mathematics

MDPI

Article

YOLOv4-Driven Appearance Grading Filing Mechanism: Toward a High-Accuracy Tomato Grading Model through a Deep-Learning Framework

Yu-Huei Cheng [1], Cheng-Yen Tseng [2], Duc-Man Nguyen [3] and Yu-Da Lin [4,*]

[1] Department of Information and Communication Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan
[2] Department of Applied Chemistry, Chaoyang University of Technology, Taichung 413310, Taiwan
[3] International School, Duy Tan University, Danang 550000, Vietnam
[4] Department of Computer Science and Information Engineering, National Penghu University of Science and Technology, Penghu 880011, Taiwan
* Correspondence: yudalinemail@gms.npu.edu.tw

Abstract: In traditional agricultural quality control, agricultural products are screened manually and then packaged and transported. However, long-term fruit storage is challenging in tropical climates, especially in the case of cherry tomatoes. Cherry tomatoes that appear rotten must be immediately discarded while grading; otherwise, other neighboring cherry tomatoes could rot. An insufficient agricultural workforce is one of the reasons for an increasing number of rotten tomatoes. The development of smart-technology agriculture has become a primary trend. This study proposed a You Only Look Once version 4 (YOLOv4)-driven appearance grading filing mechanism to grade cherry tomatoes. Images of different cherry-tomato appearance grades and different light sources were used as training sets, and the cherry tomatoes were divided into four categories according to appearance (perfect (pedicled head), good (not pedicled head), defective, and discardable). The AI server ran the YOLOv4 deep-learning framework for deep image learning training. Each dataset group was calculated by considering 100 of the four categories as the difference, and the total numbers of images were 400, 800, 1200, 1600, and 2000. Each dataset group was split into an 80% training set, 10% verification set, and 10% test set to overcome the identification complexity of different appearances and light source intensities. The experimental results revealed that models using 400–2000 images were approximately 99.9% accurate. Thus, we propose a new mechanism for rapidly grading agricultural products.

Keywords: deep learning; quality control; deep image learning; YOLOv4

MSC: 68U10

Citation: Cheng, Y.-H.; Tseng, C.-Y.; Nguyen, D.-M.; Lin, Y.-D. YOLOv4-Driven Appearance Grading Filing Mechanism: Toward a High-Accuracy Tomato Grading Model through a Deep-Learning Framework. *Mathematics* **2022**, *10*, 3398. https://doi.org/10.3390/math10183398

Academic Editor: Ioannis E. Livieris

Received: 24 August 2022
Accepted: 15 September 2022
Published: 19 September 2022

1. Introduction

Fruits that cannot be stored for a long time, such as cherry tomatoes, are prone to rot owing to tropical climates [1]. Failure to screen such fruits in time could lead to severe problems. Once a rotten fruit is identified, it needs to be immediately removed while grading; otherwise, the entire batch of fruit is prone to rot [2]. This situation becomes more severe with an insufficient agricultural workforce.

Deep learning [3] has become a popular artificial intelligence technique for solving engineering problems [3–6]. Convolutional neural network (CNN) is a class of deep neural networks most commonly used to examine visual images, such as safe prediction and assessment of sports injuries [7], disease prediction [8,9], non-intrusive load monitoring [10], and nitrogen deficiency prediction of rice crops [11]. To study tomato leaf diseases, Robert et al. proposed an object-detection algorithm based on a faster region-based convolution neural

network (R-CNN). The region proposal network (RPN)-based faster R-CNN uses feature maps as input and produces a series of object targets to detect tomato leaf diseases [12]. Anirudh et al. used Xception combined with YOLO version 3 (YOLOv3) to replace the original Inception v3, utilizing deep separable convolution to classify tomato leaf wilt in India [13]. Additionally, Piyush et al. proposed an R-CNN algorithm based on deep learning to correctly identify infected areas of tomato leaf diseases in India. In this manner, they helped farmers diagnose diseases and intervene in time to improve the survival rate of crops [14]. Choi et al. proposed a CNN based on Inception-ResNet v2 to identify mineral nutrients from growth images of tomato plants in a greenhouse. Predicting the lack of mineral nutrients in tomatoes can improve their yield and prevent diseases caused by such a lack [15].

Jiang et al. proposed Resnet-50 as the fundamental network model in China. They changed the activation function of the network to Leaky-ReLU to identify the disease characteristics on the leaf surface. Their model can successfully predict leaf spot blight, late blight, and yellow leaf roll [16]. Surampalli et al. proposed a CNN to detect tomato leaf diseases and mitigate India's heavy crop losses caused by plant diseases and pests. They used an open-source algorithm for image segmentation and image processing technology to identify tomato leaf diseases [17]. Azeddine et al. proposed an intelligent mobile application model based on a deep CNN to identify ten types of tomato leaf diseases [13]. Qimei et al. proposed a tomato disease detection method based on a faster R-CNN in China. To identify different tomato diseases effectively, they combined faster R-CNN with different CNNs (including vgg16, resnet50, and resnet101) [18]. Jiayue et al. proposed a target detection algorithm based on a regression model to improve tomato yield and promote the control of tomato fruit quality in China. The author used YOLO version 2 (YOLOv2) to detect healthy tomato fruits with common physiological diseases [19]. Bhole and Kumar proposed an automatic mango fruit grading system that uses non-destructive techniques such as thermal imaging and transfer learning with a pretrained SqueezeNet model [20]. Hidetomo et al. proposed an acoustic detection technology for estimating the shelf life of fruits and vegetables in Korea. Consumers often decide the cooking method according to the maturity of vegetables or choose better vegetables in supermarkets based on their freshness. The authors scanned the audible frequency band of the samples using an acoustic probe and captured the transmitted acoustic signal using a microphone. Transmission signals collected from the samples under various storage conditions were used to train the classifier. They found that the transmission signal amplitude decreases significantly with the maturity of tomatoes [21].

Current research has focused on tomato plants, leaf diseases, mineral nutrition, and other areas. However, research on the image recognition of fruit appearance defectives remains an important research topic. This study collected and labeled images of several cherry tomatoes for cherry tomato quality. We proposed a YOLO version 4 [22] (YOLOv4)-driven appearance grading filing mechanism for image deep learning training to grade cherry tomatoes accurately.

