computers

MDPI

Article

Optimizing Loss Functions for You Only Look Once Models: Improving Object Detection in Agricultural Datasets

Atsuki Matsui [1], Ryuto Ishibashi [1] and Lin Meng [2,*]

[1] Graduate School of Science and Engineering, Ritsumeikan University, 1-1-1, Nojihigashi, Kusatsu 525-8577, Shiga, Japan; ri0106hi@ed.ritsumei.ac.jp (A.M.); ri0097fx@ed.ritsumei.ac.jp (R.I.)

[2] College of Science and Engineering, Ritsumeikan University, 1-1-1, Nojihigashi, Kusatsu 525-8577, Shiga, Japan

[*] Correspondence: menglin@fc.ritsumei.ac.jp

Abstract: Japan faces a significant labor shortage due to an aging population, particularly in the agricultural sector. The rising average age of farmers and the declining participation of younger individuals threaten the sustainability of farming practices. These trends reduce the availability of agricultural labor and pose a risk to lowering Japan's food self-sufficiency rate. The reliance on food imports raises concerns regarding price fluctuations and sanitation standards. Moreover, the challenging working conditions in agriculture and a lack of technological innovation have hindered productivity and increased the burden on the existing workforce. To address these challenges, "smart agriculture" presents a promising solution. By leveraging advanced technologies such as sensors, drones, the Internet of Things (IoT), and automation, smart agriculture aims to optimize farm operations. Real-time data collection and AI-driven analysis play a crucial role in monitoring crop growth, assessing soil conditions, and improving overall efficiency. This study proposes enhancements to the YOLO (You Only Look Once) object detection model to develop an automated tomato harvesting system. This system uses a camera to detect tomatoes and assess their ripeness for harvest. Our objective is to streamline the harvesting process through AI technology. Our improved YOLO model integrates two novel loss functions to enhance detection accuracy. The first, "VSR", refines the model's ability to classify tomatoes and determine their harvest readiness. The second, "SBCE", enhances the detection of small tomatoes by training the model to recognize a range of object sizes within the dataset. These improvements have significantly increased the system's detection performance. Our experimental results demonstrate that the mean Average Precision (mAP) of YOLOv7-tiny improved from 61.81% to 70.21%. Additionally, the F1 score increased from 0.61 to 0.71 and the mean Intersection over Union (mIoU) rose from 65.03% to 66.44% on the tomato dataset. These findings underscore the potential of our proposed system to enhance efficiency in agricultural practices.

Keywords: object detection; YOLO; loss

check for updates

Academic Editors: Selene Tomassini and M. Ali Akber Dewan

Received: 13 December 2024
Revised: 20 January 2025
Accepted: 26 January 2025
Published: 30 January 2025

Citation: Matsui, A.; Ishibashi, R.; Meng, L. Optimizing Loss Functions for You Only Look Once Models: Improving Object Detection in Agricultural Datasets . *Computers* **2025**, *14*, 44. https://doi.org/10.3390/computers14020044

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recently, the labor shortage caused by Japan's aging population has become a critical issue. This challenge is particularly severe in the agricultural sector, where farmers struggle with a significant workforce decline. The sustainability of farming practices is under threat, primarily due to the aging population. According to data from the Ministry of Agriculture, Forestry and Fisheries [1], the number of farmers has decreased from 1.302 million in 2020 to 1.114 million in 2024. In addition, the average age of farmers has increased from 67.8 to 68.7 over the same period.

Furthermore, slow technological innovation has hindered productivity while increasing workloads, which has exacerbated the labor shortage in agriculture. These factors collectively contribute to the steady decline in the agricultural workforce. To address these challenges, "smart agriculture" presents a promising solution. Smart agriculture leverages advanced information technology and data analysis to optimize farming processes. This includes using sensor technology, drones, the Internet of Things (IoT), and automation. For instance, real-time data collection via sensors and AI-based analysis enables precise crop growth and monitoring of soil conditions. Additionally, using Unmanned Aerial Vehicles (UAVs) and automated tractors enhances workflow efficiency.

To further enhance agricultural productivity, AI is being implemented to automate various tasks, such as visual inspection and condition checking [2–4]. In this study, we aim to apply AI for agricultural products [5]. This paper focuses on developing an AI-based system for detecting tomato leaf diseases. This system enables farmers to automatically monitor crop conditions, reducing their workload and allowing for more efficient resource allocation.

We employ an AI model called "You Only Look Once" (YOLO) to automate the detection of tomato leaf diseases. YOLO is a widely recognized object detection model known for its high detection speed, which makes it well suited for industrial applications. Its speed enables the real-time inspection of large volumes of crops, while its lightweight design reduces initial costs and power consumption compared to other AI models. Specifically, we use YOLOv7, a model in the YOLO series that offers a balance of superior detection performance and compact size. Our proposed improvements enhance YOLOv7's effectiveness for agricultural applications. This paper introduces two key enhancements to YOLOv7. First, we propose an improvement to the object loss function ($Loss_{obj}$) that trains the model to better account for the distribution of object sizes in the dataset, placing more emphasis on each object. Second, we introduce a novel classification loss function to train the model's classification head, thereby increasing the reliability of its predictions. By optimizing the separation between class dimensions, this approach reduces classification errors.

These enhancements are integrated into YOLOv7 to improve the performance in detecting tomato leaf diseases, as shown in Figure 1. We evaluate the effectiveness of our proposed methods using the PASCAL VOC dataset, commonly used for benchmarking object detection models. The results highlight the quantitative benefits of our approach, underscoring its potential for improving agricultural efficiency.

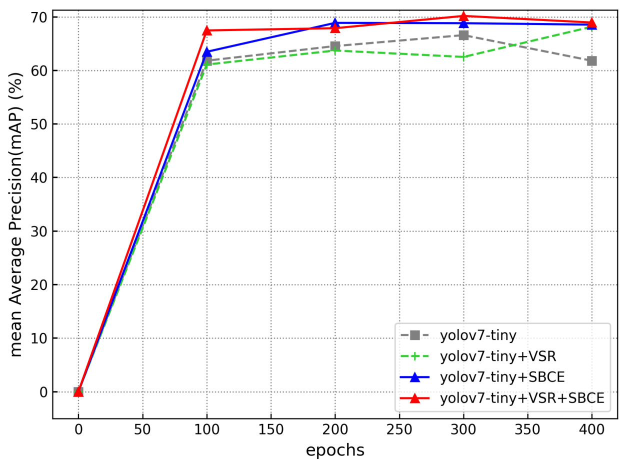


Figure 1. Comparison between our proposal and base model (YOLOv7-tiny).

In summary, the major contributions of this paper are as follows:

- Developing a new loss function to classify the condition of tomatoes more correctly.
- Developing a new loss function to detect small tomato leaf disease efficiently.

