues that CNN is widely used in modern AI systems, but also brings challenges in terms of throughput and energy efficiency to the underlying hardware. In this paper, we deploy the neural network on FPGA by utilizing the Vitis AI to alleviate these problems.

3. Application and Dataset

In this research, our aim is to develop and accelerate an application of detection system for empty-dish recycling robots. In order for the empty-dish recycling robot to work correctly, we need to prepare the dish dataset in advance for training an object detection model.

3.1. The Empty-Dish Recycling Robot

The working process of empty-dish recycling robot is shown in Figure 2. In addition to the external robotic arm for grasping, the robot is equipped with an inner camera system for taking photographs and an object detection system. It also has components of a recycling-station and body as Figure 2a shows. The object detection system is based on an object detection network that has been trained to recognize different kinds of dishes. When the model has been loaded and the camera is going to take a photo, the state of the robot preparing to grasp the dish is shown in Figure 2a. As the photo is successfully taken, the system performs object detection based on the photo, detects the positions of different dishes, and then passes the position information to the grasping system. Figure 2b is the process of the robot grasping the dish according to the position information. When the grasping process is finished, the robot arm puts the dish into the recycling platform as shown in Figure 2c. After repeating the above process several times, the dishes in the photo are sorted and recycled.

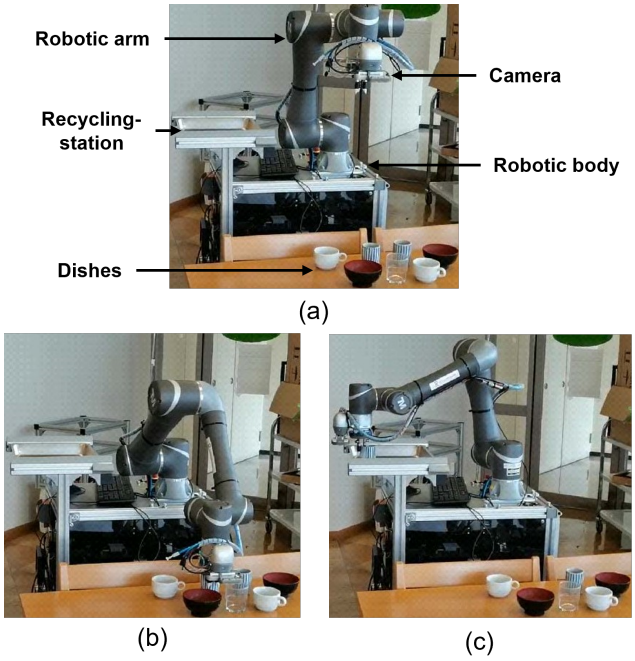


Figure 2. Working process of the proposed empty-dish recycling robot. (**a**) preparation stage, (**b**) grasping process, (**c**) recycling process.

3.2. Dataset for Object Detection Training

The object detection model needs to be trained in advance on a dataset. The main purpose of this research is to detect different kinds of dishes. There are 16 kinds of dishes, including Fish-dish, Towel-dish, Rice-bowl, Chopsticks-two, Towel, Cup, Spoon, Soup-bowl, Water-cup, Tea-cup, Waster-paper, Square-bowl, Chopsticks-one, Paper, Chopsticks-cover, Wine-cup. As Figure 3 shows, the two images contain all 16 kinds of dishes.

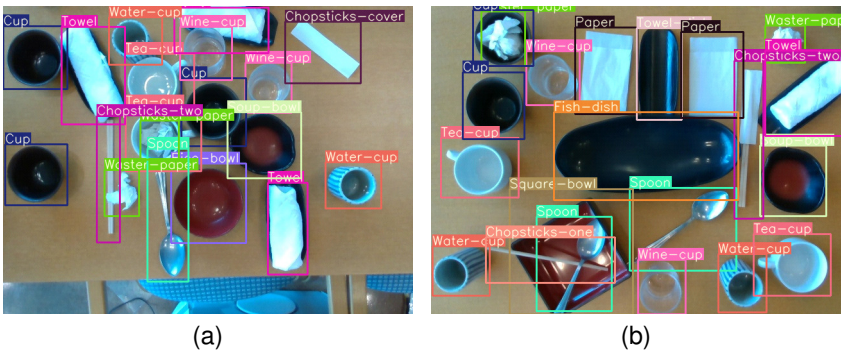


Figure 3. Examples of 16 kinds of dishes in the training dataset. (**a**) first example, (**b**) second example.

The training set refers to the dataset used to train the model, and the testing set is the dataset used to test the accuracy of the trained model. It should be noted that the data in the two sets must not overlap. For the research, we prepare 343 dish images in total, of which 275 images are used as the training set and 68 images are used as the testing set. All images have the same size, which is 640 × 480 pixels. To improve the randomness, each image contains many different kinds of dishes, and the type, quantity, and locations of the dishes are not fixed.

In image classification projects, each image corresponds to a single label, but in object detection projects, each image typically contains labels of multiple different objects, and the position of each object needs to be recorded. Since this research is an object detection project, the dataset is created by manually drawing boxes for all the dishes in all images and recording the coordinates of the four vertices of each box, which mean the location

of the dish. Finally, each image has a label file, which contains the type of each dish and its coordinates in the image. These label files are essential in the subsequent training and testing process. It is important to note that, in order to comply with the CNN model framework, these coordinates are relative coordinates in the range of 0 to 1, and need to be multiplied by the width or height of the image to be the actual absolute coordinates.

4. Research Methods

The formalized process of deploying deep learning models such as YOLOv3 on the FPGA platform is divided into two parts: software process and hardware process. Thus, the research methods can be roughly divided into two parts: software method and hardware method. The software method means that the object detection model is trained on the software platform, and the size of the model is reduced by pruning. After operations on the software platform, we use Vitis AI to deploy the pruned model to the FPGA, which is the hardware method.

4.1. Software Method

In this research, we employ YOLOv3 [32] as the kernel to perform dish detection. To extract features, YOLOv3 adopts a convolutional network followed by a fully connected layer to obtain prediction values. In YOLOv3, the size of the output feature map could be adjusted by modifying the convolutional step size, hence there is no special restriction on input image size. Meanwhile, YOLOv3 is inspired by the idea of pyramidal feature maps, which takes small-size feature maps to detect large-size objects and large-size feature maps to detect small-size objects.

