A Small Object Detection Framework for Unmanned Aerial Vehicles Images

AI CUP 2022 Fall Competition TEAM 2610

Jing-En Huang, and Jia-Wei Liao

January 26, 2023

Abstract

In this competition, we construct a detection model utilizing deep learning methods. In deep learning, object detection models can be classified into one-stage and two-stage methods. Recently, YOLO-based models have been considered state-of-the-art in the realm of one-stage methods that exhibit high speeds in the inference process. Conversely, RCNN-based models are well-known two-stage methods that demonstrate high accuracy in inference. At the begining, we utilize YOLOv5 to construct the baseline model which results in a public score of 0.708642. Then, we implement augmentation techniques and super-resolution as pre-processing methods. Moreover, we proposed a novel small object augmentation (SOA) algorithm to generate more data. This innovative method allows us to enrich our dataset with a large number of additional samples, which can be used to improve the performance of our model. After conducting experiments, we obtain a public score of 0.736959. Finally, we develop a post-processing algorithm that increases the public score to 0.739363 with the best private score being 0.754987, which is a significant improvement. Our source code is available at https://github.com/AngusBB/Competition, and the model weights can be downloaded from our Google Drive folder at https://drive.google.com/drive/folders/1sCA3ife1VMrB3eI59pjFryXG8k4LKElY?usp=sharing.

Keywords: Object Detection, Super-Resolution, Sliding Window, Data Augmentation, Small Object Augmentation, Weighted Box Fusion, Kmeans.

Contents

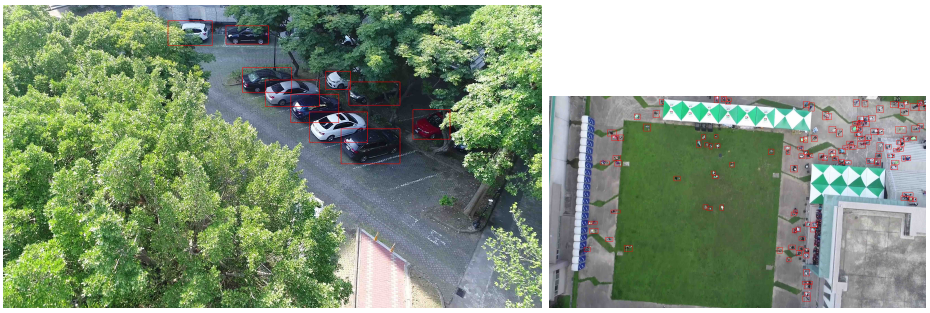
List of Figures

List of Tables

Chapter 1

Introduction

The competition dataset consists of 1,000 images that are divided into four categories: car, hov, person, and motorcycle. The images have resolutions of either 1920 × 1080 or 1344 × 720, as shown in Figure 1.1. In order to train the best model, we use the cross-validation technique to split the dataset into 900 images for training and 100 images for validation. The public and private datasets for inference each contain 500 images.

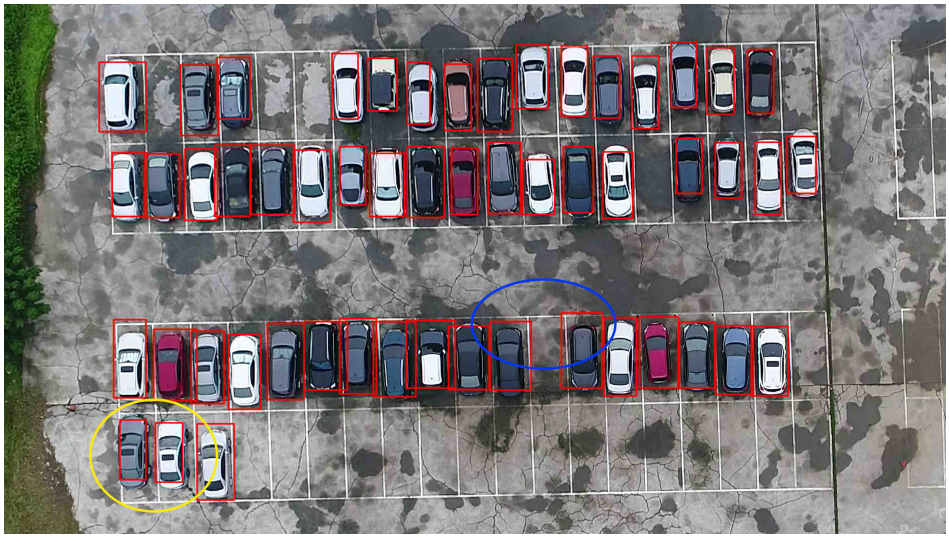


(a) img0375 (b) img0875

Figure 1.1: Examples of two different size training images.

The training dataset has some defects, as shown in Figure 1.2. First, some labels are not accurately marked which would lead to confusion and difficulty in training the model. For example, many objects are labeled with the wrong class, preventing the model from learning the correct associations between objects and their labels. Second, the bounding boxes of some objects are only partially marked, which would lead to difficulty for the model to identify the objects completely. Third, objects far from the camera are not labeled, which would also lead to difficulty for the model to learn effectively. To remedy these defects, we have relabeled the data to ensure that the labels are accurate and complete. This helps improve the model's performance and enables it to learn more effectively.

(a) Yellow van is labeled into hov (img0003) and car (img0029) in different images.

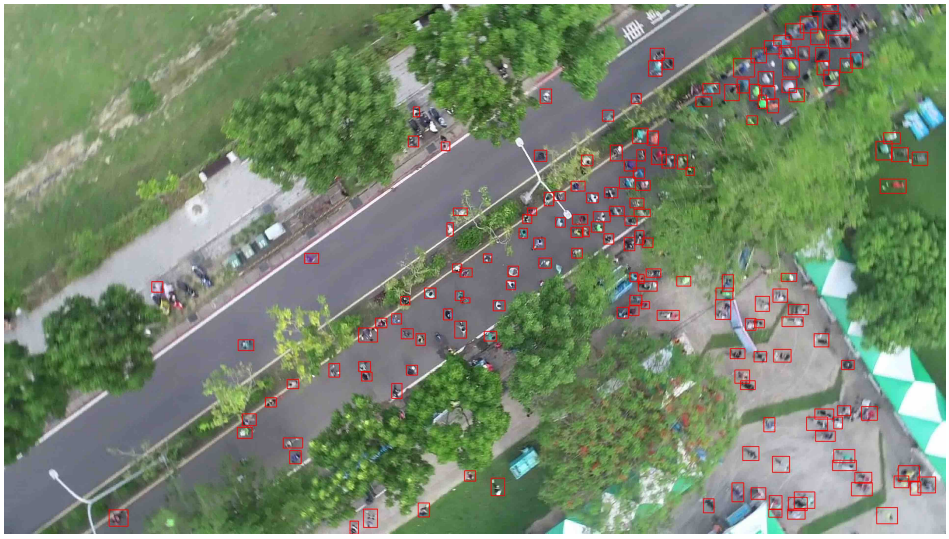


(b) Many bounding boxes of cars are either insufficient (yellow) or exceed (blue) the object seriously.