The remainder of this paper is structured as follows: Section 2 reviews related work on the current state of tomato leaf disease detection. Section 3 introduces our proposed loss function. Section 4 describes the dataset used in this study. Section 5 presents the results, and Section 6 discusses their implications. Finally, Section 7 concludes this paper and outlines directions for future research.

2. Related Work

This section explains the related works of this study.

2.1. Object Detection

Object detection is a key technology in image processing [6]. It identifies and classifies objects within images, and unlike image recognition, it can detect multiple types of objects in a single image. This versatility enables object detection models to be applied across a wide range of scenarios [7]. For example, automated driving systems rely on onboard cameras to detect pedestrians and vehicles, while manufacturing plants employ object detection to automatically inspect products and parts for external damage.

Object detection models are generally categorized based on their detection workflows. The first category is the two-stage detector, exemplified by models such as R-CNN [8] and Faster R-CNN [9]. These models achieve high detection accuracy but often sacrifice detection speed, making them less suitable for applications that require real-time performance. The second category is the one-stage detector, represented by models like YOLO and SSD [10]. These models prioritize high detection speed but typically offer slightly lower detection accuracy than two-stage detectors. Choosing the appropriate model depends on the specific use case and the dataset's characteristics. Despite its advantages, object detection has a significant drawback: the time-intensive process of creating training data. Each image in the training dataset must be manually annotated with detection targets, which requires substantial effort and time. This challenge makes generating large-scale datasets suitable for training effective object detection models difficult.

2.2. You Only Look Once

YOLO (You Only Look Once) [11] is one of the most popular object detection models, first introduced by Joseph Redmon in 2015. Compared to models such as R-CNN, EfficientDet [12], and DETR [13], YOLO stands out for its lightweight architecture and faster detection speed, making it well suited for real-time applications.

Over the past decade, researchers have continuously enhanced YOLO, resulting in a series of improved versions [14–16]. Various techniques have been employed to refine these models, including advanced feature extraction methods and modifications to training targets. These improvements aim to optimize the training process and further enhance YOLO's detection accuracy, speed, and overall efficiency.

2.2.1. Training

YOLO extracts image features through convolution operations applied to the training dataset. For example, Figure 2 shows the flow of YOLO's object detection on one of the person images of PASCAL VOC dataset. The backbone of YOLO is composed of convolutional blocks that are responsible for extracting these features. These features, known as feature maps, acquire a broader receptive field as the convolutional layers are stacked. However, smaller features may be lost as the receptive field increases.

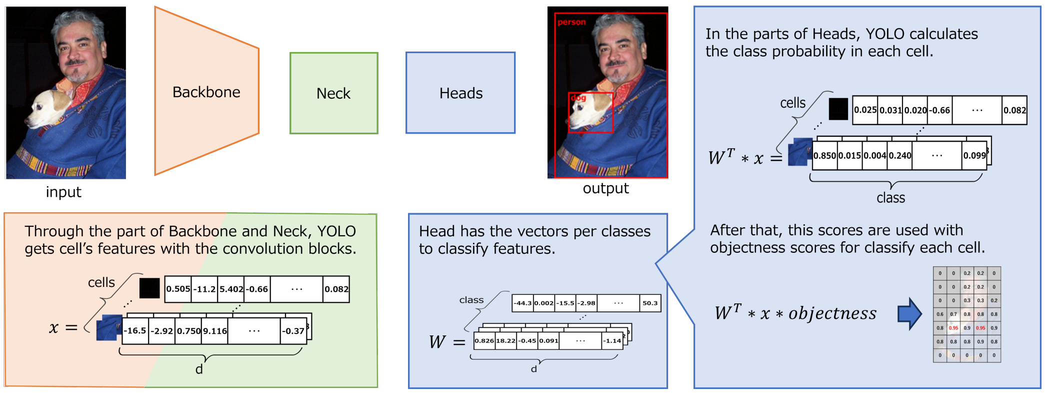

Backbone
Neck
Heads
input
output
person
dog
In the parts of Heads, YOLO calculates the class probability in each cell.
cells
0.025 0.031 0.020 -0.66 ⋯ 0.082
$W^T * x =$
0.850 0.015 0.004 0.240 ⋯ 0.099
class
Through the part of Backbone and Neck, YOLO gets cell's features with the convolution blocks.
cells
0.505 -11.2 5.402 -0.66 ⋯ 0.082
$x =$
-16.5 -2.92 0.750 9.116 ⋯ -0.37
d
Head has the vectors per classes to classify features.
class
-44.3 0.002 -15.5 -2.98 ⋯ 50.3
$W =$
0.826 18.22 -0.45 0.091 ⋯ -1.14
d
After that, this scores are used with objectness scores for classify each cell.
$W^T * x * objectness$

Figure 2. YOLO uses image feature vector (x) and the parameter (W). x is made from the images in the backbone and neck parts. W is the parameter of class representative vectors in the head part. YOLO compares the similarity between these vectors.

To address this issue, the neck component of YOLO is designed to share information among feature maps, combining them to generate feature maps that retain a variety of information. Based on these enriched feature maps, YOLO calculates loss scores and updates the model parameters iteratively. This constitutes the training procedure for YOLO. YOLO employs three key loss functions as the foundation for parameter updates.

L_{IoU} is a score that evaluates the model's ability to predict the object's shape. YOLO predicts the shape of an object for each cell in an image divided into $N \times N$ grids. YOLO assumes that each cell represents the center of an object and predicts the object's height (h) and width (w). To assess the accuracy of the predicted shape, YOLO uses a score called the Intersection over Union (IoU), as shown in Figure 3. IoU is the ratio of the intersection between the Ground Truth (GT) and the Prediction (Pr). IoU can be written as follows:

$$IoU = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Intersection_{ij}}{GT + Pr_{ij} - Intersection_{ij}}, \tag{1}$$

L_{IoU} can be written as follows:

$$L_{IoU} = 1 - IoU, \tag{2}$$

Considering various information such as the aspect ratio, many methods are proposed for IoU to predict more correctly [17–19].

L_{obj} is a score that evaluates the model's ability to predict the presence or absence of an object in the image. YOLO predicts the presence or absence of an object for each image cell divided into $N \times N$ grids. YOLO predicts whether an object is present in the image by determining whether a cell contains part of an object. L_{obj} is defined using Binary Cross Entropy (BCE) with the Ground Truth (GT) of each cell and the model's Prediction (Pr). L_{obj} can be written as follows:

$$L_{obj} = \sum_{i=1}^{n} \sum_{j=1}^{n} -(GT_{ij} * \log(Pr_{ij}) + (1 - GT_{ij}) * \log(1 - Pr_{ij})), \tag{3}$$

With various methods such as scaling, BCE is improved as a new $Loss_{obj}$ [20,21].