2. Methods

Due to defective features, traditional machine learning can be challenging to model and migrate. The advantages of deep learning, such as YOLOv4, can avoid the complex process of image preprocessing and reduce the complexity of the network, especially for multiple-input vectors. YOLOv4 improves the shortcomings of the traditional manual method, achieves more efficient defective detection, and can effectively avoid the complexity of data reconstruction in feature extraction and classification. We used the YOLOv4 deep-learning framework for image deep-learning training to accurately grade cherry tomatoes (Figure 1). The YOLOv4 deep-learning framework can be divided into four steps: data collection, deep-learning framework development, cherry tomato quality grading, and the actual test.

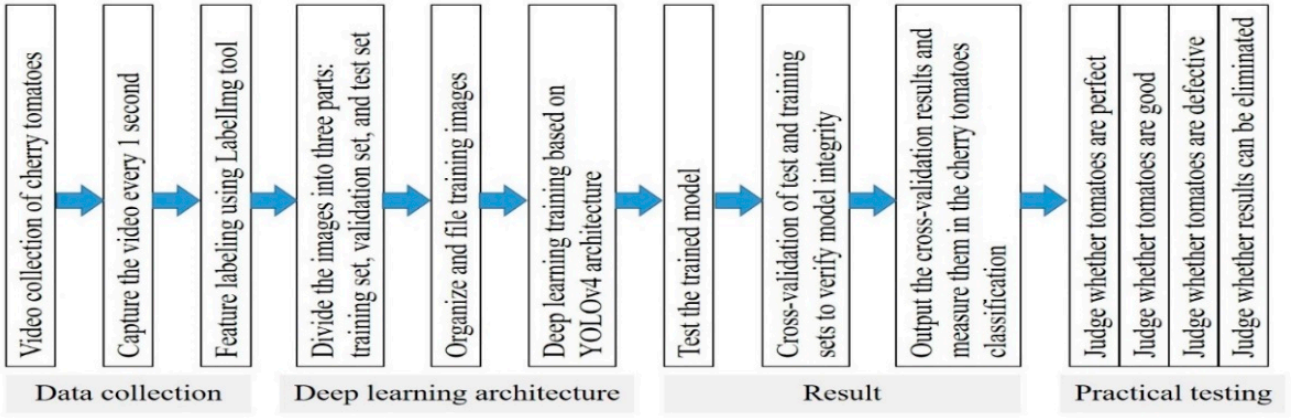

Figure 1. Flowchart of cherry tomato grading.

2.1. Data Collection

We collected various cherry tomato videos from tomato orchards. Cherry tomatoes were manually placed at different angles to completely record their appearances with varying face shapes during the recording process. Each video was converted into images every 1 s to obtain 2000 cherry tomato images. Based on expert-assessed appearance, the images were classified into four categories: perfect (pedicled), good (not pedicled), defective, and discardable. Perfect tomatoes are pedicled cherry tomatoes that are not damaged in appearance. Cherry tomatoes that are not pedicled and not damaged in appearance are good. Cherry tomatoes with a damaged appearance are defective and can be made into ketchup, canned tomatoes, etc. Rotten, moldy, and cracked cherry tomatoes were classified for culling. We stacked 100 images per category, thus obtaining datasets of sizes 400, 800, 1200, 1600, and 2000. We used image annotation software to frame the bounding boxes of the cherry tomato images and define their feature categories. After the images were labeled, the corresponding XML files were generated, including the category names and coordinate values.

2.2. Deep-Learning Architecture

A CNN is a feedforward neural network whose artificial neurons can satisfy the requirement of partially covering the surrounding cells. The CNN can effectively exploit the two-dimensional structure of the input data. Its internal structure includes one or more convolutional layers, a fully connected layer (corresponding to a classical neural network), correlation weights, and pooling layers. The backpropagation algorithm trains the CNN. This model performs better than other deep feedforward neural networks, especially in image and speech recognition. We divided the images of the cherry tomatoes into training, validation, and test sets. The training set was mainly used for model fitting and training in the training phase; it directly participates in the model parameter adjustment. The validation set was used to assess the initial capabilities of the model and provide a basis for hyperparameter tuning during training. The test set was the final generalization used to evaluate the model. The model training diversity description split the images proportionally into an 80% training set, 10% validation, and 10% test set. During the model training process, the convolutional layer comprised a set of feature maps formed by sliding different convolution kernels on the input image and performing certain operations. The factor stride controls the size of the output feature map. At each sliding position, the convolution kernel and the input cherry tomato image perform the corresponding product and sum operations, projecting the information of the image onto the elements in the feature map. The convolution kernel size is smaller than the input image; it overlaps or is parallel to the input image. The convolution kernel computes all elements in the feature map, and the feature maps have the same weights.

A convolution is a nonlinear form of downsampling. This study used max pooling, which divides the input cherry tomato image into rectangular regions and outputs the maximum value of each sub-region. The pooling layer continuously reduces the data space size to reduce the overall number of parameters and calculations. Pooling layers were inserted between the convolutional layers, reducing the edge sensitivity of the convolutional layers. The step size was two, and the pool size was varied according to the step size. A 2×2 max-pooling layer divides every two elements from the image into 2×2 memory blocks and considers the maximum value. During the deep-learning training, multiple convolutions, max-pooling layers, and high-level inference were fully connected. In convolution, the neurons of a fully connected layer are related to all the activations of the previous layer. In a fully connected layer, all the neurons have weighted connections. The convolutional layer captures sufficient input features to recognize the input image of cherry tomatoes, and a model can be trained to recognize cherry tomatoes.

We employed the YOLO technology for real-time object detection to achieve real-time automatic defective detection for cherry tomatoes. YOLO was developed into the fourth version (YOLOv4). After the camera lens captured the cherry tomato defects, the processing method of the cspparknet53 framework of YOLOv4 was run through the AI server. YOLOv4 has the characteristics of real-time object detection, tracking, and judgment. The object-detection framework is illustrated in Figure 2. During training, the image input of cherry tomatoes was generated into a one-stage detector through four stages of input, trunk, neck, and sensory prediction, and then combined with spark prediction to form a two-stage detector object-detection framework. As shown in Figure 3, the network architecture of YOLOv4 comprises a CBM, CBL, res unit, cspx, and spp. The CBM is the smallest element in the network structure and comprises the Conv + BN + Mish activation function. The difference between CBL and CBM is that the activation function uses leaky_relu. The Res unit refers to the residual structure in the RESNET network so the network can be built deeper. Furthermore, cspx is based on the cspnet network structure and comprises three convolutional layers and an X res unit module. Spp uses 1×1, 5×5, 9×9, and 13×13 multi-scale fusion and concatenates the feature maps of different kernel sizes as the output. The concept is tensor splicing, which expands the dimensions of two tensors, such as $26 \times 26 \times 256$ and $26 \times 26 \times 512$, and after splicing the two tensors, the output result was $26 \times 26 \times 768$. Moreover, add is the addition of tensors without dimension expansion, such as adding $104 \times 104 \times 128$ and $104 \times 104 \times 128$, where the result is still $104 \times 104 \times 128$.