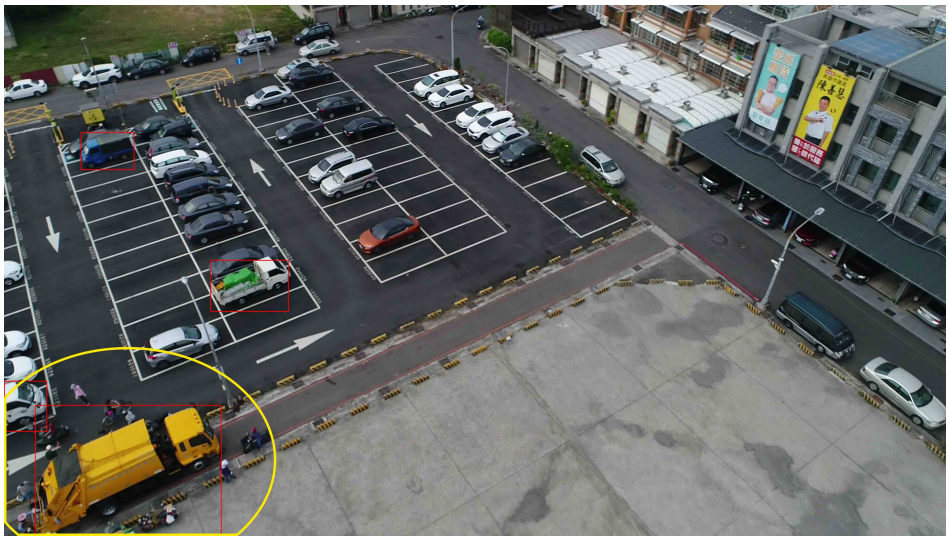
(c) The extremely small and distant objects are not labeled.

Figure 1.2: Example of problems in training set.

This competition presents two main difficulties, as shown in Figure 1.3. One is that some objects are too small and blurry to easily recognize by the model. Another is that images taken at an angle can cause the bounding boxes to be distorted and elongated. To address these challenges, we use a super-resolution method to enlarge the images, which improves their quality and makes it easier for the model to recognize the objects. Additionally, we have designed an algorithm that automatically generates augmentation data, which can help the model improve its recognition of small objects. These approaches help us overcome the difficulties of this competition and improve the performance of our model.



(a) Objects in image are too small and blurry.



(b) The bounding boxes in image are stretched very long.

Figure 1.3: Examples of difficulties to image recognition in training set.

Chapter 2

Related Work

Object detection is a key technology in computer vision with various applications, such as industrial production management, autonomous driving, and security surveillance. The main goal of object detection is to identify and localize objects in an image or video. It can be performed using either one-stage or two-stage models.

One-stage models, such as YOLO (You Only Look Once), are advantageous for their fast prediction speed and are widely used in applications as real-time performance is the main concern. However, they tend to have lower accuracy compared to two-stage models. On the other hand, two-stage models, such as R-CNN (Regional Convolutional Neural Network) [1] and its variants: Fast R-CNN [2], Faster R-CNN [3], and RetinaNet [4], are advantageous for their high accuracy and are widely used in applications as precision is the main concern. These models typically involve two stages: a region proposal stage, which generates a set of candidate object regions, and a classification and regression stage, which refines the object locations and predicts the class labels. We show the architecture of the one-stage and two-stage models in Figure 2.1.

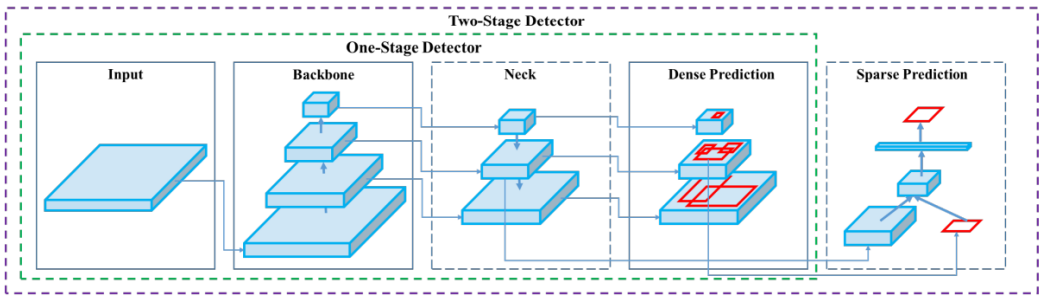


Figure 2.1: Architecture of detection model.

In recent years, object detection has become a popular research topic in Taiwan, with many teams from both industry and academia working on the development and improvement of models. In 2020, the team at Academia Sinica proposes YOLOv4 [5], which

improves the accuracy of the one-stage model, and subsequently proposes YOLOR [6] and YOLOv7 [7] to further improve efficiency and accuracy. We show the comparison of the accuracy of the YOLO-based models in Figure 2.2.

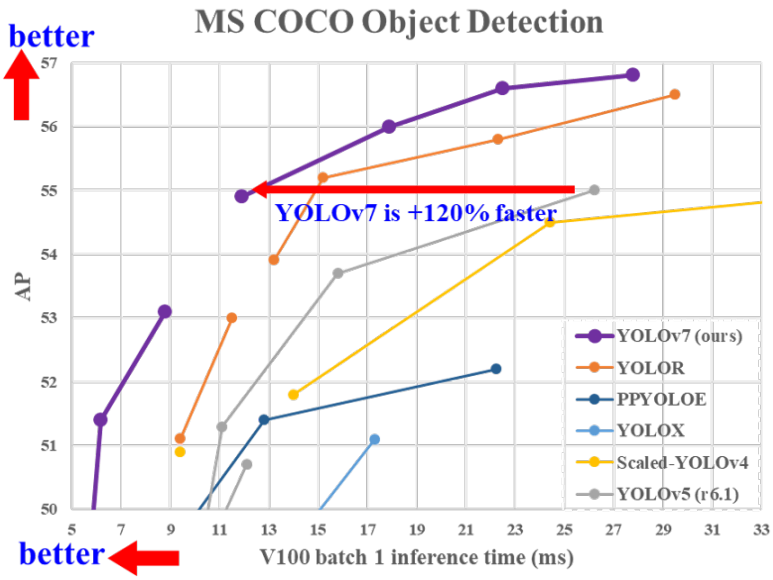


Figure 2.2: Comparison for YOLO-based models.