L_{cls} is a score that evaluates the model's ability to classify objects. YOLO predicts the object class for each image cell divided into $N \times N$ grids. YOLO receives the feature map from the neck, and the head creates a vector x containing d dimensions per cell. The head also has a vector W with d dimensions for each class. By calculating each cell's class information from these two vectors, the head predicts which objects are associated with each cell. L_{cls} is defined using Cross Entropy (CE) with the Ground Truth (GT) of each cell and the model's Prediction (Pr). L_{cls} is written as follows:

$$L_{cls} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{c} -(GT_{ij} * \log(P_{ijk}) + (1 - GT_{ij}) * \log(1 - P_{ijk})), \tag{4}$$

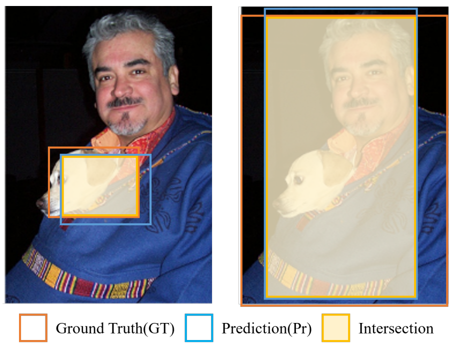


Figure 3. About IoU .

2.2.2. Test

YOLO uses the optimized parameters obtained during training to calculate the final results during testing. For object detection, YOLO combines two outputs, as shown in Figure 4. The first output relates to classification and objectness, which are calculated using the vectors x, the parameter W and the *objectness* score. YOLO computes this output for each image cell divided into $N \times N$ grids. The second output pertains to the shape of the object. By applying non-maximum suppression to combine the predictions, assuming each cell as the object's center, YOLO accurately predicts the object's shape. By calculating and integrating these outputs simultaneously, YOLO, as a one-stage detector, achieves faster detection speeds.

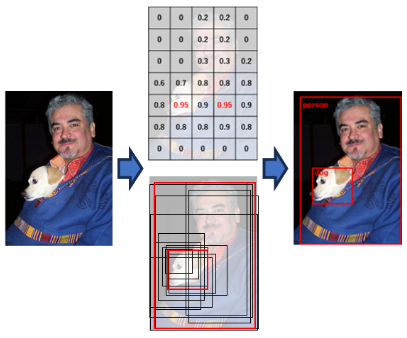


Figure 4. YOLO's output.

3. Method

In this paper, we propose two improvements to the loss function ($Loss$) to train YOLO-based models. By incorporating these new loss components into the training process and optimizing the training targets, our model achieves superior detection performance as an industrial AI compared to baseline models.

In the agricultural sector, where tasks are often large-scale and resource-intensive, smaller model sizes are essential for reducing installation costs and power consumption. However, small AI models face challenges such as limited expressive power and lower detection accuracy due to having fewer parameters. While larger models can enhance expressive power by training on extensive datasets, achieving high accuracy with smaller models requires optimal parameter tuning and advanced loss functions to enhance learning efficiency.

In addition, industrial data sets, particularly in agriculture, are often imbalanced and have limited images due to the challenges associated with data collection and preparation. Training with such datasets presents unique challenges compared to training with large-scale datasets. In this context, each back-propagation step becomes significantly more impactful, highlighting the importance of designing loss functions and training strategies to maximize the effectiveness of small, imbalanced datasets.

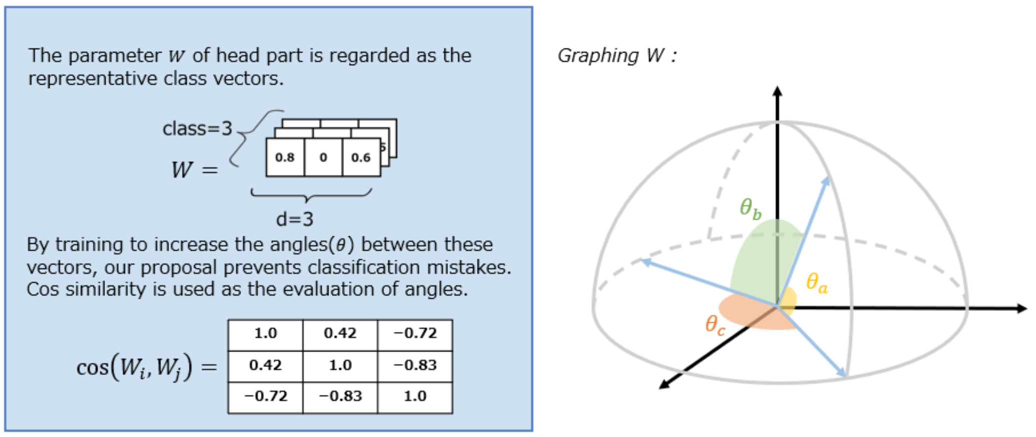
3.1. Vector Similarity Regularization

To solve this problem, the first one, referred to as "Vector Similarity Regularization (VSR)", incorporates head parameter into the loss function. As shown in Section 2, the head classifies objects in images using the vectors x and the parameter W. Classification is performed by comparing the similarity between these matrix. Therefore, the parameter W is regarded as the representative class vector. In practice, YOLO compute class probabilities for each $N \times N$ segmented cell of the image. The object's shape and classification are detected simultaneously by predicting which object each cell belongs to. When the values of the parameter W are too similar across classes, the model's classification ability is weakened. Regularization is applied to impose certain constraints and guide the learning process. Such studies have been proposed along with various constraints [22]. Our proposal trains YOLO to improve classification (Precision) by eliminating the similarity of the elements of the parameter W across classes. Cosine similarity, a widely adopted measure for evaluating vector distances, is used to evaluate the similarity between the vector x and the parameter W.

Cosine similarity can be written as follows:

$$\cos(W_i, W_j) = \frac{W_i \cdot W_j}{\|W_i\| \|W_j\|}, \tag{5}$$

An example of the calculation is shown in Figure 5. When the dataset contains three classes and the head has three dimensions per representative class vector, VSR trains the sum of θ_a, θ_b, and θ_c to increase. The cosine similarity $\cos(W_i, W_j)$ represents the similarity between each pair of dimensions. Since the diagonal elements of a matrix are always 1, the average of the off-diagonal elements is calculated.



The parameter W of head part is regarded as the representative class vectors.

By training to increase the angles(θ) between these vectors, our proposal prevents classification mistakes. Cos similarity is used as the evaluation of angles.

$\cos(W_i, W_j) =$

1.0	0.42	−0.72
0.42	1.0	−0.83
−0.72	−0.83	1.0

Figure 5. Vector Similarity Regularization (VSR): VSR regards the parameter W as representative class vectors. L_{VSR} trains W to separate representative features between each class.

Algorithm 1 illustrates the flow for calculating L_{VSR}. This proposal uses representative class vectors, with one vector being compared for similarity to the others. The similarity is calculated only for the corresponding dimensions. After computing all similarities, the average of these values is used as the final score.