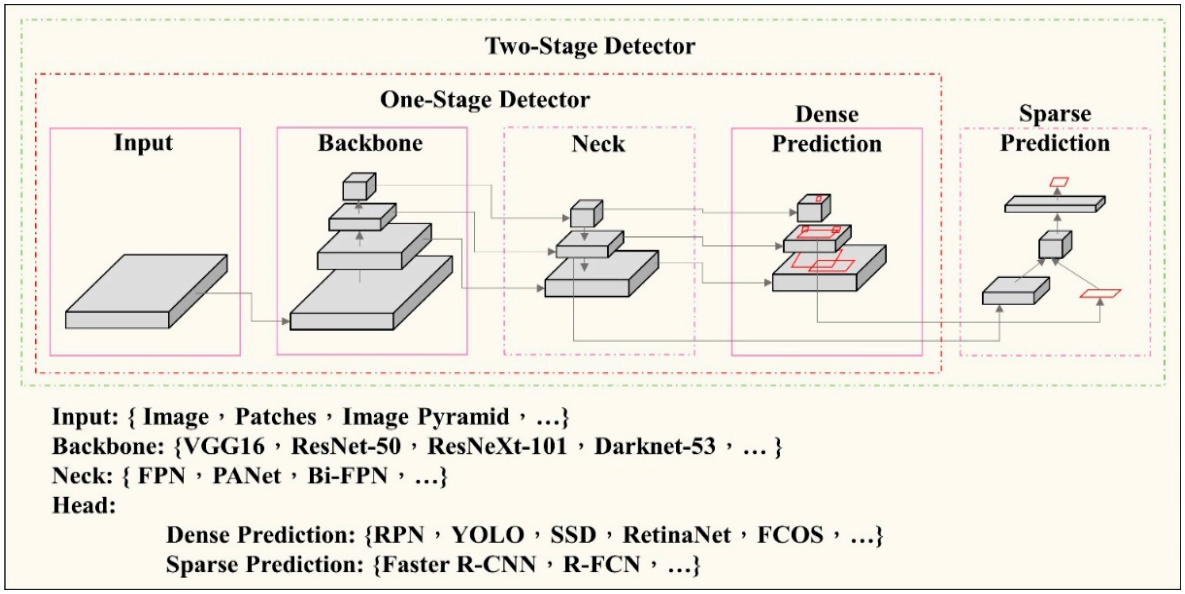

Figure 2. Object-detection framework.

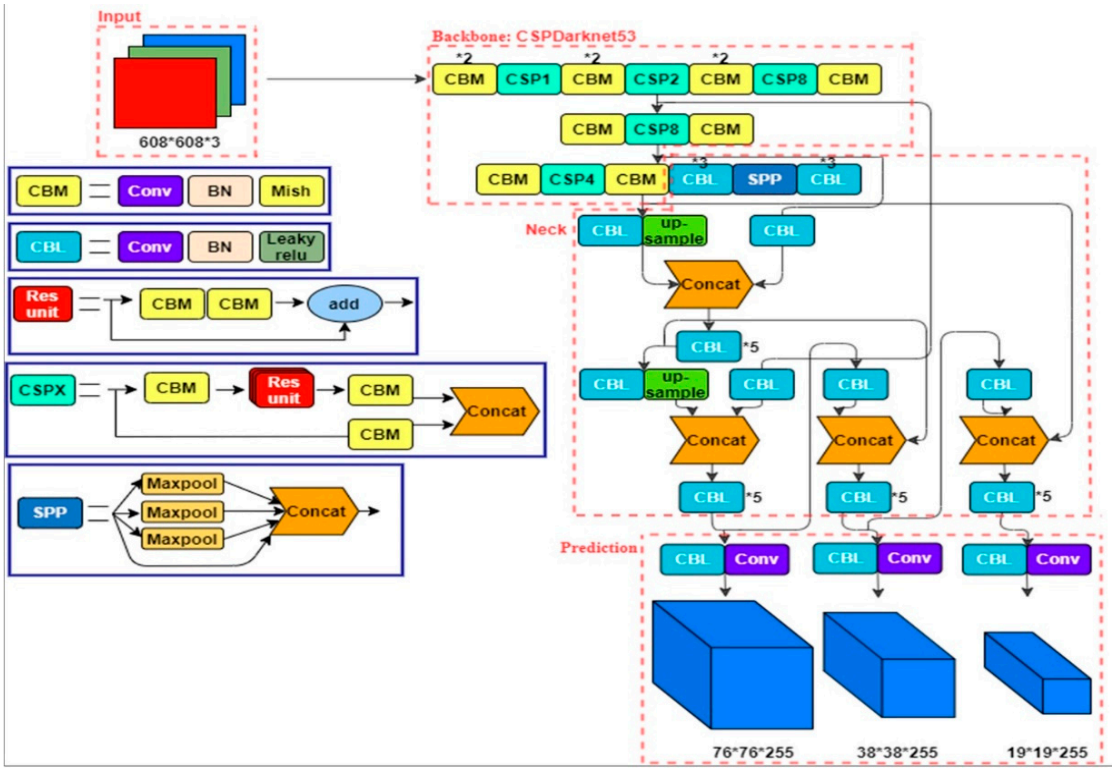

Figure 3. YOLOv4 network architecture (*2 on the CBM layer indicates that two CBM layers are used; *3 on the CBM layer indicates that three CBM layers are used; *5 on the CBM layer indicates that five CBM layers are used.).