In the YOLOv3 convolutional network, the time complexity [41] of a single convolutional layer is $O(M^2K^2C_{in}C_{out})$, where M^2 represents the size of the output feature map of the convolutional layer, K^2 represents the size of the convolution kernel, C_{in} represents the number of input channels, and C_{out} represents the number of output channels. Therefore, the time complexity of the entire YOLOv3 convolutional network is $O(sum_{i=1}^{N} M_i^2 K_i^2 C_{i-1} C_i)$, where N represents the number of layers of the convolutional neural network, i represents the serial number of the convolutional layer, C_{i-1} represents the number of input channels of the i-th layer, and C_i represents the number of output channels of the i-th layer.

4.1.1. Training Process

For training the model, we need to prepare the cfg file of YOLOv3 in advance, i.e., the configuration file. The cfg file details the structure of YOLOv3, including the total number of convolutional layers, the parameters of each convolutional layer, the types of activation function, the size of the input image, the learning rate, and the settings of anchors. As the object categories vary with different tasks, the number of classes and filters of some convolutional layers in the cfg file also needs to be adjusted. Considering that the accuracy of the detection improves as the input image size increases, we take 608 pixels instead of using the default 416 pixels.

4.1.2. Pruning Process

After the YOLOv3 is trained, we apply the pruning technique to compress the model and thus improve the performance of the network. The core idea of pruning comes from [42]. Specifically, the scaling factor γ, i.e., the weights of the layer, of batch normalization (BN) layer is used as the importance factor. A smaller γ means that the corresponding channel is less important. Nowadays, many CNN models have a BN layer after the convolutional layer, and the importance of both the convolutional layer and the BN layer can be determined based on the scaling factors. By performing threshold filtering on the scores, the less important channels are screened out and then removed from the network. As a result, we can reduce the size of the model without affecting the accuracy of the model too much and achieving the effect of compression. In the specific implementation, it can be divided into three steps as follows:

1. Performing sparse training for the YOLOv3 model with darknet framework. Sparse training in this research means applying the L1 regularization on the weights of BN layers, which results in sparsity of the weights. In this research, we first performed the sparse training and then performed pruning (Method A). However, we found that the model accuracy decreased significantly after pruning. Thus, we adjust some parameters and then perform sparse training again as Method B to improve the accuracy. Specifically, we mainly adjust the learning rate by warping up the learning rate in the first few epochs, and decaying the learning rate in an appropriate proportion after the training reached 70% of the total epoch. After the adjustment, the model accuracy of method B improves significantly compared to method A. This indicates that sparse training plays an important role and can directly affect the accuracy of the model after pruning.
2. Pruning the model after sparse training. We first set an expected pruning percentage, i.e., the proportion of channels that are expected to be cut. Then, threshold concerning the scaling factors is determined and the channels with the scaling factor value smaller than the threshold are pruned. In detail, we tried a total of three methods: Prune method, Shortcut method, and Slim method. Prune method is a conservative strategy. There are five groups of 23 shortcut connections in YOLOv3 corresponding to the added operation. To ensure that the two input dimensions of the shortcut are the same after channel pruning, this method does not prune the layers directly connected to the shortcut. As a result, the prune method avoids the dimension processing. The shortcut method prunes the convolutional layers involved in the shortcut. It uses the mask of the first convolutional layer in the shortcut of each group, and a total of five masks to prune shortcut-related convolutional layers of the five groups. Thus, the pruning rate is improved compared with Prune Method. Slim method has the highest channel pruning rate. First, the mask of each convolutional layer is found with a global threshold. In addition, then for each group of shortcuts, the pruned mask of each connected convolutional layer is taken as a merge set and pruned, so that each relevant layer is considered, and the reserved channels of each layer are also limited.
3. Fine-tuning the pruned model. After pruning, the accuracy of the model decreases inevitably. Thus, a fine-tuning operation, which means retraining the pruned model, is required to compensate for the accuracy loss. Similar to the sparse training process, we can also adjust the learning rate in the fine-tuning process to improve the accuracy of the model. After fine-tuning, we obtain a lightweight YOLOv3 model with a high accuracy.

4.2. Hardware Method

The Vitis AI [43] is a convenient and efficient tool which can deploy and accelerate AI inference on Xilinx hardware platforms, including edge devices and Alevo accelerator cards. In this research, we deploy the pruned YOLOv3 model on the ZCU102 evaluation board using the Vitis AI. Specifically, as shown in Figure 4, the whole process can be approximately divided into the following steps:

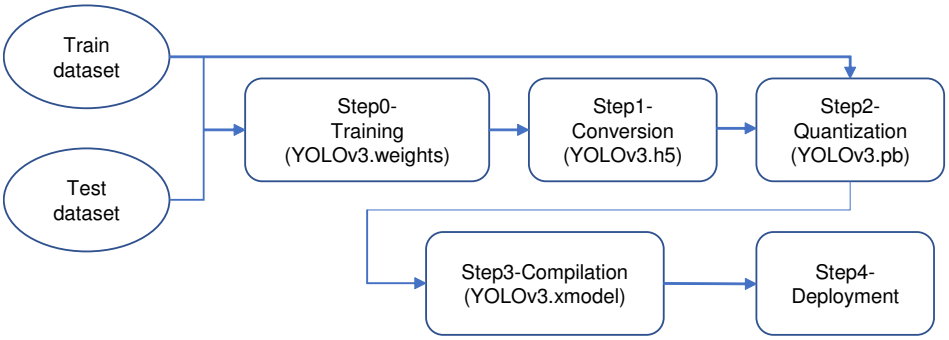


Figure 4. Flow of deploying the YOLOv3 model on the ZCU102 evaluation board.