Algorithm 1 Vector Similarity Regularization

```
Input: $W_{ij}$: Head vectors per classes
Output: $L_{VSR}$
1: $l$ = []
2: for i in c do
3:     for j in c do
4:         if $i \neq j$ then
5:             $l$.append(cos($W_i$,$W_j$))
6:         end if
7:     end for
8: end for
9: $L_{VSR}$ = average($l$) + 1
```

By training to minimize the cosine similarity between each vector toward -1, our proposal eliminates the similarity of elements between class vectors. Since the loss function requires a non-negative value, we add +1 to ensure that the minimum value is 0 and define L_{VSR} accordingly. L_{VSR} can be written as follows:

$$L_{VSR} = \text{average}(\cos(W_i, W_j)) + 1. \tag{6}$$

This proposal trains YOLO to learn a representative vector for each class, preventing multiple vectors from being similar to vector x. Only one class vector is similar to an object, which can improve classification accuracy. This proposal is classified as "Metric Learning". While various methods have been proposed in the field of "Image Recognition [23–25]", previous YOLO proposals do not adopt it. One reason for this is the training method used in YOLO. YOLO calculates the loss per $N \times N$ cell within the image. Therefore, the entire image feature cannot be treated at one time. This paper incorporates distance learning into YOLO using head vectors instead of image feature vectors. The weights are defined as "$loss_{obj}$:$loss_{cls}$:$loss_{iou}$(:$loss_{aux}$) = 10:1:2(:1)" for the PASCAL VOC dataset. This ratio remains in the open-source code. On the other hand, the $loss_{aux}$ ratio is defined by us. We set the weights such that the auxiliary losses are small compared to the other losses. When this loss is too large, the class information is broken, and detection will be difficult. To achieve this, we carefully choose the weight ratios.

3.2. Scaled Binary Cross Entropy

The second one, referred to as "Scaled Binary Cross Entropy (SBCE)", integrates object size into $Loss_{obj}$. Industrial datasets such as agricultural data often exhibit an imbalanced distribution of object sizes, as shown in Figure 6. This imbalance must be addressed when aiming to train AI models more efficiently. On the other hand, Binary Cross Entropy (BCE) is used to calculate the loss regardless of object size. BCE can be written as follows:

$$BCE = \sum_{i=1}^{n}\sum_{j=1}^{n} -(GT_{ij} * \log(Pr_{ij}) + (1 - GT_{ij}) * \log(1 - Pr_{ij})), \tag{7}$$

BCE is not suitable for training unbalanced and small data because it does not consider an uneven distribution of object sizes. Also, excessive back-propagation with small datasets should be avoided, as it can lead to overfitting. When a user implements AI into industries such as agriculture, users often need to create custom datasets. These original datasets, however, often suffer from imbalances in object size distribution. To address this issue, our proposed Scaled Binary Cross Entropy (SBCE) method incorporates object size into the loss function, emphasizing the importance of each object during training. Adapting the training process to fit the characteristics of the dataset can help to mitigate data imbalance.

As mentioned earlier, these original datasets often contain a large number of small objects. Treating large and small objects equally under these conditions can negatively affect the training process. To remedy this, SBCE scales $Loss_{obj}$ for small objects, prioritizing their detection. Larger loss values lead to greater parameter adjustments, making it easier for the model to detect small objects.

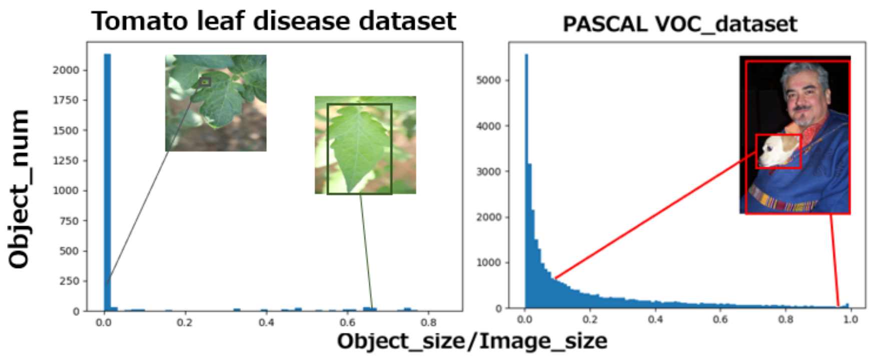


Figure 6. Examples of datasets with poor distributions for training.

SBCE can be written as follows:

$$SBCE = \sum_{i=1}^{n}\sum_{j=1}^{n} -(GT_{ij} * \log(Pr_{ij})^{r} + (1 - GT_{ij}) * \log(1 - Pr_{ij})^{r}), \quad (8)$$

To prioritize smaller objects during training, the SBCE scales the BCE using r. It becomes bigger than 1 with sizes below a predefined threshold to scale $Loss_{obj}$. The smaller the object size, the larger the value of r, and the larger the value of $Loss$. Since larger values of $Loss_{obj}$ lead to more significant parameter updates, r indicates the importance of that object in training. To determine this threshold, SBCE needs to define M before training as shown in Algorithm 2. M represents the maximum object size for which the $Loss_{obj}$ is scaled. When YOLO adopts SBCE, the object sizes in the Ground Truth (GT) dataset must be computed. The object size percentage in the images ($w \times h/W \times H$) is pre-calculated and used during training. For GT objects with sizes larger than M, r is set to 1, and the standard BCE is applied. In contrast, for GT objects smaller than M, r is set to a value between 1 and 2, and $Loss_{obj}$ is scaled accordingly.

Algorithm 2 Scaled Binary Cross Entropy

```
Input: : $M$:$max_{size}$, $GT$:Ground Truth, $Pr$:Prediction
Output: $L_{obj}$
1: for image in Dataset do
2:     get image width as $W$
3:     get image height as $H$
4:     $image_{size} = W * H$
5:     for object in image do
6:         get object width as $w$
7:         get object height as $h$
8:         $object_{size} = w * h / image_{size}$
9:         if $object_{size} < M$ then
10:            $r = 2 - object_{size}/M$
11:        else if $M \leq object_{size}$ then
12:            $r = 1$
13:        end if
14:        $L_{obj} = SBCE(GT, Pr, r)$
15:    end for
16: end for
```

The main advantage of the proposed method is the flexibility to adjust the scaling range according to the characteristics of the dataset. In this paper, we define an appropriate range for datasets that contain many small objects. When the dataset contains many large objects and the user aims to detect larger objects, the suitable range can be specified for the size of the object.

4. Dataset and Hyper-Parameter

In this study, we evaluate our proposed methods using two datasets. First, our model is trained on the PASCAL VOC dataset, which is widely used for quantitative evaluation. Additionally, we train our model on a tomato leaf disease dataset to demonstrate its application in industrial AI.