The image input of cherry tomatoes first passes through the csppark-net53 memory block of the YOLOv4 network architecture in Figure 3, that is, the backbone memory block in object detection, to build and learn a deep network, including the five items mentioned above: CBM, CBL, res units, cspx, and spp. We performed tensor splicing and Concat to expand the Jade Girl Tomato image dimension and performed image splicing to improve the training speed quickly. In the cspparknet53 memory block of the YOLOv4 network architecture, the CBM module adopts the Mish activation function. The Mish activation function is a smooth nonmonotonic activation function, which is defined by Equation (1):

$$f(x) = x \cdot \tanh(\varsigma(x)) \quad (1)$$

where $\varsigma(x) = \ln(1 + e^x)$ is the softmax activation function, sum. After normalizing the output, each cell's output was compressed between 0 and 1. The sum of the outputs is equal to one, and the outputs are equal to the classification probability distribution representing the probability of each cherry tomato class. The Mish activation function can effectively improve the training stability and average accuracy. The ReLU activation function is shown in Equation (2):

$$f(x) = \max(0, x) \quad (2)$$

Because the output of the ReLU activation function is not centered around zero, only one positive gradient exists. When $x > 0$ in forwarding propagation, the backpropagation process transfers the upstream value to the downstream value. Conversely, the signal traveling downstream in the backpropagation stops if the input $x \leq 0$ in the forwarding propagation. Thus, the training weights cannot be updated, and the network cannot learn. Therefore, a ReLu activation function was used at the neck layer to avoid stopping the training at the backbone layer. The Mish activation function is characterized by low computational cost, smoothness, nonmonotonicity, upper unbounded, and lower bound.

A comparison revealed that the accurate value of Mish in each model was higher than that of Swish and ReLu. The complexity of the Mish activation function increased only slightly (Mish requires approximately 1 s more per epoch than ReLU). When the ReLU activation function is selected, the gradient may explode, the output has no upper limit, and the model cannot converge rapidly. Therefore, in YOLOv4, the Mish function fits into the backbone and is the most helpful for model training on cherry tomatoes. CSPDarknet53 is a convolutional neural network and backbone network for object detection using DarkNet-53. Due to the improvement of the detector MAP index, a backbone with strong image feature extraction ability has been considered, which should not be too large and affect the detection speed. The Bochkovskiy study showed that the CSPDarknet53 neural network is the optimal model for a detector's backbone [22]. Therefore, CSPDarknet53 was chosen as the backbone of YOLOv4, as shown in Table 1. The advantages of using the Mish activation function instead of the activation function in the backbone are outlined in Table 1. Compared with YOLOv3, YOLOv4 can be lightweight, enhance the learning ability of CNN, help the training of the cherry tomato model, and improve the recognition accuracy of multi-input cherry tomato images.

Table 1. Comparison of CSPResNet50, CSPDarknet53, and Efficient-B3 (The bold indicates the best value in parameter aggregation.).

Backbone Model	CSPResNet50	CSPDarknet53	Efficient-B3
Input network resolution	512 × 512	512 × 512	512 × 512
Receptive field size	425 × 425	725 × 725	**1311 × 1311**
Parameters	20.6 M	**27.6 M**	12.0 M
Average size of layer output (W × H × C)	**1058 K**	950 K	668 K
BFLOPs (512 × 512 network resolution)	31 (15.5 FMA)	**52 (26.0 FMA)**	11 (5.5 FMA)
FPS (GPU RTX 2070)	62	**66**	26

Figure 4 presents a schematic of the mosaic data enhancement. The mosaic data augmentation process extends the CutMix [23] method to generate a new data augmentation algorithm that differs from the CutMix method's overlay fusion of two images. Instead, it uses four images for clipping and stitching to form a new image [24]. When we input multiple cherry tomato images, we obtained batches of data from the total dataset and then cut and stitched each cherry tomato image at random positions. Four cherry tomato images were randomly chosen, cut, stitched at any position, and synthesized into new images repeated for batch size times. Finally, a new batch-sized cherry tomato image augmented by mosaic data was obtained and sent back to the neural network for training. This procedure enriches the background of the detected object, adds several small targets during random scaling, maintains the characteristics of the input image, effectively reduces the demand for the GPU, and speeds up the training time of the model.

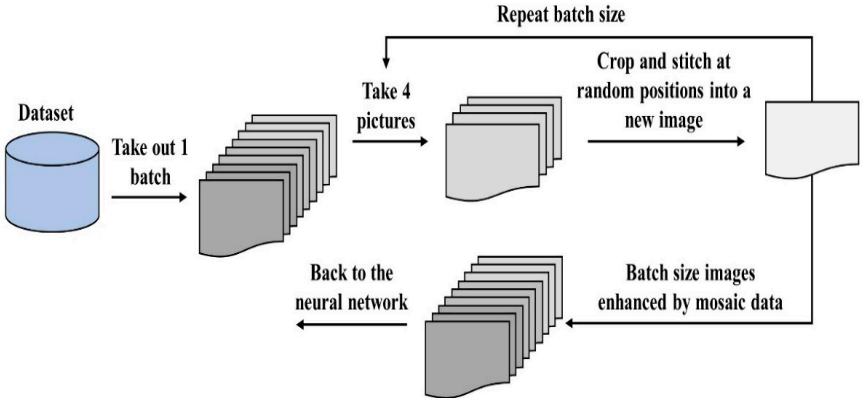

Figure 4. Mosaic data enhancement.

3. Results and Discussion

The experimental environment for this study includes an AI server. The server model, ASUS ws880t, supports the ECC Intel ® Xeon ® W workstation with Ubuntu 20.04. The processor is an Intel Xeon w-2133 with 64 GB memory and three NVIDIA RTX 2080ti graphics processors. The three graphics cards use a tu102 core and have 34 SM units, 11 GB gddr6 VRAM, 4352 CUDA cores, 272 shadow units (TMU), 88 raster (ROP) units, and 13.4TFLOPS single floating-point performance. All support the NVLink multi-card series. The cherry tomato identification model can be trained more quickly to maximize computational performance.

We collected and filed images using YOLOv4. Each group of datasets was calculated by superimposing 100 images of their categories as the difference, and there were five groups of datasets in total. The total images taken were 400, 800, 1200, 1600, and 2000. Each group of datasets was split into an 80% training set, 10% verification set, and 10% test set. The training times of each data group were mainly 10,000 times, while overcoming the identification complexity caused by different appearances and different light sources, avoiding the situation of detailed parameters or different settings in each dataset, reducing human factors, and improving the accuracy of the experiment. Then, the five groups of datasets were trained using the abovementioned deep-learning method. After the training, five models suitable for identifying the appearance of cherry tomatoes were generated, and the models were tested individually. First, the models were applied to the original data verification set to verify data integrity. The cross-validation results after verification were the output. We used the newly obtained cherry tomato images in the test set to test the weight of the trained model, measure the classification ability of the cherry tomato defective detection model, and accurately identify the perfect, good, defective, and discardable categories.