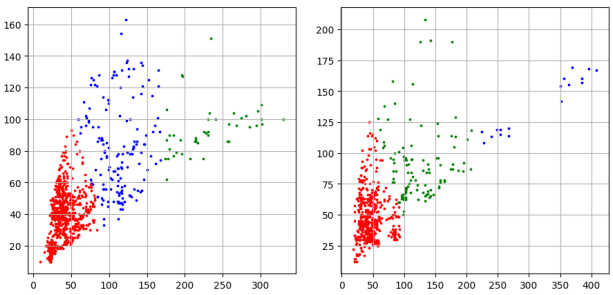
After a series of experiments, we have found that YOLOv5 [8] is the most stable and accurate model among the YOLO variants. Based on this fact, we adopt it as the main framework for our object detection tasks. We believe that YOLOv5 has the potential to greatly improve the performance of various applications in Taiwan's industrial production management.

Chapter 3

Methodology

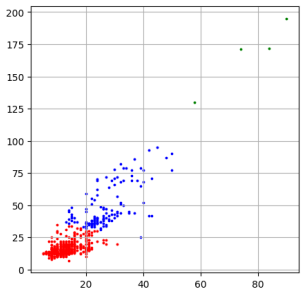
3.1 Data Analysis

We plot the length and width of the bounding box as a scatter plot, shown in Figure [**?**]. As we can see, the distribution of hov's points is scattered. This may be related to the height and angle at which the image is captured. If the shooting angle is more inclined, the bounding box may be stretched longer. In addition, we use the K-means algorithm to classify the further scatter plots of the length and width of the bounding boxes of the four categories, hoping to divide the categories into finer ones for training. However, it was later found that the size of the bounding box of the same category in the same image may also have a huge difference, so we gave up the idea of training the model separately according to the size of the bounding box.

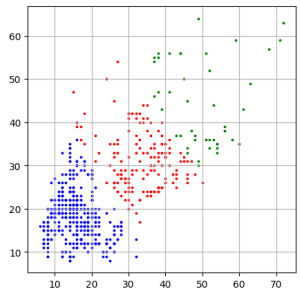


(a) Car.

(b) Hov.



(c) Person.



(d) Motorcycle.

Figure 3.1: Kmeans result for the bounding boxes of the height and width (pixels).

In Figure 3.2, we show the area distribution of bounding boxes. The horizontal coordinate represents the area of the bounding box, and the vertical coordinate represents the quantity. As we can see in the bar chart, there are two images with a bounding box area of 0, indicating errors in the original dataset. The majority of bounding box areas are concentrated between 1,225 and 2,340. The blue bin represents the extreme values in the distribution of bounding box areas, which may be due to the inclination of the image during capture. This analysis allows us to understand better the distribution of bounding box areas in our dataset and identify potential issues such as incorrect annotations or distorted images.

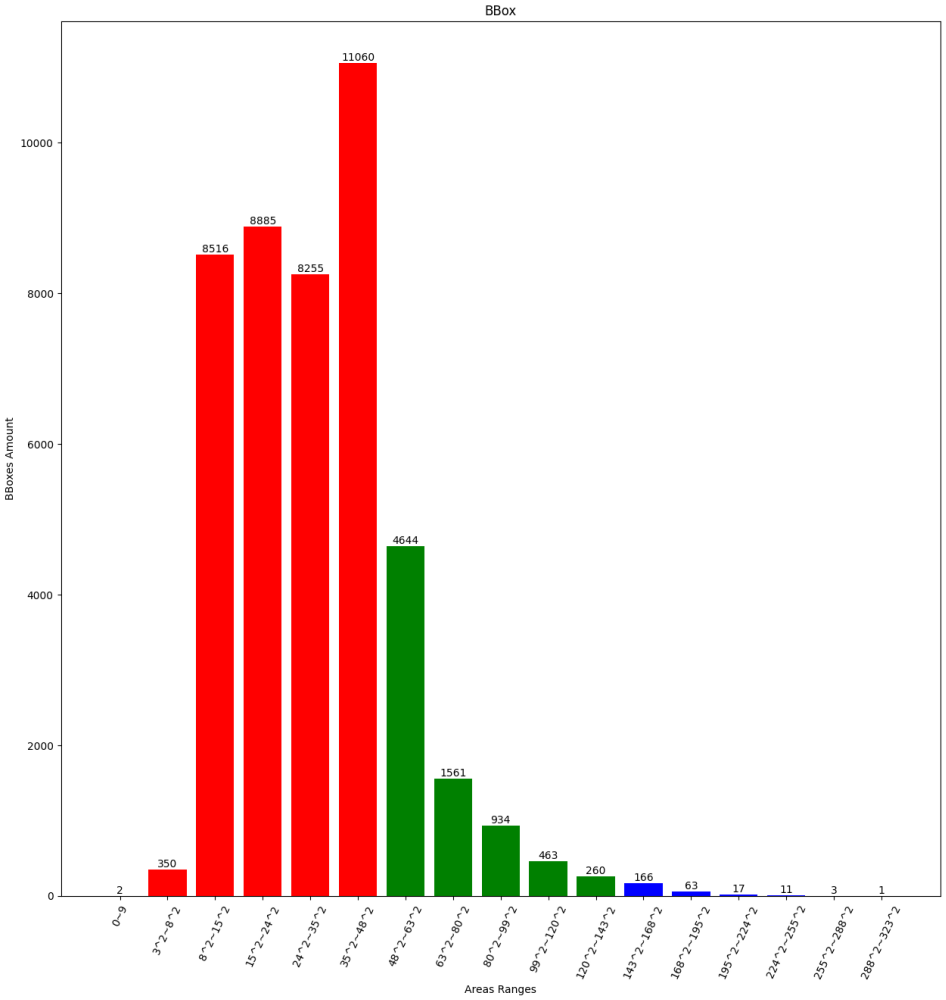


Figure 3.2: Area distribution of bounding box (pixels).

3.2 Data Pre-processing

Before the training process, we first normalized the images using z-score transformation. This normalization technique helps to control the gray scale distribution of the images, resulting in values that are centered around -5 to 5. This can speed up the convergence of the model's parameters and improve the stability of its predictions.