4.1. PASCAL VOC Dataset

Our proposed methods are evaluated using the PASCAL VOC dataset [26], which enables quantitative evaluation. The dataset contains 8069 training images and 997 test images, providing a reliable basis for assessing the proposed methods. Additionally, the dataset includes 20 object classes (e.g., airplane, person, dog, etc.) , as shown in Figure 7, further supporting the quantitative evaluation of the proposed approaches.

Figure 7. PASCAL VOC dataset.

4.2. Tomato Leaf Disease Dataset

This dataset consists of images of diseased tomato leaves [27]. These diseases are classified into six types (bacterial spot, black spot, early blight, late blight, leaf mold, and target spot). Leaves without disease symptoms are labeled as healthy , as shown in Figure 8. The model is trained using 645 images, with the results evaluated on 61 inference images and 31 test images. All image sizes are 640 × 640 pixels.

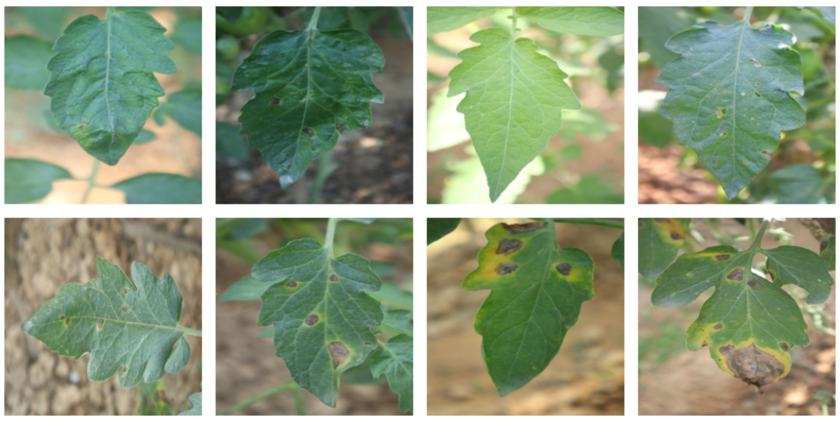


Figure 8. Tomato leaf disease dataset.

4.3. Hyper-Parameter

YOLO has various hyper-parameters for training, which are determined based on the size of the training data and the model. For the tomato leaf disease dataset, we set the batch size to 8, the number of epochs to 400, and the input shape to 640 $\times$ 640 pixels. For the PASCAL VOC dataset, the batch size is set to 64, the number of epochs to 500, and the input shape to 640 $\times$ 640 pixels. The optimizer used is Adam, and weight decay is adjusted from 1×10^{-3} to 1×10^{-5} using a cosine annealing function.

5. Results

In this paper, YOLOv7 is used as the baseline model due to its optimal dimensionality for representing vectors. When the dimensionality is too large, the VSR may easily produce a cosine similarity of -1. Conversely, when the dimensionality is too small, the VSR cannot achieve a cosine similarity of -1 without distorting the vector. To achieve these conditions, we balance YOLOv7.

As mentioned earlier, this paper trains and evaluates the proposed method on two datasets: the PASCAL VOC dataset and the tomato leaf dataset. Table 1 presents the evaluation results of the proposed method using the PASCAL VOC dataset.

Table 1. Results of PASCAL VOC dataset.

Model	mAP50 (%)	F1	mIoU (%)
YOLOv7-tiny	81.51	0.766	52.91
+VSR	**83.53**	0.802	**54.32**
+SBCE$_{maxobj}$	81.96	0.787	53.47
+VSR+SBCE$_{maxobj}$	83.53	0.806	54.24
YOLOv7	97.68	0.966	64.87
+VSR	97.71	0.967	64.77
+SBCE$_{maxobj}$	97.56	0.966	64.62
+VSR+SBCE$_{maxobj}$	97.57	0.965	64.83

Additionally, Tables 2 and 3 present the evaluation results of the proposed method using the tomato leaf disease dataset. Table 2 defines the size of the largest object in the dataset as M for SBCE. Table 3 also defines the largest object size as M for SBCE, focusing on the detection of smaller disease symptoms that are more critical to identify, as shown in Figure 9. Tables 4 and 5 present the evaluation results between proposed method and existing method on the tomato leaf disease dataset. Finaly, Table 6 present the ablation study about w_{VSR}, and Table 7 compares between our model and existing models on tomato leaf disease dataset.

Table 1 shows that the loss improvement of our proposed methods is effective for the PASCAL VOC dataset. Compared to the base model (YOLOv7-tiny and YOLOv7), VSR improved the mAP of YOLOv7-tiny from 81.51% to 83.53% and the mAP of YOLOv7 from 97.68% to 97.71%. Additionally, SBCE improved the map of YOLOv7-tiny from 81.51% to 81.96%. When both proposed methods are applied together, the mAP of YOLOv7-tiny improved from 81.51% to 83.53%.

Table 2 demonstrates that the loss improvement achieved by the proposed method is effective for the tomato leaf disease dataset. Compared to the base models (YOLOv7-tiny and YOLOv7), VSR improved the mAP of YOLOv7-tiny from 61.81% to 68.16%. Additionally, SBCE improved the mAP of YOLOv7-tiny from 61.81% to 68.55% and the mAP of YOLOv7 from 70.60% to 74.27%. When both proposed methods are applied together, the mAP of YOLOv7-tiny improved from 61.81% to 68.95%, and the mAP of YOLOv7 improved from 70.60% to 73.24%.

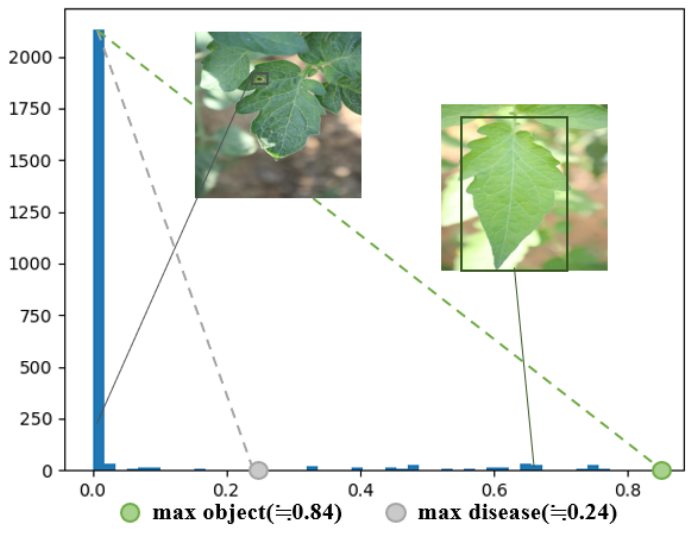


Figure 9. The distribution of tomato leaf disease dataset.

Table 2. Results of tomato leaf disease dataset, which defines max object size as *M*.