We ran these five datasets on an AI server for in-depth learning training using the abovementioned experimental process. We set the parameters of each group to be consistent to reduce the impact of different parameters on the experimental results and avoid affecting the accuracy and reliability of the experiment. The experimental results are presented in Figure 5A. The experimental results revealed that 400 images of cherry tomatoes were input, 100 images of each category. From the experimental data, we can see that when the training time was approximately 1000–2500 times, the floating range of the mAP is extensive, and while the AVG loss decreased in a curve, it finally became 0.5080. In the process of curve decline, continuous cutting, and splicing at random positions to synthesize new images for training and then repeating batch size times, the mosaic data enhancement mentioned in the previous paragraph improves the generalization ability. However, when the number of training times reached 3000, the mAP continued to be 100%, indicating that the model was overfitted during training. Therefore, we added 100 images to each category (perfect, good, defective, and discardable) to determine whether the experimental results differed.

Figure 5B shows the experimental result after inputting 800 images of cherry tomatoes and 200 images of each category. The experimental data showed that when the number of training times reached 4000, the mAP continued to be 100%, and the AVG loss was 0.4721. Therefore, the model was still overfitted during the training. Hence, we added 100 images (perfect, good, defective, and discardable) to each category to determine whether the experimental results differed and whether fitting would occur. In Figure 5C, the experimental results revealed that 1200 images of cherry tomatoes were input, including 300 images of each category. The experimental data showed that the AVG loss was 0.9009, and the mAP was 100%. According to the training results, the model was still overfitted during training. However, in this training process, when the training time was approximately 8000, the AVG loss value suddenly increased to approximately 3.00, and the mAP decreased to 99%. Therefore, it was inferred that if the number of images increases, there is a chance of reducing the occurrence of overfitting. Thus, 100 images (perfect, good, defective, and discardable) were added to each category to ensure different results. Figure 5D shows the experimental result after inputting 1600 images of cherry tomatoes, 400 images of each

category. The experimental data showed that when the number of training times reached 4500, the mAP continued to be 100%, while the AVG loss was 2.1369. The model was still overfitted during training. Therefore, we added 100 images (perfect, good, defective, and discardable) for each category to the last data group to determine whether the experimental results differed and whether fitting would occur. Figure 5E shows the experimental result when 2000 images of cherry tomatoes were input, that is, 500 images of each category, as follows: the mAP was 99.7%, and the AVG loss was 0.7813. Therefore, it can be inferred that if we continue to increase the number of images of cherry tomatoes, we may effectively reduce overfitting.

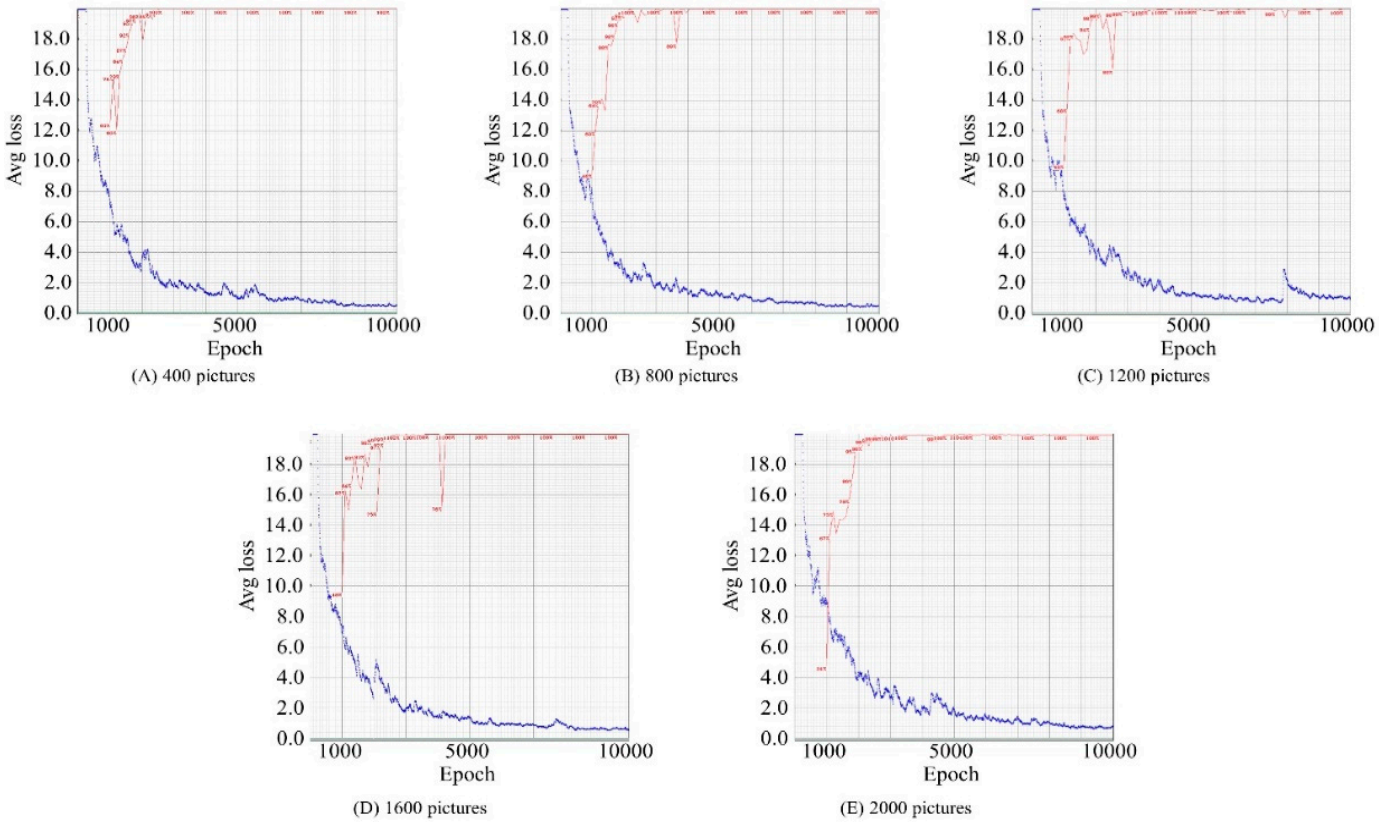

Figure 5. Training results. The blue line is the AVG loss and red line is the mAP.