1. Model conversion. With the weights file and the corresponding cfg file, the YOLOv3 model could be converted into a keras.h5 model that will be used for subsequent operations.
2. Model quantization. Typically, when training a CNN, we use 32-bit floating point weights and activations on CPU or GPU platforms. The Vitis AI quantizer can convert 32-bit floating point weights and activations to INT8 format, so we can use it to reduce computational complexity without loss of detection accuracy. In addition, in this research, 100 unlabeled images are used for quantitative calibration. After this process, we can obtain a .pb file converted from the .h5 file and then use the .pb file for next operation.
3. Model compilation. After model quantization, we can use the Vitis AI compiler to convert the .pb file to a .xmodel file, which can run on the FPGA board.
4. Model deployment. In this process, we need to copy the compiled .xmodel file and .prototxt configuration file to the SD card that has been prepared; then, we can use the YOLOv3 model to implement object detection of dishes on FPGA. In addition, we need to adjust some parameters in the .prototxt file to obtain better results. Generally speaking, the biases in the .prototxt file are the anchors in the original cfg file, and they should be arranged in the order of the anchors. However, in fact, in the experiment, we find that, if the biases are set in the order of the original anchors, there will be many boxes that are obviously not the correct detection results, so the biases must be modified. After trying, by swapping the first 6 numbers and the last 6 numbers of the biases, the test results returned to normal.

On the FPGA platform, there are two ways to verify the accuracy of the model. The first way is to perform object detection on specific pictures, as shown in Figure 5. The second way is to infer all 68 test images in batches to obtain a result file similar to the original label file, including the type and coordinates of each dish in the test image, and the confidence of each test result. Then, we need to compare the result file with the original label file to determine accuracy of the model. It should be noted that, unlike the relative coordinates in the label file, the coordinates in the result file are absolute coordinates, so accurate test results can only be obtained after some transformation. In addition to accuracy, we also recorded the time from running the YOLOv3 model on the ZCU102 evaluation board to obtain the result file, as an evaluation of performance of the model.

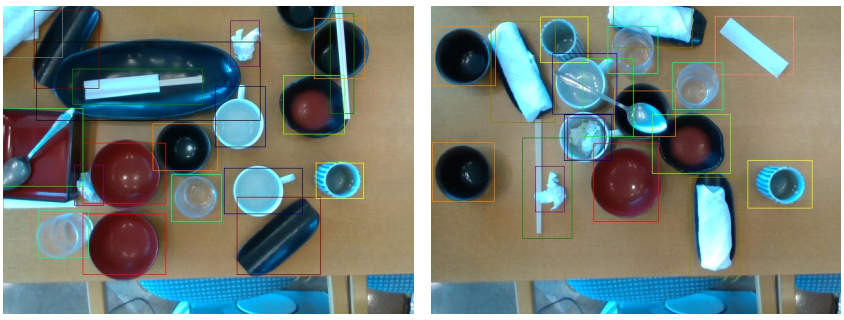


Figure 5. Examples of detection results on FPGA. (**a**) first example, (**b**) second example.

5. Experimental Results

In terms of the experimental environment in this research, we adopt NVIDIA GeForce RTX 3080 Ti for training the model. The CPU is Intel Core i9-10900 2.80 GHz processor, with 64 GB RAM. The operating system is 64-bit Ubuntu 18.04.4 LTS, and the CUDA version is 11.5. As for the FPGA, we adopt the Zynq UltraScale+ MPSoC ZCU102 evaluation board of Xilinx.

5.1. Software Results

In this research, we first use the dish dataset to train the YOLOv3 model under darknet. During the training process, we save the model every 1000 epochs, and we also save the model with the largest mean average precision (mAP) for subsequent research. Figure 6 shows the results of the trained model from 1000 epochs to 30,000 epochs. Generally speaking, in object detection, mAP is a comprehensive indicator for evaluating the detection ability of a model. Therefore, in this research, we call the model with the largest mAP in a certain training process as the best model of the training process.

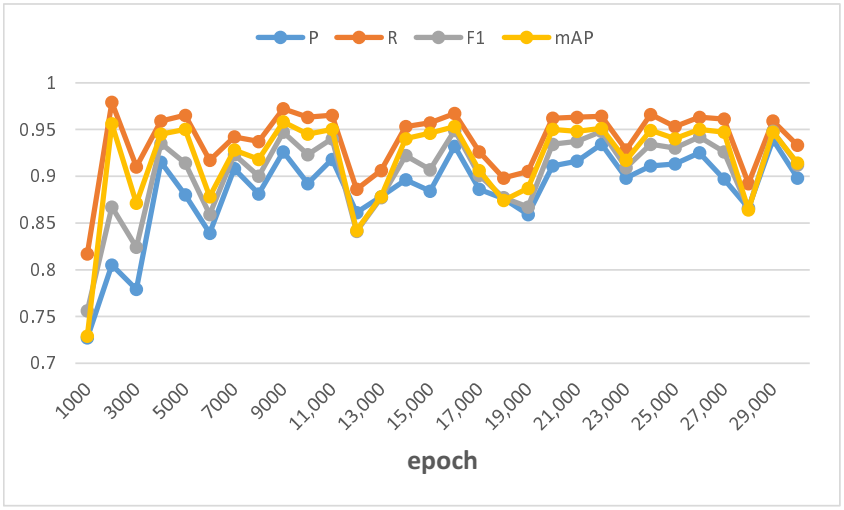


Figure 6. Detection results under darknet.

The evaluation indicators and related definitions in Figure 6 are as follows [44,45]:

TP True Positive means that the samples are classified as positive samples and assigned correctly;

TN True Negative means that the samples are classified as negative samples and assigned correctly;

FP False Positive means that the samples are classified as positive but wrongly assigned;

FN False Negative means that the samples are classified as negative samples but wrongly assigned;

P Precision means the proportion of the number of correctly assigned positive samples to the total number of all assigned positive samples. The formula is (1);

R Recall means the proportion of the number of correctly assigned positive samples to the total number of real positive samples, and the formula is (2).