3.2.1 Super-Resolution

To increase the recognition rate of our model, we implemented a super-resolution method. The general method for image resizing is the bilinear method, but to improve the clarity of small objects in the image, we experimented with bicubic and deep learning-based super-resolution models for comparison. Since most deep learning-based super-resolution methods are supervised learning, we sought out additional dataset [9] to train the super-resolution model. We tested the SRCNN [10] and EDSR [11] models and found that the predicted results were similar to those obtained using bicubic. As a result, we ultimately adopted the simpler bicubic method as a means of improving the resolution.

3.2.2 Sliding Window

The sliding windows method can reduce the computational complexity of the model and therefore reduce the cost of training. However, when we try to use sliding windows for training, we find that during the detection phase, large targets are easily split into two small bounding boxes. Our solution is to use K-means, as described in Section , to find images with large bounding boxes and resize them to a fixed size, then train them together with other sliding window images. This approach avoids the issue of large targets being split into multiple bounding boxes.

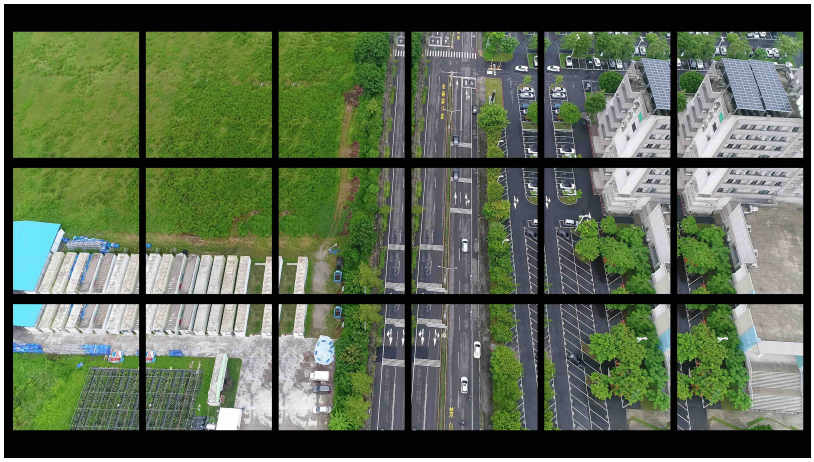


Figure 3.3: The result of image after applying sliding window.

3.2.3 Data Augmentation

To prevent overfitting, we employed data augmentation techniques and divided the data into four parts. This helped to ensure that our model was able to generalize well to unseen data and improve its overall performance.

Standard Transformation

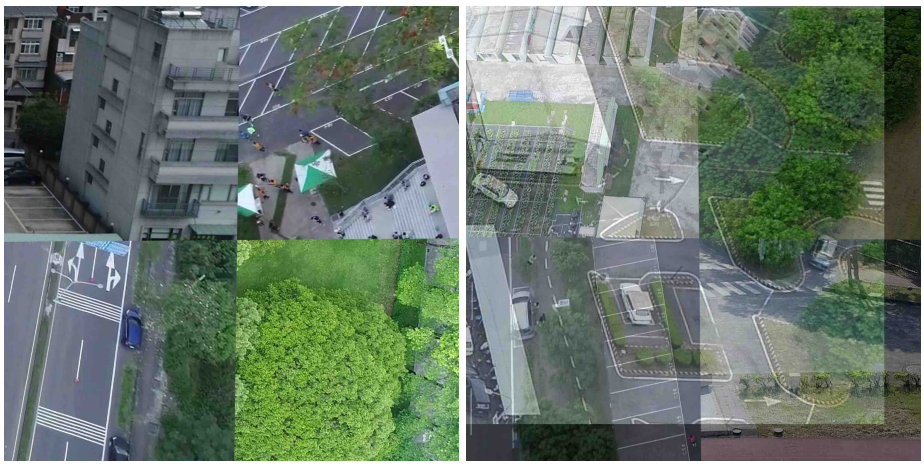
We use HSV transformation, random flipping, random rotation, random translation, random scaling, and random shearing as the basic transformation. These techniques are built into the YOLOv5 algorithm and proved to be effective at preventing overfitting and improving the model's ability to generalize to new data.

Mosaic

Mosaic combines multiple training images into a single composite image in a specific ratio. It can help the model to learn how to identify objects at smaller scales than it normally would, which can be useful in improving its performance on tasks such as object detection. Moreover, mosaic is particularly useful in training, as it significantly reduces the need for a large mini-batch size, which can speed up the training process and improve the convergence of the model.

Mixup

Mixup [12] is a data augmentation technique. It takes two samples from the training data to create a new mixed sample by a convex combination of their image's grayscale. It can help the model generalize better to new data.



(a) Mosaic. (b) Mixup.

Figure 3.4: Results for the image after mosaic and mixup.

Small Object Augmentation (SOA)

To address the difficulties problem discussed in Chapter 1, we have designed a novel algorithm for generating small object data augmentation, which has been instrumental in increasing the density of small objects in images. The algorithm is outlined below:

1. Random apply Mosaic, Affine transformation, Mixup and Flipping to the image.
2. Search for all bounding boxes with an area smaller than the specified threshold.
3. Randomly selected the objects that we wanted to copy and paste into the images.
4. Copy and paste the selected small objects three times in random places and avoid overlapping with other existing objects.