Several hyper-parameters can be adjusted in the cfg file of YOLOv4. The hyper-parameters were tuned based on rules of thumb by try and error. The width and height of the normalized image size must be a multiple of 32. Here, we used a standard resolution size, 416 × 416. Batch is a one-time input batch size data for model training. If the dataset is relatively small, the Full Batch Learning can be used to better represent the sample population, so as to more accurately orient to the direction of the optimal value. For large datasets, it is not feasible to load all the data at once due to memory constraints. Therefore, a smaller batch size needs to be set. Subdivisions are used to subdivide the data thrown into the memory. When the GPU memory is not enough, this parameter can be adjusted to increase the amount of batch input, so as to better face the direction of the optimal value. Each batch represents the completion of one weight parameter update, and max_batches is the maximum number of weight updates, and the training is stopped after max_batches of weight updates are completed. The setting of max_batches will affect the results of model training. If max_batches is too small, the optimal value may not be found. If max_batches is too large, computing resources will be wasted. Steps can change the learning rate after a certain number of iterations. It helps the training process to escape the local optimum. It is usually recommended to increase the max_batches gradually by 70%, 80%, 90%, etc. Other hyper-parameters use YOLOv4 preset parameters. The experimental parameters of the five

groups were set to be consistent to avoid deviations in the experimental results owing to different parameters, as shown in Table 2. This table lists the CFG parameters and various information sets for each group of experimental data. First, the width and height denote the width and height of the input image, respectively. Furthermore, batch and subdivisions mean that 64 images were loaded simultaneously. Forward propagation was completed 16 times. After the forward propagation of 64 images each time, the backpropagation and update were completed once, and max batches represent the training times of the model. Here, we uniformly trained 10,000 times, and the steps were max_batches as the benchmark, multiplied by 0.8 and 0.9, respectively. These classes comprise several training categories. The images of cherry tomatoes were classified into four categories: perfect, good, defective, and discardable; the classes were set as 4, and the allocation ratios of training percentage, verification percentage, and test percentage were 80%, 10%, and 10% of the mainstream. The experimental results and times required for the different numbers of images are evident in Figure 6. According to the experimental results, we can infer that overfitting occurred in the training process of the five datasets, that is, with the 400, 800, 1200, 1600, and 2000 tomato images. When we trained the model, we mainly used the model to fit the input cherry tomato images and let the model correctly predict the samples that did not appear in the training set. The model's generalization ability is the prediction ability of the samples outside the training set. When the model can capture the characteristics of a single sample of the training set and consider it as a "general law", or when the model has too many iterations and insufficient training data, it cannot find the generalized characteristics. Therefore, the model tends to "memorize and remember" the training data rather than "learn the law" from the training data. This reduces the model's generalization ability, which is known as overfitting. Therefore, it is speculated that due to the lack of training data and our five groups of data sets, the images are of a single category (cherry tomato), which is difficult to generalize, thereby resulting in the easier fitting of the model in the training process.

Table 2. Parameters and experimental results of each group.

CFG Parameters and Model Training	Number of Images				
	400	**800**	**1200**	**1600**	**2000**
Width	416	416	416	416	416
Height	416	416	416	416	416
Batch	64	64	64	64	64
Subdivisions	16	16	16	16	16
Max_batches	10,000	10,000	10,000	10,000	10,000
Classes	4	4	4	4	4
Steps	8000 and 9000	8000 and 9000	8000 and 9000	8000 and 9000	8000 and 9000
Training percentage	80%	80%	80%	80%	80%
Validation percentage	10%	10%	10%	10%	10%
Test percentage	10%	10%	10%	10%	10%
mAP	100.0%	100.0%	99.8%	100.0%	99.7%
Avg loss	0.5080	0.4721	0.9009	2.1369	0.7813
Environment			AI server (Ubuntu 20.04)		
Time	2.01 h	2.58 h	2.98 h	3.19 h	3.6 h

Our fitting model mainly predicts unknown results (those not in the training set); it classifies cherry tomatoes that are not present in this dataset. Therefore, to prevent overfitting, we can avoid it in the following ways: increasing the number of images, data enhancement, dropout, and early stopping. Data enhancement can be achieved using geometric, color, rotary reflection, and composite images. Limited input data can manually generate more data and expand the training set. Dropout is a typical deep-learning technique. In neural networks, some neurons (including hidden and visible neurons) are randomly discarded to reduce the dependence of the model on each neuron and affect the convergence of the model. Because YOLOv4 belongs to supervised learning,

we can set early stopping to solve when we encounter considerable iterations and long training times. Observing the dataset loss change during training can determine the timing of the early ending of the training. Another application of YOLOv4-driven appearance grading filing mechanism could be for general object grading, which can be used in various agricultural, augmented reality [25], and virtual reality applications [26].