$F1$ F1 score is the harmonic mean of precision and recall. It is an evaluation indicator that integrates Precision and Recall. It avoids the single maximum value of Precision or Recall and is used to comprehensively reflect the overall indicator. The formula is (3):

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

The Precision–Recall curve can measure the quality of the object detection model, but it is not convenient for comparing the models. On the basis of the precision–recall (P–R) curve, by calculating the average value of the precision value corresponding to each recall value, an evaluation indicator is obtained: average precision (AP), which is used to measure the detection capability of trained model on the class of interest. Before calculating AP, in order to smooth the P–R curve and reduce the influence of curve jitter, the P–R curve is first

interpolated. Given a Recall value R, the Pinterp used for interpolation is the maximum Precision value between the current R value and the next Recall value R'. AP calculation can be defined as the area of the interpolated precision–recall curve and the x-axis, as shown in Formula (4), where R_1, R_2,..., R_n is the Recall value corresponding to the first interpolation of the Precision interpolation segment in ascending order. The average value of each category of AP is denoted as mAP. Suppose there are K detection types in total and the formula is shown in (5), and mAP measures the detection ability of the trained model on all types:

$$AP = sum_{i=1}^{n-1}(R_{i+1} - R_i)Pinterp(R_{i+1}) \quad (4)$$

$$mAP = \frac{sum_{i=1}^{K} AP_i}{K} \quad (5)$$

For the best model with the largest mAP, we also recorded the specific detection results of each dishes, as shown in Table 1. At the same time, we use this best model as the base model for subsequent pruning process and deployment to FPGA.

Table 1. Detection results of base model.

Class	P	R	F1	mAP
all	0.557	0.987	0.686	0.964
Fish-dish	0.5	1	0.667	0.974
Towel-dish	0.755	1	0.86	0.992
Rice-bowl	0.855	1	0.922	0.992
Chopsticks-two	0.411	0.988	0.581	0.958
Towel	0.598	1	0.748	0.974
Cup	0.91	0.987	0.947	0.978
Spoon	0.263	0.952	0.413	0.901
Soup-bowl	0.803	0.984	0.884	0.975
Water-cup	0.375	1	0.545	0.975
Tea-cup	0.918	1	0.957	0.994
Waster-paper	0.358	1	0.527	0.951
Square-bowl	0.553	0.979	0.707	0.944
Chopsticks-one	0.315	0.906	0.468	0.859
Paper	0.389	1	0.56	0.975
Chopsticks-cover	0.679	1	0.809	0.988
Wine-cup	0.234	1	0.379	0.995

5.2. Hardware Results

When evaluating the performance on FPGA, we choose the model with the highest mAP in each training process, which is called the best model. First, we deploy the original best model trained under darknet to FPGA for test, then deploy it after additional sparse training by Method A and Method B to FPGA for testing. The results are shown in Tables 2 and 3. Among them, Table 2 shows the AP of each type of dishe, and Table 3 shows the mAP of the model and some other indicators, including the total number of parameters of the model, the size of the model deployed to FPGA, and the total time required to test all 68 images.

Then, for both the process of sparse training and fine-tuning, we choose and deploy the best models respectively to FPGA again for test. For sparse training Method A, the results are shown in Tables 4 and 5. For sparse training Method B, the results are shown in Tables 6 and 7. When pruning, we use three methods: Prune method, Shortcut method, and Slim method, respectively. In addition, we set the pruning ratio to 30%, 50%, and 80%, respectively, and record the results. Compared with Table 3, two results are added in Tables 5 and 7, which are the number of channels actually pruned and the actual pruning ratio during the pruning process. Due to unknown reasons, after using the sparse training Method B and the Shortcut pruning method, an error occurred during the compilation

process of converting the .weight model to the .xmodel model, resulting in a compilation failure, so that the AP, mAP, and test time of the model can not be obtained.

Table 2. Detection accuracy of model before pruning on FPGA.

Class	Origin	Sparse Method A	Sparse Method B
Chopsticks-two	0.958	0.958	0.936
Cup	0.985	0.987	0.985
Water-cup	0.98	0.986	0.97
Tea-cup	0.999	0.989	0.999
Waster-paper	0.992	0.955	0.965
Chopsticks-one	0.949	0.92	0.912
Wine-cup	1	1	1
Fish-dish	0.979	0.878	0.948
Spoon	0.94	0.932	0.921
Soup-bowl	0.969	0.953	0.969
Square-bowl	0.974	0.972	0.943
Rice-bowl	0.988	0.995	0.99
Towel	0.97	0.953	0.971
Chopsticks-cover	0.947	0.895	0.99
Towel-dish	0.966	0.997	0.936
Paper	0.97	0.96	0.964

Table 3. Performance of the model before pruning on FPGA.

Class	Origin	Sparse Method A	Sparse Method B
mAP	0.97	0.96	0.964
Parameters	61,657,117	61,657,117	61,657,117
Size	62 MB	62 MB	62 MB
Time	14.411 s	14.453 s	14.544 s

Table 4. Detection accuracy using sparse training Method A on FPGA.

Class	Prune30%	Prune50%	Prune80%	Shortcut30%	Shortcut50%	Shortcut80%	Slim30%	Slim50%	Slim80%
Chopsticks-two	0.869	0.789	0.545	0.872	0.706	0.012	0.748	0.135	0.148
Cup	0.987	0.981	0.982	0.986	0.984	0.549	0.985	0.159	0.138
Water-cup	0.976	0.983	0.963	0.984	0.974	0.255	0.987	0.717	0.427
Tea-cup	0.997	0.983	0.98	0.982	0.987	0.37	0.972	0.573	0.173
Waster-paper	0.891	0.792	0.78	0.845	0.803	0.098	0.684	0373.	0.217
Chopsticks-one	0.831	0.697	0.68	0.814	0.689	0.002	0.736	0.035	0.027
Wine-cup	1	1	1	1	1	0.174	1	0.732	0.316
Fish-dish	0.819	0.814	0.779	0.825	0.811	0.239	0.786	0.25	0.096
Spoon	0.847	0.808	0.682	0.851	0.741	0.06	0.748	0.041	0.041
Soup-bowl	0.961	0.95	0.841	0.979	0.972	0.368	0.97	0.586	0.754
Square-bowl	0.906	0.874	0.902	0.942	0.903	0.086	0.908	0.52	0.277
Rice-bowl	0.979	0.984	0.789	0.996	0.987	0.041	0.968	0.511	0.742
Towel	0.949	0.879	0.765	0.881	0.821	0.176	0.672	0.5	0.134
Chopsticks-cover	0.895	0.793	0.532	0.889	0.851	0.005	0.772	0.065	0.028
Towel-dish	0.917	0.543	0.568	0.81	0.713	0.053	0.747	0.344	0.019
Paper	0.931	0.827	0.357	0.618	0.884	0.453	0.726	0.667	0.237

Table 5. Performance using sparse training Method A on FPGA.