Model	mAP50 (%)	F1	mIoU (%)
YOLOv7-tiny	61.81	0.61	65.03
+VSR	68.16	0.71	63.83
+SBCE$_{maxobj}$	68.55	0.64	65.80
+VSR+SBCE$_{maxobj}$	68.95	0.70	65.30
YOLOv7	70.60	0.66	65.98
+VSR	69.35	0.70	48.83
+SBCE$_{maxobj}$	74.27	0.75	64.58
+VSR+SBCE$_{maxobj}$	73.24	0.73	65.69

Table 3 demonstrates that the loss improvement achieved by the proposed method is effective for the tomato leaf disease dataset. Compared to the base models (YOLOv7-tiny and YOLOv7), VSR improved the mAP of YOLOv7-tiny from 68.16% to 61.81%. Additionally, SBCE improved the mAP of YOLOv7-tiny from 63.16% to 61.81% and the mAP of YOLOv7 from 70.60% to 71.95%. When both proposed methods are applied together, the mAP of YOLOv7-tiny improved from 61.81% to 70.21%, and the mAP of YOLOv7 improved from 70.60% to 73.09%.

Table 3. Results of tomato leaf disease dataset, which defines max disease size as *M*.

Model	mAP50 (%)	F1	mIoU (%)
YOLOv7-tiny	61.81	0.61	65.03
+VSR	68.16	0.71	63.83
+SBCE$_{maxdis}$	63.16	0.58	65.67
+VSR+SBCE$_{maxdis}$	70.21	0.71	66.44
YOLOv7	70.60	0.66	65.98
+VSR	69.35	0.70	64.20
+SBCE$_{maxdis}$	71.95	0.74	65.47
+VSR+SBCE$_{maxdis}$	73.09	0.75	64.00

Figures 10 and 11 show the results of disease detection in the image. Our proposed methods enable the detection of diseases on leaves in the background of the images, which YOLOv7-tiny fails to detect.

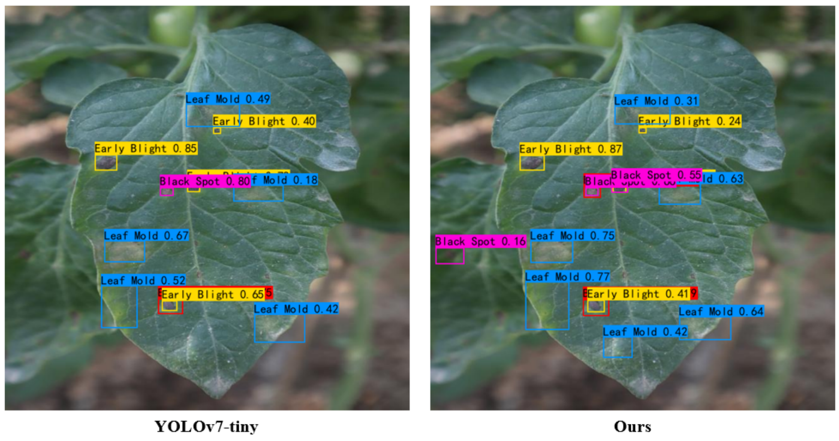


Figure 10. Comparison of image results on YOLOv7-tiny.

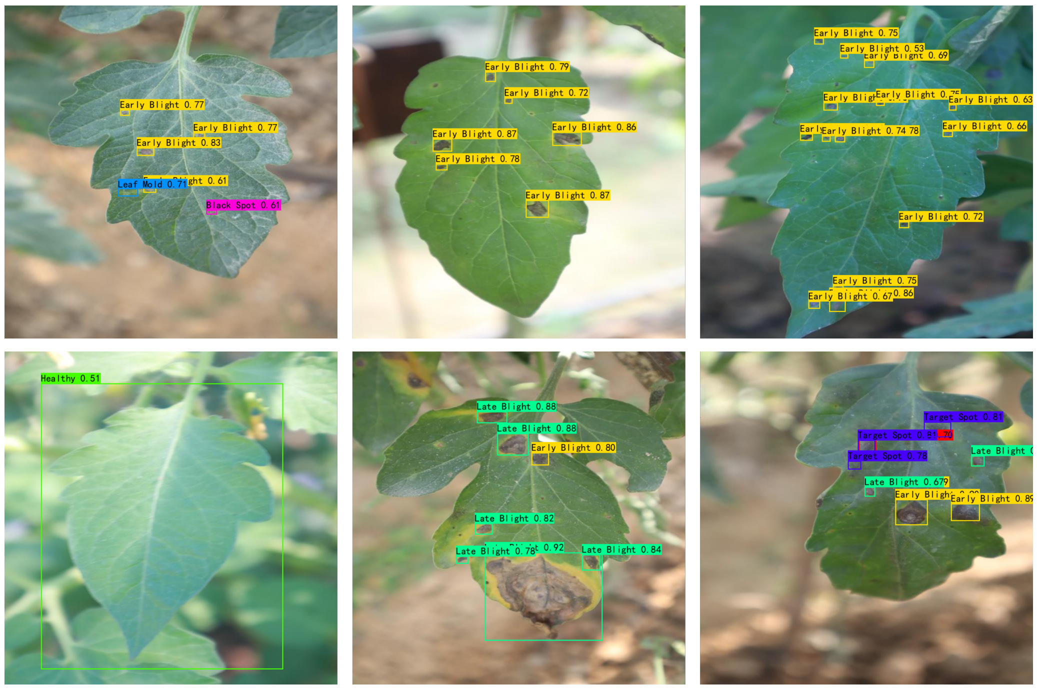


Figure 11. Detection results on YOLOv7 with our proposal.

As shown in Table 4, our proposed method achieves higher detection performance compared to existing methods. While focal loss reduces the mAP from 61.81% to 49.42%, our proposed method improves the mAP from 61.81% to 68.55%. Additionally, as shown in Table 5, the detection accuracy for leaf mold is particularly improved with both SBCE$_{maxdis}$ and SBCE$_{maxobj}$ on YOLOv7.

Table 4. Comparison results of tomato Leaf disease dataset with existing loss functions.

Model	mAP50 (%)	F1	mIoU (%)
YOLOv7-tiny	61.81	0.610	65.03
+Focal Loss	49.42	0.376	64.81
+SBCE$_{maxdis}$	63.16	0.580	65.67
+SBCE$_{maxobj}$	68.55	0.640	65.80

Table 5. Evaluation results of SBCE$_{maxobj}$ and SBCE$_{maxdis}$ on tomato leaf disease dataset.