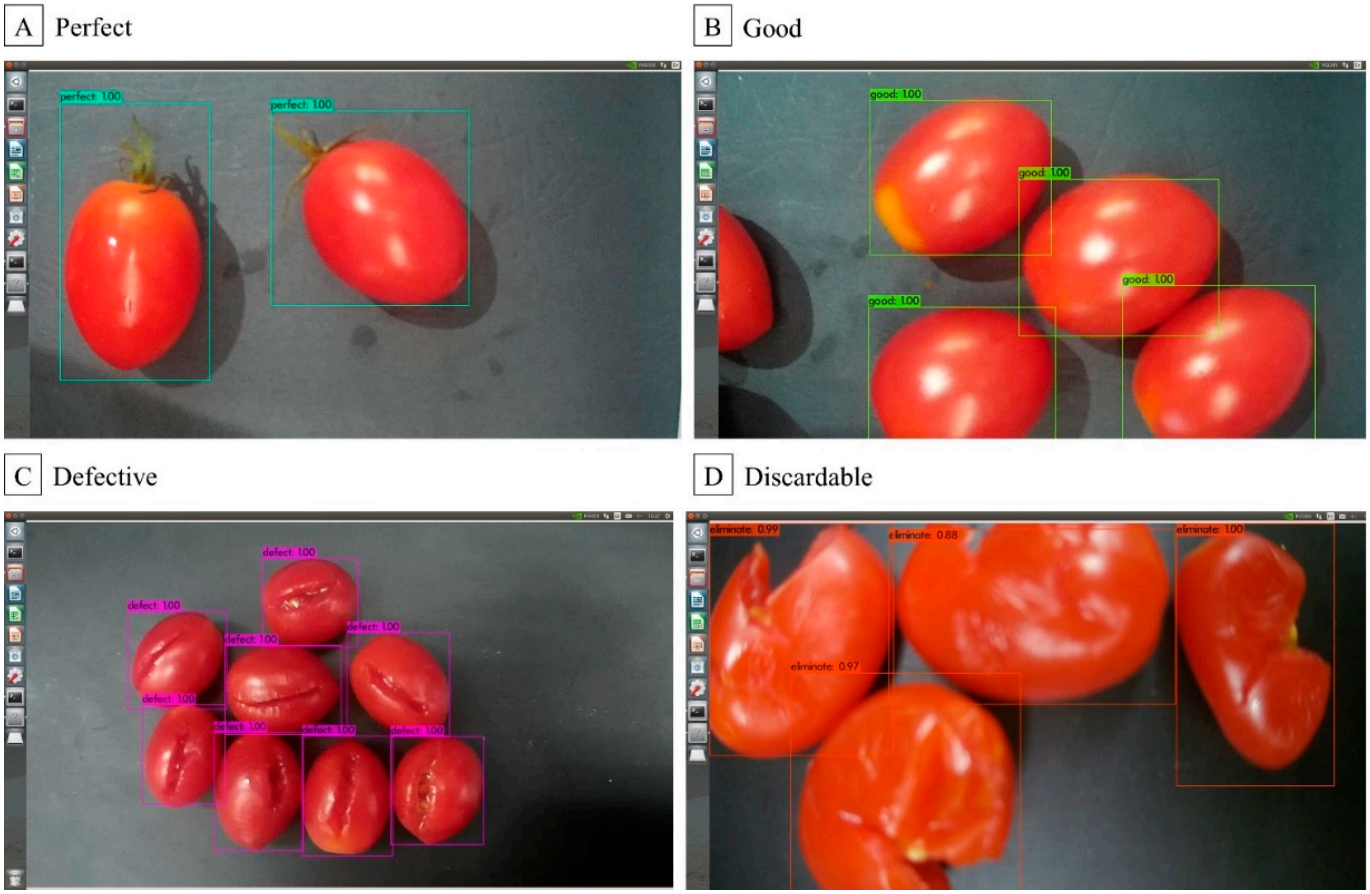

Figure 6. Result of YOLOv4-driven appearance grading filing mechanism.

The machine learning to classify fruits has become a vital topic. The grading filing mechanism based on machine learning can be used for grading when grading standard rules are defined according to fruit features. This mechanism enables fast execution, saves time, and reduces manual labor for automated sorting systems. Naik and Patel reviewed the literature on fruit classification and grading [27]. They explored the classification and grading capabilities of popular machine learning methods such as artificial neural networks [28], biogeography-based optimization [29], support vector machines [30], and principal component analysis [31] under different fruits. Their results confirmed that machine learning could achieve fruit detection, tracking, and estimation. Furthermore, machine learning can detect multiple objects in the same frame simultaneously without consuming much extra time. However, the above algorithms required the experience for data training, and it takes a lot of time to collect and train data to train the model. Most of these algorithms aim at approximate functions. In the color-based fruit classification problem, when fruits of the same shape differ only in the color threshold, the algorithm may not be as expected [32], and operation speed remains an important challenge. In order to enable the future application of an automatic cherry tomatoes quality classification system suitable for factories, we combined the recognition speed of machine learning techniques using the YOLOv4 network architecture. YOLOv4 is optimized for fast running speed and parallel computation of object detectors in production systems rather than

a low computation volume theoretical indicator [22]. Therefore, the designed objects can be easily trained and used. YOLOv4 has real-time, high-quality, and convincing object detection results for anyone using traditional GPUs for training and testing. Most studies have successfully used YOLOv4 for fruit location identification and ripening stage identification [33]. However, only a few studies examined the results of YOLOv4 for identifying fruit quality. In terms of identification accuracy and efficiency, the YOLOv4-driven appearance grading filing mechanism was based on the parameter model suggested by the YOLOv4 literature, which can achieve good results in grading cherry tomatoes. However, many improved versions of YOLOv4 can still be used in more complex fruit species. The ablation study can effectively improve the accuracy of the YOLOv4-driven appearance grading filing mechanism and apply it to more fruit quality classifications.

4. Conclusions

A high-precision cherry-tomato grading model can be derived based on the deep-learning framework and appearance grading archive. The model trained in this study exhibited a good recognition effect on a single category. This method is suitable for identifying cherry tomatoes and obtaining accurate identification and classification results. In this study, we did not have to consider how the degree of damage of tomato is finely graded. This is because, based on the expert-assessed appearance, severely damaged cherry tomatoes will be classified as discardable. Other categories will be classified according to the appearance they conform to. Adding cherry tomato feature images from different light sources and high background complexity to the dataset can reduce overfitting. Feature samples of cherry tomatoes with different light sources will be added in future studies. Several features with high background complexity and different categories were added for deep learning and training. Distinct classes can be compared and learned during the model training to reduce the possibility of overfitting. Accurate grading and filing mechanism helps to detect rotten fruit, realize instant grading and elimination, and reduce the number of rotten fruit in the whole batch. Furthermore, our study can serve as a pilot study for related research to improve future technological novelty. Our study proposes the idea of a hierarchical filing mechanism that could provide more technically novel studies in the future.

Author Contributions: Y.-H.C.: conceptualization, methodology, and project administration. C.-Y.T. and D.-M.N.: conceptualization, writing—review and editing. Y.-D.L.: methodology, writing, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Ministry of Science and Technology (MOST) in Taiwan in Taiwan (under Grant no. 110-2622-E-324-005, 111-2622-8-005-003-TE1, 111-2218-E-005-009, and 111-2821-C-324-001-ES).

Data Availability Statement: Not applicable.