Class	Prune30%	Prune50%	Prune80%	Shortcut30%	Shortcut50%	Shortcut80%	Slim30%	Slim50%	Slim80%
mAP	0.922	0.856	0.759	0.892	0.864	0.18	0.838	0.388	0.236
Prune channels	4012	6688	10,700	4618	7692	16,250	7777	12,877	20,633
Prune ratio	0.153	0.254	0.407	0.176	0.292	0.618	0.296	0.49	0.784
Parameters	40,257,085	26,227,273	9,942,629	37,017,469	21,481,070	6,684,490	21,124,184	6,788,468	6,246,822
Size	42 MB	28 MB	12 MB	39 MB	23 MB	8936 KB	23 MB	9044 KB	8460 KB
Time	10.948 s	8.780 s	6.686 s	10.430 s	8.260 s	6.487 s	8.128 s	7.853 s	8.438 s

Table 6. Detection accuracy using sparse training Method B on FPGA.

Class	Prune30%	Prune50%	Prune80%	Shortcut30%	Shortcut50%	Shortcut80%	Slim30%	Slim50%	Slim80%
Chopsticks-two	0.882	0.887	0.867	0.906	0.907	/	0.944	0.906	0.602
Cup	0.983	0.977	0.948	0.99	0.988	/	0.969	0.833	0.816
Water-cup	0.978	0.975	0.975	0.963	0.979	/	0.972	0.978	0.977
Tea-cup	1	0.988	0.985	0.999	0.999	/	1	0.988	0.906
Waster-paper	0.923	0.961	0.945	0.974	0.989	/	0.992	0.971	0.872
Chopsticks-one	0.943	0.924	0.918	0.951	0.922	/	0.879	0.878	0.718
Wine-cup	1	0.998	1	1	0.998	/	0.998	0.991	0.998
Fish-dish	0.886	0.8	0.837	0.859	0.889	/	0.86	0.434	0.882
Spoon	0.917	0.914	0.882	0.899	0.898	/	0.928	0.915	0.815
Soup-bowl	0.969	0.963	0.967	0.971	0.969	/	0.967	0.971	0.957
Square-bowl	0.966	0.941	0.931	0.953	0.94	/	0.944	0.869	0.651
Rice-bowl	0.986	0.997	0.989	0.987	0.986	/	0.986	0.994	0.818
Towel	0.963	0.913	0.968	0.945	0.938	/	0.965	0.982	0.962
Chopsticks-cover	0.942	0.919	0.972	0.892	0.925	/	0.892	0.895	0.839
Towel-dish	0.998	0.999	0.983	0.999	0.994	/	0.992	0.962	0.912
Paper	0.925	0.931	0.891	0.916	0.951	/	0.929	0.974	0.901

Table 7. Performance using sparse training Method B on FPGA.

Class	Prune30%	Prune50%	Prune80%	Shortcut30%	Shortcut50%	Shortcut80%	Slim30%	Slim50%	Slim80%
mAP	0.954	0.943	0.941	0.95	0.955	/	0.951	0.909	0.852
Prune channels	4012	6688	10,700	5643	9555	19,725	7777	12,957	20,698
Prune ratio	0.153	0.254	0.407	0.215	0.363	0.75	0.296	0.493	0.787
Parameters	40,534,434	26,372,586	9,900,539	35,527,894	18,907,404	2,095,122	23,217,395	9,604,954	5,709,836
Size	42 MB	28 MB	12 MB	37 MB	20 MB	3481 KB	25 MB	12 MB	7793 KB
Time	10.687 s	8.892 s	6.828 s	9.614 s	7.208 s	/	8.440 s	6.692 s	5.925 s

In this research, we mainly evaluate the results on FPGA, which are shown from Tables 2–7. Among these tables, Tables 2, 4 and 6 indicate detection accuracy of 16 different kinds of dishes, respectively. Tables 3, 5 and 7 indicate performance of testing all 68 testing images on FPGA. Many metrics are used for comparison of performance: mAP, prune channels, prune ratio, parameters, size, and time. Our aim is to achieve acceleration of the model as much as possible, while maintaining the detection accuracy, so it is ideal to obtain the minimum of time, while maintaining the maximum of mAP. We use the method of pruning for acceleration, so when prune channels and prune ratio are larger, parameters and size are smaller, the time will be smaller. However, since there is no linear relationship between these metrics, we record them all in the research process, so as to find the pruned model with the highest comprehensive performance.

6. Analysis and Discussion

In this section, we analyze the experimental results, and discuss the limitations and future directions of this research.

For the original model trained under darknet, we test it under CPU, and it takes a total of 1242.554 s to test all 68 images. However, it takes only 14.411 s to test all 68 images under FPGA, as shown in Table 3, which is 86 times faster than CPU. This result shows that FPGA can greatly improve the speed of deep learning compared to CPU.

6.1. Analysis on Sparsity Training

By making comparisons on the results in general, the performance of sparse training Method B is better than sparse training Method A. In terms of the mAP for both methods:

- If we use sparse training Method A, then no matter which pruning method is used, the final mAP will drop significantly as the pruning ratio rises—especially when the Slim pruning method is used, only when the pruning ratio is set to 30%, it can achieve mAP of more than 80%, and when the pruning ratio is 50% or 80%, the mAP cannot even reach 40%, indicating that the model accuracy has been significantly affected.

- If we use sparse training Method B, we can find that the decrease of mAP is not obvious; even when the Slim method is used and the pruning ratio is set to 80%, which has the largest decrease of accuracy, its mAP only decreases about 11% and is much better compared with sparse training Method A.

It can be seen that sparse training is very critical, and different parameter setting during sparse training affects the effect of pruning greatly. In fact, sparse training means the balance of accuracy and sparsity, and it is worth studying how to find a good strategy to make the sparse model achieve high accuracy as well as high sparsity.