Model	Class	AP50 (%)	F1	Recall (%)	Precision (%)
YOLOv7	Bacterial Spot	50.00	0.67	50.00	100.00
	Black Spot	59.68	0.50	35.71	83.33
	Early Blight	84.33	0.86	84.85	87.50
	Healthy	100.00	1.00	100.00	100.00
	Late Bright	80.21	0.73	60.00	92.31
	Leaf Mold	30.00	0.40	33.33	50.00
	Target Spot	90.00	0.89	100.00	80.00
+SBCE$_{maxobj}$	Bacterial Spot	50.00 (±0)	0.67 (±0)	50.00 (±0)	100.00 (±0)
	Black Spot	52.67 (−7.01)	0.55 (+0.05)	42.86 (+7.15)	75.00 (−8.33)
	Early Blight	79.36 (−4.97)	0.79 (−0.07)	81.82 (−3.03)	77.14 (−10.36)
	Healthy	100.00 (±0)	1.00 (± 0)	100.00 (±0)	100.00 (±0)
	Late Bright	77.89 (−2.32)	0.78 (+0.05)	70.00 (+10.00)	87.50 (−4.81)
	Leaf Mold	60.00 (+30.00)	0.73 (+0.33)	66.67 (+33.34)	80.00 (+30.00)
	Target Spot	100.00 (+10.00)	0.73 (−0.16)	100.00 (±0)	57.14 (−22.86)
+SBCE$_{maxdis}$	Bacterial Spot	50.00 (±0)	0.50 (−0.17)	50.00 (±0)	50.00 (−50.00)
	Black Spot	52.77 (−6.91)	0.57 (+0.07)	42.86 (+7.15)	85.71 (+2.38)
	Early Blight	82.99 (−1.34)	0.86 (±0)	81.82 (−3.03)	90.00 (+2.50)
	Healthy	100.00 (±0)	0.92 (−0.08)	85.71 (−14.29)	100.00 (±0)
	Late Bright	77.86 (−2.35)	0.78 (+0.05)	70.00 (+10.00)	87.50 (−4.81)
	Leaf Mold	56.67 (+26.67)	0.73 (+0.33)	66.67 (+33.34)	80.00 (+30.00)
	Target Spot	83.33 (−6.67)	0.80 (−0.09)	100.00 (±0)	66.67 (−13.33)

Table 6 presents the ablation results for the L_{aux} weight. With a weight of 0.2, our VSR achieves an mAP from 81.51% to 83.53% on the PASCAL VOC dataset. Other weight values also achieve higher mAP, such as 82.42% and 82.20%. In addition, all weights show a better mAP with SBCE on yolov7-tiny.

Table 6. Ablation study of L_{aux} weight on PASCAL VOC dataset.

Model	mAP50 (%)	F1	mIoU (%)
YOLOv7-tiny	81.51	0.766	52.91
+VSR(w = 0.1)	82.42	0.786	54.29
+VSR(w = 0.1)+SBCE	82.71	0.804	54.05
+VSR(w = 0.2)	83.53	0.802	54.32
+VSR(w = 0.2)+SBCE	83.53	0.806	54.24
+VSR(w = 0.3)	82.20	0.770	52.78
+VSR(w = 0.3)+SBCE	82.60	0.794	53.47

Table 7 presents the results of our proposed methods and existing YOLO series models. When compared to other YOLO models with similar architectures, our proposed methods achieve higher mAP and $F1$ scores.

Table 7. Comparison results of our proposed methods with existing models on PASCAL VOC dataset.

Model	mAP50 (%)	F1
YOLOX-tiny	64.57	0.548
YOLOX-nano	79.47	0.747
YOLOv7-tiny	81.51	0.766
YOLOv8-nano	79.73	0.768
YOLOv8-s	82.70	0.813
Ours (VSR+SBCE)	83.53	0.802

6. Discussion

Our proposed methods demonstrate improved performance on the PASCAL VOC and tomato leaf disease datasets. As shown in Table 1, YOLOv7-tiny shows better performance on the PASCAL VOC dataset with our proposed enhancements. However, applying SBCE slightly reduces the performance of YOLOv7. This result suggests that, with sufficiently large datasets and model sizes, YOLOv7 can achieve satisfactory performance without additional training adjustments, such as head vector optimization using VSR. Furthermore, the 20-class configuration of the PASCAL VOC dataset increases the complexity of VSR training. For SBCE, the wide variation in object sizes within a single class, combined with a large number of training samples, makes weighted back-propagation less effective. These factors explain the limited improvements observed with our methods on the PASCAL VOC dataset.

Conversely, as shown in Table 2, our proposed methods are highly effective when applied to smaller datasets and models. Smaller models, such as YOLOv7-tiny, naturally struggle with expressive power due to their limited number of parameters. However, VSR facilitates the creation of more optimal parameters, compensating for this limitation. Additionally, SBCE effectively weights back-propagation, which is especially beneficial for datasets with limited training samples, where each back-propagation step carries greater significance.

For these reasons, YOLOv7-tiny, with its smaller model size, achieved a significant improvement in mean Average Precision (mAP) of 7.14%. Similarly, YOLOv7 also showed an improvement in mAP of 2.64%. These results emphasize the efficacy of our proposed methods, especially in scenarios involving small models and limited datasets.

Table 3 presents the results of adjusting the scaling range of SBCE from the maximum object size in the dataset to the maximum disease size. Reducing the scaling range makes the weighted values relatively more significant, which helps to clarify the training target and leads to more efficient training. Compared to the results in Table 2, this adjustment improved the mean Average Precision (mAP) for YOLOv7 from 68.95% to 70.21%.

This improvement is likely influenced by inadequate annotations, as illustrated in Figure 12. For example, detecting multiple diseases in a single annotated image, as shown in the “Train Image” example, can lead to false positives, which negatively impact the training process.

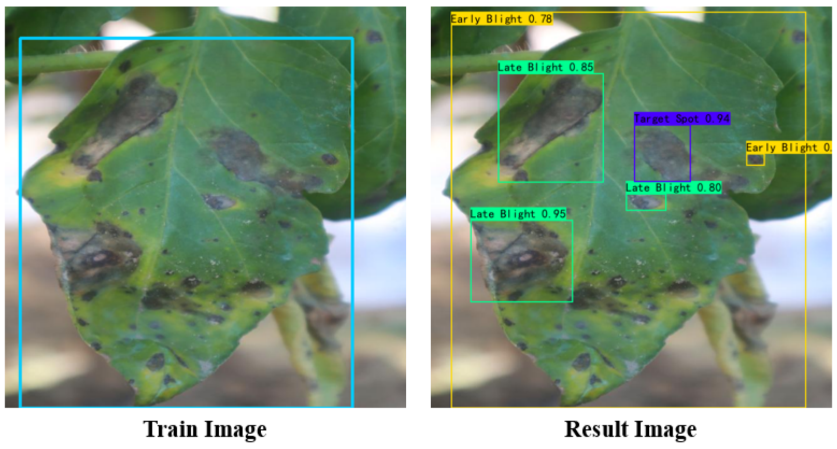


Figure 12. An example of poor annotation in tomato leaf disease dataset.