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

References

1. Sogang, T.N.; Monkouop, Y. Past, present and future of urban agriculture in cameroon: Major contemporary challenges (1993–2017). *J. Agric. Chem. Environ.* **2022**, *11*, 1–14. [CrossRef]
2. Corrado, A.; Palumbo, L. Essential farmworkers and the pandemic crisis: Migrant labour conditions, and legal and political responses in italy and spain. In *Migration and Pandemics*; Springer: Cham, Switzerland, 2022; pp. 145–166.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
4. Yang, C.-H.; Moi, S.-H.; Ou-Yang, F.; Chuang, L.-Y.; Hou, M.-F.; Lin, Y.-D. Identifying risk stratification associated with a cancer for overall survival by deep learning-based coxph. *IEEE Access* **2019**, *7*, 67708–67717. [CrossRef]
5. Yang, C.-H.; Moi, S.-H.; Hou, M.-F.; Chuang, L.-Y.; Lin, Y.-D. Applications of deep learning and fuzzy systems to detect cancer mortality in next-generation genomic data. *IEEE Trans. Fuzzy Syst.* **2020**, *29*, 3833–3844. [CrossRef]
6. Su, X.; Xue, S.; Liu, F.; Wu, J.; Yang, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Jin, D. A comprehensive survey on community detection with deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–21. [CrossRef] [PubMed]
7. Song, H.; Montenegro-Marin, C.E. Secure prediction and assessment of sports injuries using deep learning based convolutional neural network. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 3399–3410. [CrossRef]

8. Umer, M.; Ashraf, I.; Ullah, S.; Mehmood, A.; Choi, G.S. Covinet: A convolutional neural network approach for predicting covid-19 from chest x-ray images. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 535–547. [CrossRef] [PubMed]
9. Shanthini, A.; Manogaran, G.; Vadivu, G.; Kottilingam, K.; Nithyakani, P.; Fancy, C. Threshold segmentation based multi-layer analysis for detecting diabetic retinopathy using convolution neural network. *J. Ambient Intell. Humaniz. Comput.* **2021**, 1–15. [CrossRef]
10. Moradzadeh, A.; Mohammadi-Ivatloo, B.; Abapour, M.; Anvari-Moghaddam, A.; Gholami Farkoush, S.; Rhee, S.-B. A practical solution based on convolutional neural network for non-intrusive load monitoring. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 9775–9789. [CrossRef]
11. Sethy, P.K.; Barpanda, N.K.; Rath, A.K.; Behera, S.K. Nitrogen deficiency prediction of rice crop based on convolutional neural network. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 5703–5711. [CrossRef]
12. De Luna, R.G.; Dadios, E.P.; Bandala, A.A. Automated image capturing system for deep learning-based tomato plant leaf disease detection and recognition. In Proceedings of the TENCON 2018–2018 IEEE Region 10 Conference, Jeju, Korea, 28–31 October 2018; pp. 1414–1419.
13. Elhassouny, A.; Smarandache, F. Smart mobile application to recognize tomato leaf diseases using convolutional neural networks. In Proceedings of the 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, 22–24 July 2019; pp. 1–4.
14. Juyal, P.; Sharma, S. Detecting the infectious area along with disease using deep learning in tomato plant leaves. In Proceedings of the 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 3–5 December 2020; pp. 328–332.
15. Choi, J.-W.; Tin, T.T.; Tu Le, H.T.; Park, G.-S.; Chien, V.D.; Kim, J.-W. A nutrient deficiency prediction method using deep learning on development of tomato fruits. In Proceedings of the 2018 International conference on fuzzy theory and its applications (iFUZZY), Daegu, Korea, 14–17 November 2018; pp. 338–341.
16. Jiang, D.; Li, F.; Yang, Y.; Yu, S. A tomato leaf diseases classification method based on deep learning. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 1446–1450.
17. Ashok, S.; Kishore, G.; Rajesh, V.; Suchitra, S.; Sophia, S.G.; Pavithra, B. Tomato leaf disease detection using deep learning techniques. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020; pp. 979–983.
18. Wang, Q.; Qi, F. Tomato diseases recognition based on faster rcnn. In Proceedings of the 2019 10th International Conference on Information Technology in Medicine and Education (ITME), Qingdao, China, 23–25 August 2019; pp. 772–776.
19. Zhao, J.; Qu, J. A detection method for tomato fruit common physiological diseases based on YOLOv2. In Proceedings of the 2019 10th International Conference on Information Technology in Medicine and Education (ITME), Qingdao, China, 23–25 August 2019; pp. 559–563.
20. Bhole, V.; Kumar, A. Mango quality grading using deep learning technique: Perspectives from agriculture and food industry. In Proceedings of the 21st Annual Conference on Information Technology Education, Virtually, 7–9 October 2020; pp. 180–186.
21. Kataoka, H.; Ijiri, T.; White, J.; Hirabayashi, A. Acoustic probing to estimate freshness of tomato. In Proceedings of the 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, Korea, 13–16 December 2016; pp. 1–5.
22. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
23. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, October 27–November 2 2019; pp. 6023–6032.
24. Zeng, G.; Yu, W.; Wang, R.; Lin, A. Research on mosaic image data enhancement for overlapping ship targets. *arXiv* **2021**, arXiv:2105.05090.
25. Minaee, S.; Liang, X.; Yan, S. Modern augmented reality: Applications, trends, and future directions. *arXiv* **2022**, arXiv:2202.09450.
26. Fu, Y. Fruit Freshness Grading Using Deep Learning. Master's Thesis, Auckland University of Technology, Auckland, New Zealand, 2020.
27. Naik, S.; Patel, B. Machine vision based fruit classification and grading-a review. *Int. J. Comput. Appl.* **2017**, *170*, 22–34. [CrossRef]
28. Jana, S.; Parekh, R. Intra-class recognition of fruits using color and texture features with neural classifiers. *Int. J. Comput. Appl.* **2016**, *148*, 1–6. [CrossRef]
29. Zhang, Y.; Phillips, P.; Wang, S.; Ji, G.; Yang, J.; Wu, J. Fruit classification by biogeography-based optimization and feedforward neural network. *Expert Syst.* **2016**, *33*, 239–253. [CrossRef]
30. Pérez, D.S.; Bromberg, F.; Diaz, C.A. Image classification for detection of winter grapevine buds in natural conditions using scale-invariant features transform, bag of features and support vector machines. *Comput. Electron. Agric.* **2017**, *135*, 81–95. [CrossRef]
31. Kusumiyati, K.; Hadiwijaya, Y.; Putri, I.E. Non-destructive classification of fruits based on vis-nir spectroscopy and principal component analysis. *J. Biodjati* **2019**, *4*, 89–95. [CrossRef]
32. Hanh, L.D. Autonomous lemon grading system by using machine learning and traditional image processing. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2022**, 1–8. [CrossRef]
33. Gai, R.; Chen, N.; Yuan, H. A detection algorithm for cherry fruits based on the improved yolo-v4 model. *Neural Comput. Appl.* **2021**, 1–12. [CrossRef]