According to the experience, for the coefficient s multiplied on the scaling factor γ, the large s generally sparse trains the model faster but the accuracy drops fast, and the small s generally sparsely trains the model slower, but the accuracy drops slowly. In addition, with the large learning rate, the model will be sparse trained fast, and the small learning rate in later stage will help the accuracy to rebound. In this research, we only adjusted the learning rate but did not optimize s. Better pruning results should be obtained if s is adjusted appropriately, for example, by setting a dynamic s so that it changes during different training process.

6.2. Analysis on Pruning Methods

For the results of the sparse training Method B, it can be found that different pruning methods also influence on the mAP obviously.

Pruning method is the most conservative method and can not reduce the size of the model as much as the latter two methods, but the actual comprehensive result is very outstanding. When we use the Pruning method, even if the pruning ratio is set to 80%, the mAP only decreases by less than 3% compared with the original unpruned model, which means the accuracy of the model is basically maintained, while the model size is reduced to 19% of the original model, the number of parameters is reduced to 16% of the original model, and the running time is less than half of the original model. Therefore, in this research, we set Prune80% model to the model with the highest comprehensive performance after pruning.

As for the Shortcut method, although there is an error when the prune ratio is set to 80%, after analyzing the results when the prune ratio is 30% and 50%, the Shortcut method has more pruning intensity and higher performance compared to the Prune method. However, the detection accuracy of these two methods is not so much different, and there are compilation errors that are temporarily unexplained. As a result, the Shortcut method is not as stable as the Prune method.

As for the Slim method, we can find that, with the increase of the pruning ratio, the pruning intensity increases significantly, the model size and the number of parameters decrease obviously, but the model accuracy also drops gradually. In fact, after the fine-tune processing on the software platform, the final detection accuracy of the pruned model returns to the same level as the original model even the slim method is used. However, after deploying the model to FPGA, the detection accuracy decreases significantly.

The Vitis AI quantizer could reduce the computational complexity without losing prediction accuracy; this is only right in the case of unpruned models. After pruning, the model loses some information, and quantization in this case will cause more information loss, leading to the condition that, when the percentage of pruning is too large, the accuracy of the model will not return to the level of the original model, even after fine-tuning.

In order to observe the effect of the reduced accuracy on FPGA more clearly, we use the original model and Prune80% model to detect dishes on each of the 68 testing images, then select two representative images for analysis. As shown in Figure 7, (a) and (b) represent the detection results of testing image A, while (c) and (d) represent the detection results of testing image B. In these four pictures, (a) and (c) show the detection results using the original model, while (b) and (d) show the detection results using the Prune80% model. From Figure 7, we can find that, after pruning, most of the dishes can be detected correctly, but there are still some dishes near the corners that can not be detected correctly, such as

Chopsticks-cover in the upper right corner of (b), and Paper at the top of (d). In addition, there are cases about repeated detection, such as the results of Chopsticks-two near the bottom of (d). In general, even though the Prune80% model has the highest comprehensive performance after pruning, it still has a little lower detection accuracy than the original model. Therefore, in our future research, we will improve the pruning method, so as to enhance the detection accuracy after pruning.

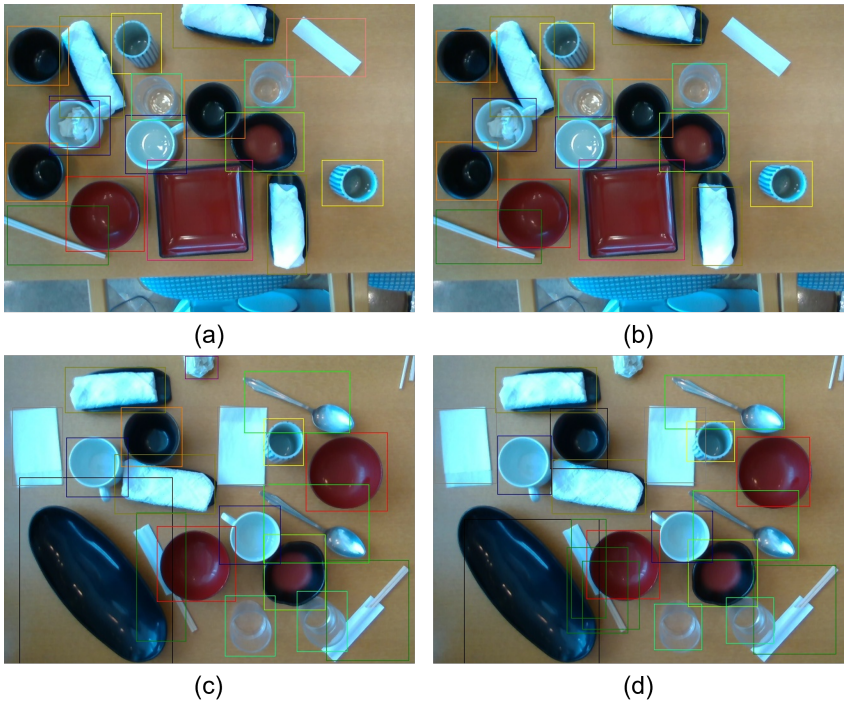


Figure 7. Examples about the effect of the reduced accuracy after pruning. (**a**) detection results of testing image A using original model, (**b**) detection results of testing image A using pruned model, (**c**) detection results of testing image B using original model, (**d**) detection results of testing image B using pruned model.

6.3. Analysis Based on AP

By observing the changes of AP for each kind of dishes after pruning, it can be found that the detection result of each kind of dishes differs.

When the sparse training Method A is used, regardless of which pruning method is used, the detection accuracy of some dishes decreases much more slowly than other kinds of dishes as the pruning ratio increases, such as Cup, Water-cup, Tea-cup, and Wine-cup. In contrast, the detection accuracy of Chopsticks-two, Chopsticks-one, and Chopsticks-cover decreases much more obviously than other kinds of dishes when the pruning ratio increases. A similar pattern is also found when the sparse training Method B is used. The reason for this result is that the number of different kinds of dishes is different in the dataset used this time. The number of each kind of dishes in the training dataset and testing dataset is shown in Table 8, which shows that the number of various cups is significantly larger. This indicates that, during the training process, these kinds of dishes provide more information and the model can learn the features of them better, which makes them to be detected more easily during test process.