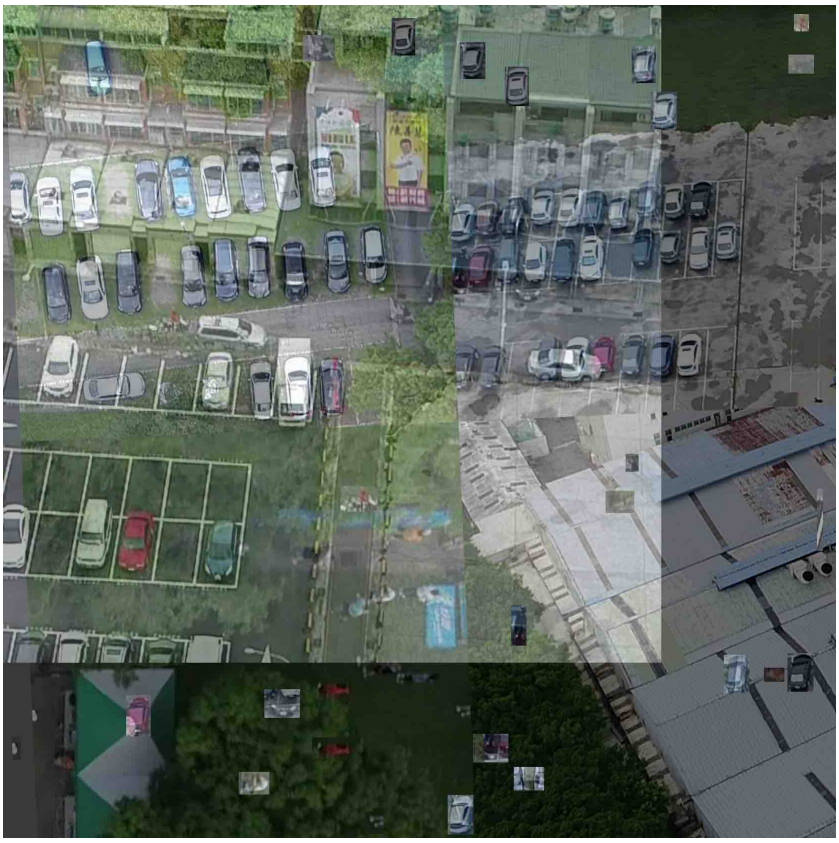


Figure 3.5: An illustration for small objects augmentation.

3.3 Model Architecture

The architecture of YOLOv5 model, as illustrated in Table 3.1, which consists of three main parts: the backbone, neck, and head. The backbone is responsible for extracting features from the input image, the neck processes the extracted features and generates a set of bounding boxes and class probabilities, and the head makes the final prediction based on the output of the neck.

Backbone	CSPNet
Neck	FPN
Head	Dense Prediction

Table 3.1: Architecture of YOLOv5

3.3.1 Cross Stage Partial Network (CSPNet)

Cross Stage Partial Network (CSPNet) [13] is based on residual block architecture, which is used in object detection models to extract features. It can extract high-level features from input images effectively. It uses skip connections, which enable the network to bypass layers and incorporate low-level features from earlier layers into the final feature representation. CSPNet improves the model's accuracy, so it's suitable for use as a backbone of detection model.

3.3.2 Feature Pyramid Networks (FPN)

The Feature Pyramid Network (FPN) [14] is a popular feature extractor used in object detection models. It is capable of detecting objects at different scales, thanks to its ability to use feature maps of different sizes. The smaller feature maps have larger receptive fields, allowing them to capture more contextual information about the objects in the input image. At the same time, the larger feature maps combine high-resolution and low-resolution features using upsampling by addition, which improves the accuracy of detecting small objects.

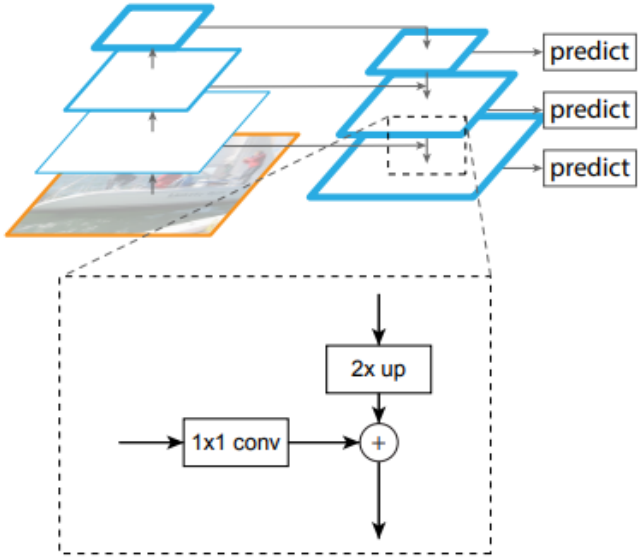


Figure 3.6: Architecture of FPN.

3.3.3 Dense Prediction

The dense prediction produces the final object detection predictions for the bounding boxes and class probabilities for each object in the input image. It includes several convolutional layers and additional components such as upsampling layers and concatenation layers.

3.4 Loss Function

YOLOv5's loss consists of three main parts: box loss, object loss, and classification loss. We introduce as follows.

3.4.1 Box Loss

The object loss measures the difference between the prediction bounding box and the ground truth bounding box. It can help the model learn to find the object's position better.

YOLOv5 model uses the Generalized Intersection over Union (GIoU) Loss [13] as part of our IoU loss function, which is modified by the IoU. It helps to improve the accuracy of our object detection model by taking into account both the geometric relationship between bounding boxes and the overlap between them. Let D be the prediction bounding box and G be the ground truth bounding box. The GIoU Loss is then defined as follows:

$$\mathcal{L}_{\text{GIoU}}(D, G) = 1 - \left(\text{IoU}(D, G) - \frac{|C \setminus (D \cup G)|}{|C|} \right)$$

where C is the smallest enclosing convex object which containing D and G.

3.4.2 Object Loss

The object loss in YOLOv5 use binary cross-entropy (BCE) loss. Let y, $\hat{y}$ be the ground truth category and predicted category. The BCE loss is defined as

$$\mathcal{L}_{BCE}(\hat{y}, y) = -\left[y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \right]$$

3.4.3 Classification Loss

The classification loss in YOLOv5 use Focal Loss, which was proposed by Kaiming's team in 2017 [15]. Their team adjusts the cross-entropy by adding a coefficient that

addresses the data imbalance problem. The focal loss is defined as

$$\mathcal{L}_{FL}(\hat{y}, y) = -\alpha(1 - \hat{y}_c)^{\gamma} \log \hat{y}_c,$$

where c is the category of y, $\hat{y}_c$ is the probability of category c in $\hat{y}$ and $\alpha, \gamma \geq 0$ are hyper-parameters.

3.5 Optimization

Optimization is a critical component of deep learning. Among the many optimization strategies, stochastic gradient descent (SGD) is the most widely-used. Researchers have proposed various algorithms to enhance the stability and efficiency of the optimization process, including our competition's choice: Stochastic Gradient Descent with Momentum (SGDM). This algorithm is known for its stability and ability to converge to a relatively good local minimum.

3.5.1 Stochastic Gradient Descent with Momentum (SGDM)

The SGD with momentum mechanism was proposed by Hinton et al. [16]. It is a linear combination of the previous update gradient and the current gradient, which helps to stabilize the optimization process. We can express the update processing as follows:

$$m_t = \beta m_{t-1} + (1 - \beta) g_t$$

$$\theta_t = \theta_{t-1} - \gamma m_t$$

where θ_t be the model parameters, g_t be the gradient of θ, m_t be the momentum, γ be the learning rate, and $\beta > 0$ be the hyper-parameter.