As shown in Table 4, our proposed methods achieve higher detection results compared to existing methods. Focal loss is a method that considers confidence for training. However, focal loss may not be suitable for training datasets that contain multiple similar classes, such as different disease types.

Additionally, Table 5 compares the results across different classes. SBCE enhances L_{obj}, allowing the model to detect more objects. However, the smaller scaling range and the lack of scaling for the healthy (leaf) loss contribute to inadequate training and lower accuracy in some cases. These challenges highlight the challenges of improving detection performance with limited and unbalanced datasets.

Table 6 presents the ablation results of L_{aux} weight. The results highlight the importance of selecting an appropriate value. Values that are too small reduce the effectiveness of VSR, while values that are too high increase the difficulty of training VSR. Therefore, it is crucial to define an optimal value for the L_{aux} weight. To prevent parameter breakdowns as shown in Figure 13 during YOLO training, this weight should be set to its optimal value.

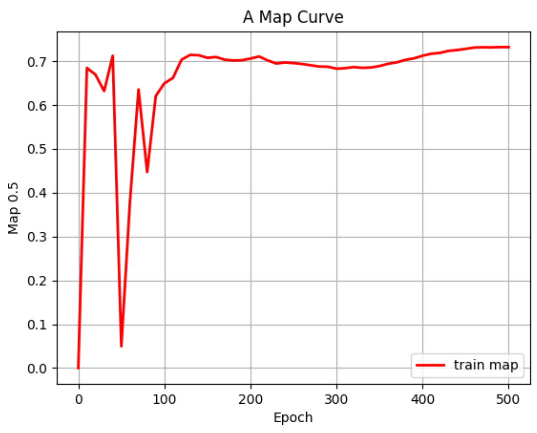


Figure 13. When YOLO trains with high L_{aux} weight like 0.50, head parameter become unstable.

Based on these findings, we conclude that our proposed method is most effective when applied to YOLOv7-tiny, as shown in Table 7. This model benefits the most from the optimized training process, leading to significant improvements in both detection efficiency and accuracy.

7. Conclusions

This paper proposes two loss improvement methods to enhance detection performance on industrial datasets. Industrial datasets are often imbalanced and contain a limited number of images due to challenges in data preparation. Training on such datasets requires careful optimization, as each back-propagation step is more significant than training on large-scale datasets. Additionally, small AI models, favored for agricultural applications due to their lower cost and energy requirements, have fewer parameters compared to large-scale AI models. This limitation requires more precise parameter tuning to achieve satisfactory performance. To address these challenges, we propose the following improvements: “VSR” optimizes class classification by separating the head vector values for each class, thereby reducing false positives and improving prediction accuracy; “SBCE” incorporates object size into the training process, ensuring that the training is appropriately tailored to the specific characteristics of the dataset. This approach enhances the model’s ability to detect objects of varying sizes in imbalanced datasets.

By integrating these improvements, our proposed method enhances the detection accuracy of compact models on imbalanced industrial datasets, making it particularly suitable for applications in agriculture and other industries where data constraints are common. In future work, we plan to further validate the effectiveness of our methods through quantitative comparisons across various model architectures and datasets. This will provide deeper insights into the broader applicability of our approach.

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

Funding: This research received no external funding.

Data Availability Statement: The PASCAL VOC dataset is widely used for object detection tasks, which is created by Visual Object Classes Challenge (VOC Challenge) concludes Everingham, L. et al. The dataset contains 8069 training images and 997 test images. It is available at [26]. The Tomato Leaf Disease dataset is created by Sylhet Agricultural University. The dataset contains 645 training images, 61 inference images and 31 test images. It is available at [27].

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

References

1. Research of the Ministry of Agriculture, Forestry, and Fisheries. Statistics on Agricultural Labor Force. Available online: https://www.maff.go.jp/j/tokei/sihyo/data/08.html (accessed on 20 January 2025).
2. Li, Q.; Wang, M.; Gu, W. Computer vision based system for apple surface defect detection. *Comput. Electron. Agric.* **2002**, *36*, 215–223.
3. Wang, Q.; Qi, F.; Sun, M.; Qu, J.; Xue, J. Identification of tomato disease types and detection of infected areas based on deep convolutional neural networks and object detection techniques. *Comput. Intell. Neurosci.* **2019**, *2019*, 9142753.
4. Tian, Y.; Yang, G.; Wang, Z.; Li, E.; Liang, Z. Detection of apple lesions in orchards based on deep learning methods of cyclegan and YOLOv3-dense. *J. Sens.* **2019**, *2019*, 7630926.
5. Matsui, A.; Meng, L.; Hattori, K. Enhanced YOLO using Attention for Apple grading. In Proceedings of the 2023 International Conference on Advanced Mechatronic Systems (ICAMechS), Melbourne, Australia, 4–7 September 2023; pp. 1–5.
6. Wu, X.; Sahoo, D. Recent advances in deep learning for object detection. *Neurocomputing* **2020** , *396*, 39–64.
7. Ishibashi, R.; Kaneko, H.; Meng, L. Enhancing DETR with Attention-Based Thresholding for Efficient Early Japanese Book Reorganization. In Proceedings of the 2023 International Conference on Advanced Mechatronic Systems (ICAMechS), Melbourne, Australia, 4–7 September 2023; pp. 1–7.
8. 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.
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149.
10. 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 Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
11. Redmon, J. 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.
12. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
13. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
14. 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.
15. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
16. Wen, H.; Dai, F.; Yuan, Y. A Study of YOLO Algorithm for Target Detection. In Proceedings of the 2021 International Conference on Artificial Life and Robotics (ICAROB2021), Online, 21–24 January 2021; pp. 287–290.

17. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
18. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.
19. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q.; Zuo, W. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Trans. Cybern.* **2021**, *52*, 8574–8586.
20. Lin, T. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002.
21. Zhang, H.; Wang, Y.; Dayoub, F.; Sunderhauf, N. Varifocalnet: An iou-aware dense object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8514–8523.
22. Kaneko, H.; Ishibashi, R.; Meng, L. Deteriorated characters restoration for early Japanese books using enhanced cyclegan. *Heritage* **2023**, *6*, 4345–4361.
23. Hoffer, E.; Ailon, N. Deep metric learning using triplet network. In Proceedings of the Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, 12–14 October 2015; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2015; pp. 84–92.
24. Qi, C.; Su, F. Contrastive-center loss for deep neural networks. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 2851–2855.
25. Deng, J.; Guo, J.; Xue, N.; Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4690–4699.
26. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338.
27. Sylhet Agricultural University Tomato Leaf Diseases Detect Dataset. 2024. Sylhet Agricultural University, Tomato Leaf DIseases Detect Computer Vision Project. Available online: https://universe.roboflow.com/sylhet-agricultural-university/tomato-leaf-diseases-detect (accessed on 20 January 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.