The reason for the low detection accuracy of several kinds of dishes related to chopsticks is not only due to the insufficient learning caused by the small quantity, but also the characteristics of these kinds of dishes themselves. For example, Chopsticks-two and Chopsticks-one are both chopsticks, and they are quite similar in appearance, which makes it more difficult for the model to distinguish them during the test process. In addition, as shown in Figure 8, in both images of the training dataset, the items marked by red boxes

are all labeled as Chopsticks-two, but they actually have some obvious differences, partly just chopsticks and partly a combination of chopsticks and cover. In addition, in Figure 3b, Towel-dish is placed below the Towel in the upper right corner, but Towel-dish is not marked during the labeling process, which also affects the learning effect of the model to a certain extent. These labeling problems, which affect the accuracy of the pruned model, will be improved in the future work.

Table 8. Number of each kind of dish in the dataset

Class	Training Dataset	Test Dataset
Fish-dish	224	50
Towel-dish	137	37
Rice-bowl	256	59
Chopsticks-two	346	82
Towel	277	58
Cup	602	153
Spoon	269	62
Soup-bowl	259	62
Water-cup	426	103
Tea-cup	476	112
Waster-paper	287	63
Square-bowl	163	48
Chopsticks-one	138	32
Paper	85	28
Chopsticks-cover	82	19
Wine-cup	516	130

(a) (b)

Figure 8. Examples of Chopsticks-two in the training dataset. (**a**) first example, (**b**) second example.

6.4. Limitations and Future Directions

In this research, there are mainly three aspects of limitations which can be improved, and these are future directions of our work.

6.4.1. Improvement of Detection Accuracy

Although pruning and quantization finally led to an inevitable decrease of detection accuracy, due to time reasons, we still did not find the optimal parameters of sparse training and pruning for our dish dataset. For future research, we plan to adjust the parameters during sparse training and improve the pruning method to alleviate the accuracy decrease.

At the same time, the dish dataset also needs to be improved due to some insufficient consideration when we built the dataset. The improvement of the dataset can also increase the detection accuracy.

6.4.2. Addition of Evaluation Criteria

In this research, we only used our own unique dataset for testing, which leads to a lack of public datasets for testing. Due to this reason, we are currently unable to provide a

statistical test to compare our method with other methods or state-of-the-art. In our future work, we will do some additional experiments on public datasets and compare the results of our method with other methods.

Since we used Vitis AI in this research, we still did not find a way to see quantitative details on the FPGA resource utilization, such as clock speed, number of gates utilized, available resources, and percentage usage. Thus, we will continue to look for ways to see these quantitative hardware metrics when using Vitis AI, and this is one of our future research directions.

6.4.3. Formalizing a Process Flow for Implementing CNN on FPGA

Although we proposed a formalized process of deploying the YOLOv3 model on the FPGA platform, which consists of the software process and hardware process, there are still many CNN models that are able to be accelerated on FPGA. The method is similar to this research: training CNN models and pruning them on the software platform to reduce the amount of parameters firstly, and then deploying them to the FPGA platform through Vitis AI. For a specific model, it is necessary to adjust the methods of sparse training and pruning, to keep the accuracy of the model as high as possible. This is also one of our future research directions.

In addition, since under some conditions, the pruned model can not be successfully deployed to the FPGA platform due to compilation errors, we will solve this problem in our future research.

In addition, considering that 3D coordinate [46] is very important for the application of empty-dish recycling robot, we plan to build a system that can be applied to FPGA, in which the robot can confirm the position of dishes more accurately and recycle them through 3D coordinates. Specifically, we can combine YOLO with other neural network algorithms. First, we use YOLO to detect the 2D coordinates of each dish. Then, we use other neural network algorithms to detect the distance from the robot to the dish, so that we can obtain the 3D position of the dish.

7. Conclusions

In this research, we propose a formalized process of deploying the YOLOv3 object detection model on FPGA, which can detect dishes with acceleration and high accuracy for empty-dish recycling robots. In order to implement the YOLOv3 model on FPGA, we first build up a unique dataset containing 16 types of dishes for model training and testing. Then, we use our unique dataset to train the YOLOv3 model on software platforms, improve the accuracy of the model, and reduce the size of the model through pruning. After the software process, we deploy the pruned model to FPGA through Vitis AI as the hardware process. Compared to the original model, the pruned model with the highest comprehensive performance has the following features: the model size is reduced from 62 MB to 12 MB, which is only 19% of origin; the number of parameters is reduced from 61,657,117 to 9,900,539, which is only 16% of origin; the running time is reduced from 14.411 s to 6.828 s, which is only less than half of origin, while the detection accuracy is decreased from 97% to 94.1%, which is only less than 3%. For future research, we will improve detection accuracy by adjusting the pruning method to alleviate the accuracy decrease, and improve the dish dataset to increase the detection accuracy. We will also add evaluation criteria such as results compared with the ground truth on public datasets, and quantitative details on the FPGA resource utilization. In addition, we will formalize a process flow for implementing CNN on FPGA, and use 3D coordinates to achieve more accurate detection for empty-dish recycling robots.

Author Contributions: Data curation, X.Y.; Investigation, Z.W.; Methodology, H.L.; Project administration, L.M.; Software, Z.W.; Supervision, L.M.; Writing—original draft, Z.W.; Writing—review and editing, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JST, the establishment of university fellowships towards the creation of science technology innovation, Grant No. JPMJFS2146.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and analysed during the current study are available from the corresponding author on reasonable request.