3.5.2 Learning Rate Schedule

To further improve the performance and help the model converge to a good local minimum, we use a learning rate schedule. At the beginning of training, the warm up method sets a lower learning rate to train for a few epochs. As is known, deep learning models are highly sensitive to the initial weights. Therefore, if we choose a high learning rate, it may cause instability in the resulting model. On the other hand, we adopt the cosine decay to accelerate the convergence of the model. We display the cosine decay with warm up mechanism as the follow:

$$\eta(t) = \begin{cases} \eta_{max} t / T_{warm\_up}, & \text{if } t \leq T_{warm\_up} \\ \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min}) \left(1 + \cos\left(\frac{t - T_{warm\_up}}{T - T_{warm\_up}} \pi\right)\right), & \text{if } t > T_{warm\_up} \end{cases},$$

where η_{max} is the maximum learning rate, η_{min} is the minimum learning rate, t is the current iteration, T_{warm_up} is the warm up iteration and T is the total iteration.

3.6 Post Processing

First, we use a test-time augmentation (TTA) technique to generate multiple samples with various transformations, including HSV transformation, random scaling, and random rotation. We then use non-maximum suppression (NMS) [17] to eliminate redundant bounding boxes and find the best boxes. Finally, we employ weighted box fusion (WBF) [18] to aggregate all predictions, resulting in improved detection accuracy.

Chapter 4

Experiment Result

Figure 4.1 shows the box loss, object loss, classification loss, respectively. The x-axis represents the epoch and the y-axis represents the loss. From the plot, we can observe that the loss steadily converges. In addition, we display the confusion matrix of classification result in Figure 4.2.

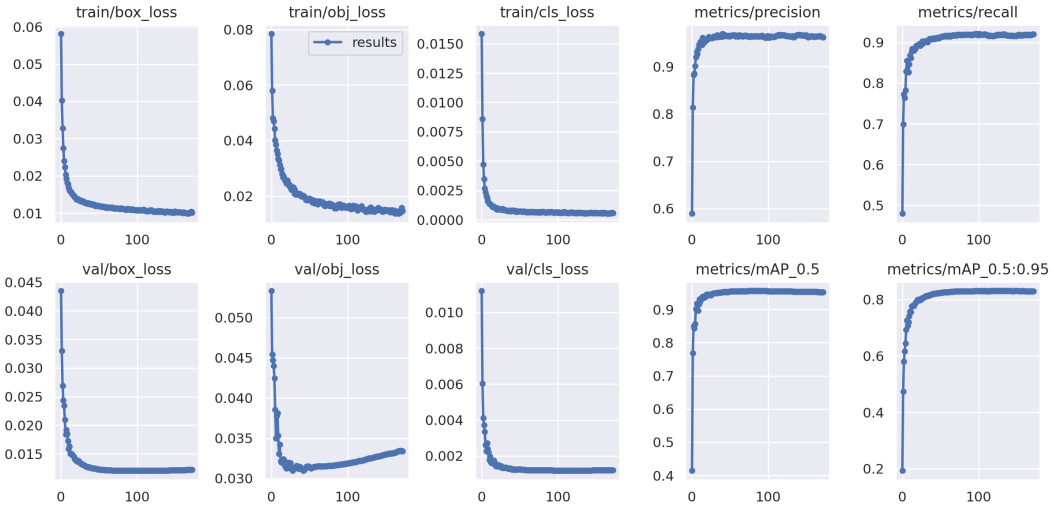


Figure 4.1: Loss curves of training data and validation data by using YOLOv5 model.

We also attempted to use the open-source mmdetection framework to implement the two-stage CascadeRCNN [19] model in this competition. In order to enhance the capabilities of CascadeRCNN, we employed the pre-processing techniques of YOLOX [20] before beginning the training process. As shown in Table 4.1, with the mAP of the model increasing by almost 0.1 after using the sliding window. Additionally, the use of Small object augmentation resulted in an even greater increase of 0.2 in the mAP of both the yolov5 and CascadeRCNN models. Despite these improvements, however, the two-stage CascadeRCNN model still did not outperform yolov5.

Model	Aug.	Resolution	S.W.	B.S.	L.R.	mAP@0.5:0.95
YOLOv5x6	Standard	1920 × 1920		8	0.1	0.51977
YOLOv5l6	Standard	2500 × 2500		8	0.1	0.52984
YOLOv5m6	Standard	3000 × 3000		8	0.1	0.53157
YOLOv5s6	Standard	3840 × 3840		8	0.1	0.53287
YOLOv5l6	-	832 × 832	✓	64	0.1	0.61529
YOLOv5l6	SOA	1664 × 1664	✓	20	0.1	**0.83071**
CascadeRCNN	Standard	3840 × 2160		2	0.0025	0.55901
CascadeRCNN	SOA	1664 × 1664	✓	10	0.025	**0.74139**

Table 4.1: Best result of validation data. Here, Aug. denote the data augmentation algorithm. S.W. denote the sliding window technique. B.S. denote the batch size. L.R. denote the learning rate.

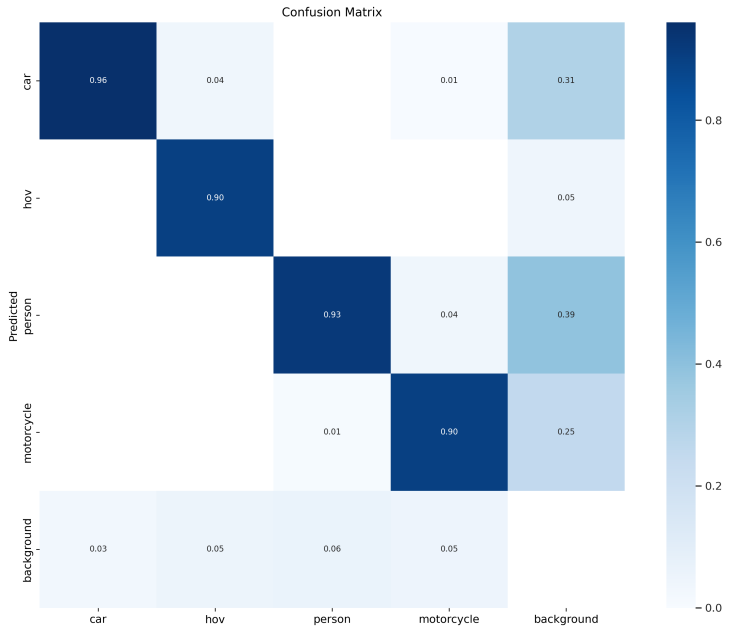


Figure 4.2: The Confusion matrix of the classification result.