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

References

1. Al-Sharman, M.; Murdoch, D.; Cao, D.; Lv, C.; Zweiri, Y.; Rayside, D.; Melek, W. A sensorless state estimation for a safety-oriented cyber-physical system in urban driving: Deep learning approach. *IEEE/CAA J. Autom. Sin.* **2020**, *8*, 169–178. [CrossRef]
2. Meng, L.; Hirayama, T.; Oyanagi, S. Underwater-drone with panoramic camera for automatic fish recognition based on deep learning. *IEEE Access* **2018**, *6*, 17880–17886. [CrossRef]
3. Lyu, B.; Tomiyama, H.; Meng, L. Frame Detection and Text Line Segmentation for Early Japanese Books Understanding. In Proceedings of the ICPRAM, Valletta, Malta, 22–24 February 2020; pp. 600–606.
4. Yue, X.; Li, H.; Fujikawa, Y.M.L. Dynamic Dataset Augmentation for Deep Learning-based Oracle Bone Inscriptions Recognition. *ACM J. Comput. Cult. Herit.* 2022, *in press*.
5. Abdelaziz, A.H. Comparing fusion models for DNN-based audiovisual continuous speech recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2017**, *26*, 475–484. [CrossRef]
6. Cai, J. A Design of Interactive Online English Translation System Based on Deep Learning. In Proceedings of the 2021 International Conference of Social Computing and Digital Economy (ICSCDE), Chongqing, China, 28–29 August 2021; pp. 8–11.
7. Lin, G.; Wen, S.; Han, Q.L.; Zhang, J.; Xiang, Y. Software vulnerability detection using deep neural networks: A survey. *Proc. IEEE* **2020**, *108*, 1825–1848. [CrossRef]
8. Shaukat, K.; Alam, T.M.; Hameed, I.A.; Khan, W.A.; Abbas, N.; Luo, S. A review on security challenges in internet of things (IoT). In Proceedings of the 2021 26th International Conference on Automation and Computing (ICAC), Portsmouth, UK, 2–4 September 2021; pp. 1–6.
9. Alam, T.M.; Shaukat, K.; Hameed, I.A.; Khan, W.A.; Sarwar, M.U.; Iqbal, F.; Luo, S. A novel framework for prognostic factors identification of malignant mesothelioma through association rule mining. *Biomed. Signal Process. Control* **2021**, *68*, 102726. [CrossRef]
10. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Chen, S.; Liu, D.; Li, J. Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies* **2020**, *13*, 2509. [CrossRef]
11. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Xu, M. A survey on machine learning techniques for cyber security in the last decade. *IEEE Access* **2020**, *8*, 222310–222354. [CrossRef]
12. Duan, X.; Wang, Y.; Kong, X.; Li, M.; Yang, Y. Mechanical design and kinematic analysis of a medical robot assisted maxillofacial surgery. In Proceedings of the 2013 ICME International Conference on Complex Medical Engineering, Beijing, China, 25–28 May 2013; pp. 596–601.
13. Kawashima, K.; Sasaki, T.; Ohkubo, A.; Miyata, T.; Kagawa, T. Application of robot arm using fiber knitted type pneumatic artificial rubber muscles. In Proceedings of the IEEE International Conference on Robotics and Automation—ICRA'04, New Orleans, LA, USA, 26 April–1 May 2004; Volume 5, pp. 4937–4942.
14. Kim, D.; Lee, K.H.; Ji, S.H.; Shon, W.H.; Kim, Y.S. Development of a medical robot system for pedicle screw surgery assisted by fluoroscopic X-ray image. In Proceedings of the Advanced Robotics and its Social Impacts, Menlo Park, CA, USA, 2–4 October 2011; pp. 62–65.
15. Shaukat, K.; Iqbal, F.; Alam, T.M.; Aujla, G.K.; Devnath, L.; Khan, A.G.; Iqbal, R.; Shahzadi, I.; Rubab, A. The impact of artificial intelligence and robotics on the future employment opportunities. *Trends Comput. Sci. Inf. Technol.* **2020**, *5*, 50–54.
16. Setiawan, A.I.; Furukawa, T.; Preston, A. A low-cost gripper for an apple picking robot. In Proceedings of the IEEE International Conference on Robotics and Automation—ICRA'04, New Orleans, LA, USA, 26 April 2004–1 May 2004; Volume 5, pp. 4448–4453.
17. Khuantham, C.; Sonthitham, A. Spraying robot controlled by application smartphone for pepper farm. In Proceedings of the 2020 International Conference on Power, Energy and Innovations (ICPEI), Chiangmai, Thailand, 14–16 October 2020; pp. 225–228.
18. 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.
19. Fukushima, K.; Miyake, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*; Springer: Berlin/Heidelberg, Germany, 1982; pp. 267–285.
20. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
21. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 25.

22. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv*, **2014**, arXiv:1409.1556.
23. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
25. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
26. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
27. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
28. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
29. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016; pp. 21–37.
30. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
31. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
32. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv*, **2018**, arXiv:1804.02767.
33. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv*, **2020**, arXiv:2004.10934.
34. Farabet, C.; Poulet, C.; Han, J.Y.; LeCun, Y. Cnp: An fpga-based processor for convolutional networks. In Proceedings of the 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 31 August 2009–2 September 2009; pp. 32–37.
35. Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; et al. Going deeper with embedded fpga platform for convolutional neural network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016; pp. 26–35.
36. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 243–254. [CrossRef]
37. Fujii, T.; Sato, S.; Nakahara, H.; Motomura, M. An FPGA realization of a deep convolutional neural network using a threshold neuron pruning. In Proceedings of the International Symposium on Applied Reconfigurable Computing, Delft, The Netherlands, 3–7 April 2017; Springer: Cham, Switzerland, 2017; pp. 268–280.
38. 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 Great Lakes Symposium on VLSI 2021 (the 31st GLSVLSI), Virtual, 22–25 June 2021.
39. 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. *Comput. Intell. Neurosci.* **2022**, *2022*, 8039281. [CrossRef] [PubMed]
40. Chen, Y.H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits* **2016**, *52*, 127–138. [CrossRef]
41. Shaukat, K.; Luo, S.; Chen, S.; Liu, D. Cyber threat detection using machine learning techniques: A performance evaluation perspective. In Proceedings of the 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 20–21 October 2020; pp. 1–6.
42. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2736–2744.
43. Xilinx Inc. 2021. Available online: https://docs.xilinx.com/r/1.3-English/ug1414-vitis-ai (accessed on 1 April 2021).
44. 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. [CrossRef]
45. 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, Jeju Island, Korea, 4–7 May 2022.
46. Yin, X.; Sasaki, Y.; Wang, W.; Shimizu, K. 3D Object Detection Method Based on YOLO and K-Means for Image and Point Clouds. *arXiv*, **2020**, arXiv:2005.02132.