4.1 Evaluation Metric

The competition score uses $\text{Hmean}_{\text{TIoU}}$ as the evaluation metric, which is harmonic mean of $\text{Recall}_{\text{TIoU}}$, $\text{Precision}_{\text{TIoU}}$, and $\text{Score}_{\text{dis}}$. Let $\{D_i\}$ be the set of detection bounding boxes, $\{G_i\}$ be the set of ground truth bounding boxes, N_D be the number of the detection bounding boxes, and N_G be the number of the ground truth bounding boxes. We give the formulation of the metric score as follows:

- **Recall of** TIoU:

$$\text{Recall}_{\text{TIoU}} = \frac{1}{N_G} \sum_{i=1}^{N_G} \text{TIoU}_i^{\text{Recall}}$$

where

$$\text{TIoU}_i^{\text{Recall}} = \frac{|G_i \cap D_j|}{|G_i \cup D_j|} \times \frac{|G_i \cap D_j|}{|G_i|}$$

- **Precision of** TIoU:

$$\text{Precision}_{\text{TIoU}} = \frac{1}{N_D} \sum_{i=1}^{N_D} \text{TIoU}_i^{\text{Precision}}$$

where

$$\text{TIoU}_i^{\text{Precision}} = \frac{|G_i \cap D_j|}{|G_i \cup D_j|} \times \left(1 - \frac{|(D_j \cap G_k) \setminus (D_j \cap G_k \cap G_i)|}{|D_i|}\right)$$

- **Distance Score:**

$$\text{Score}_{\text{dis}} = \frac{1}{N_G} \sum_{i=1}^{N_G} \exp\left(\frac{-\|G_i(x,y) - D_j(x,y)\|_2^2}{C}\right)$$

where C is the given positive constant.

Based on the feedback from the public and private scores, we observed that a better detection result for small objects results in a lower score. To improve the score, we submit that performed weighted box fusion on the model that had the highest harmonic mean of validation precision and recall scores with the baseline model, significantly reducing the probability of small objects being detected and resulting in our highest private score.

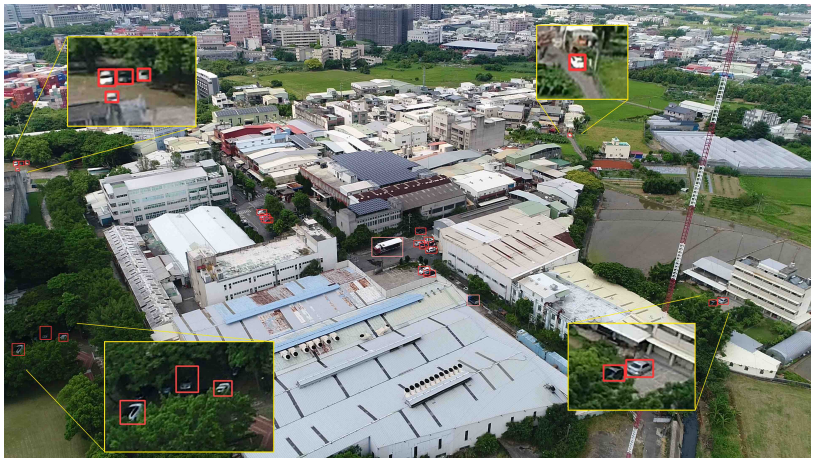
Model	Aug.	Resolution	S.W.	Confident	Public Score	Private Score
YOLOv5l6	Standard	1400 × 2500		0.25	0.708642	-
YOLOv5l6	Standard	1802 × 3250		0.25	0.711584	-
YOLOv5x6	Standard	1400 × 2500		0.25	0.724100	-
YOLOv5m6	Standard	2560 × 4550		0.2	0.730144	-
YOLOv5m6	Standard	2560 × 4550		0.35	0.733960	-
YOLOv5m6	Standard	2560 × 4550		0.35	0.736440	-
YOLOv5m6	Standard	2560 × 4550		0.4	0.736959	-
YOLOv5l6	Standard	2160 × 3840	✓	0.4	**0.739363**	-
YOLOv5l6	SOA	3072 × 5376	✓	0.4	0.724054	0.752086
YOLOv5l6	SOA	3072 × 5376	✓	0.4	0.724934	0.753749
YOLOv5l6	SOA	3072 × 5376	✓	0.4	0.730291	0.754976
YOLOv5l6	SOA	3072 × 5376	✓	0.4	0.725854	**0.754987**
CascadeRCNN	Standard	2160 × 3840		0.4	0.604081	-
CascadeRCNN	SOA	2160 × 3840	✓	0.4	0.660597	-

Table 4.2: Result of the public and private score. Here, Aug. denote the data augmentation algorithm. S.W. denote the sliding window technique in training process.

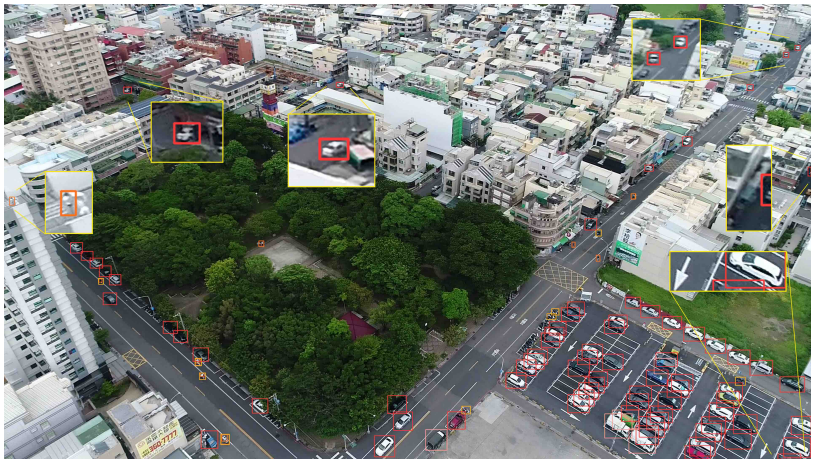
4.2 Displaying of Prediction Results

In this section, we present our model's finest result.

(a)



(b)



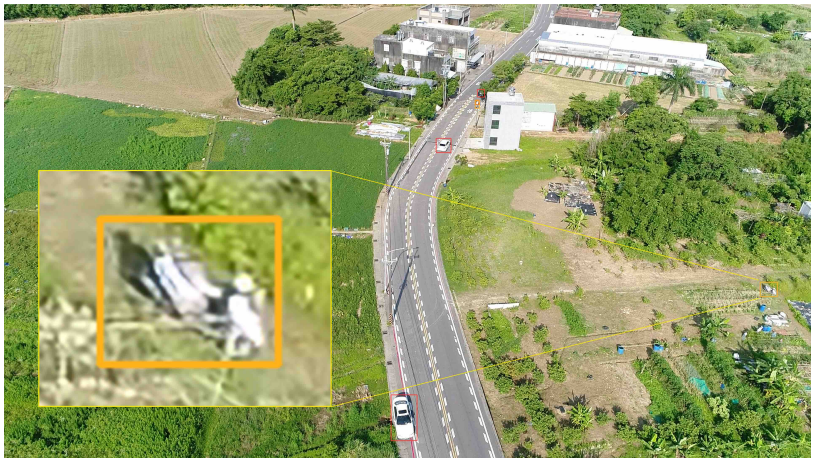
(c)

Figure 4.3: Detection effectiveness of extremely small and distant objects.

(a)

(b)

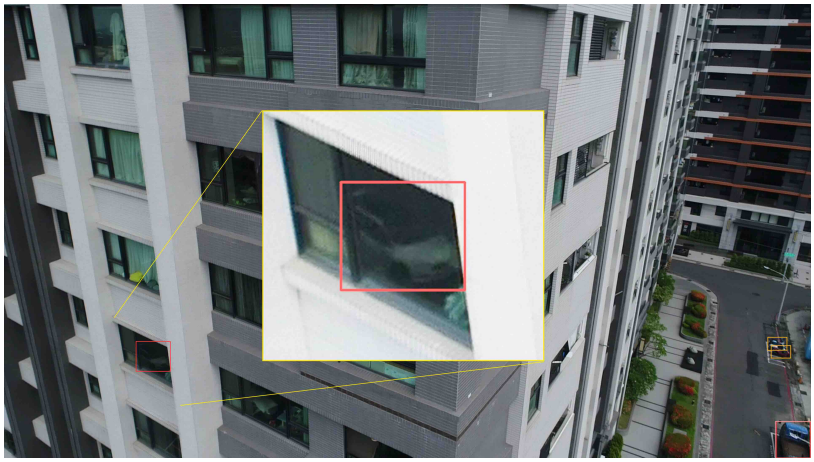
Figure 4.4: Detection effectiveness of severely cut objects



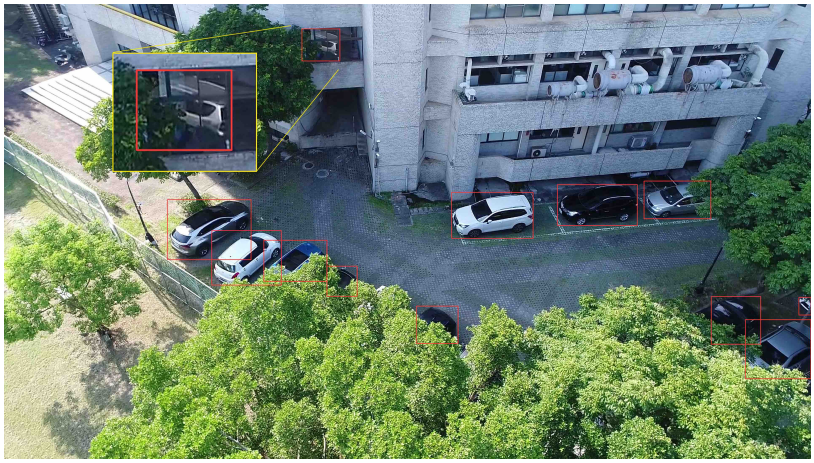
(a)

(b)

Figure 4.5: Detection effectiveness of objects not recognizable by the human vision.



(a)



(b)

Figure 4.6: Detection effectiveness of objects reflected in windows.

Chapter 5

Conclusion

In this competition, we have implemented a state-of-the-art one-stage detection model to build the baseline framework. Our goal was to develop a model that can accurately detect small objects. To address some of the challenges presented by the training dataset, we have implemented several strategies. As mentioned in Chapter 1, we have relabeled the data to ensure that the labels are accurate and complete. This helps improve the performance of the model and enables it to learn more effectively. In addition, we have used the super-resolution technique to enlarge the images, which makes it easier for the model to detect small objects. Moreover, We have also proposed a small object augmentation (SOA) algorithm that generates additional samples, which further improves the model's ability to recognize small objects. To reduce the computation complexity of the model, we have employed the sliding window technique, which allows us to train the model more efficiently. It has improved the training speed of the model. For post-processing, we have used TTA, NMS, and WBF to further improve the performance of the model. As shown in leaderboards in Figure 5.1, we have achieved a public score of 0.739363 and a private score of 0.754987 and are in the top 5 rankings in this competition, which is an encouraging result.

Public Leaderboard | Private Leaderboard

#	隊伍名稱	成員	提交次數	分數	上傳時間
1	TEAM_2334	2	120	0.74645	12/5/2022 8:49:52 AM
2	TEAM_2432	1	114	0.745735	12/4/2022 10:06:20 AM
3	TEAM_2082	1	33	0.739696	11/30/2022 11:39:52 PM
4	TEAM_2144	1	161	0.739406	12/3/2022 9:45:05 PM
5	TEAM_2610	2	44	0.739363	11/30/2022 8:03:44 PM

(a) Public leaderboard.

Public Leaderboard | Private Leaderboard

#	隊伍名稱	成員	提交次數	分數	上傳時間
1	TEAM_2334	2	120	0.765476	12/5/2022 9:48:22 PM
2	TEAM_2432	1	114	0.764277	12/5/2022 6:56:47 AM
3	TEAM_2080	2	64	0.758381	12/2/2022 8:34:48 PM
4	TEAM_2144	1	161	0.756348	12/5/2022 5:17:26 PM
5	TEAM_2610	2	44	0.754987	12/5/2022 11:29:34 PM

(b) Private leaderboard.

Figure 5.1: TBrain leaderboard.

Bibliography

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[5] Cui Gao, Qiang Cai, and Shaofeng Ming. Yolov4 object detection algorithm with efficient channel attention mechanism. In *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 1764–1770, 2020.

[6] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021.

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[8] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, lorenzomammana, tkianai, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, Jake Poznanski, Lijun Yu, changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, hopesala, pritul dave, and yzchen. ultralytics/yolov5: v3.0, August 2020.

[9] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[11] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[12] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[13] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[16] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[17] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6469–6477, 2017.

[18] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021.

[19] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[20] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

Appendix A

The Result for the Small Object Augmentation (SOA)

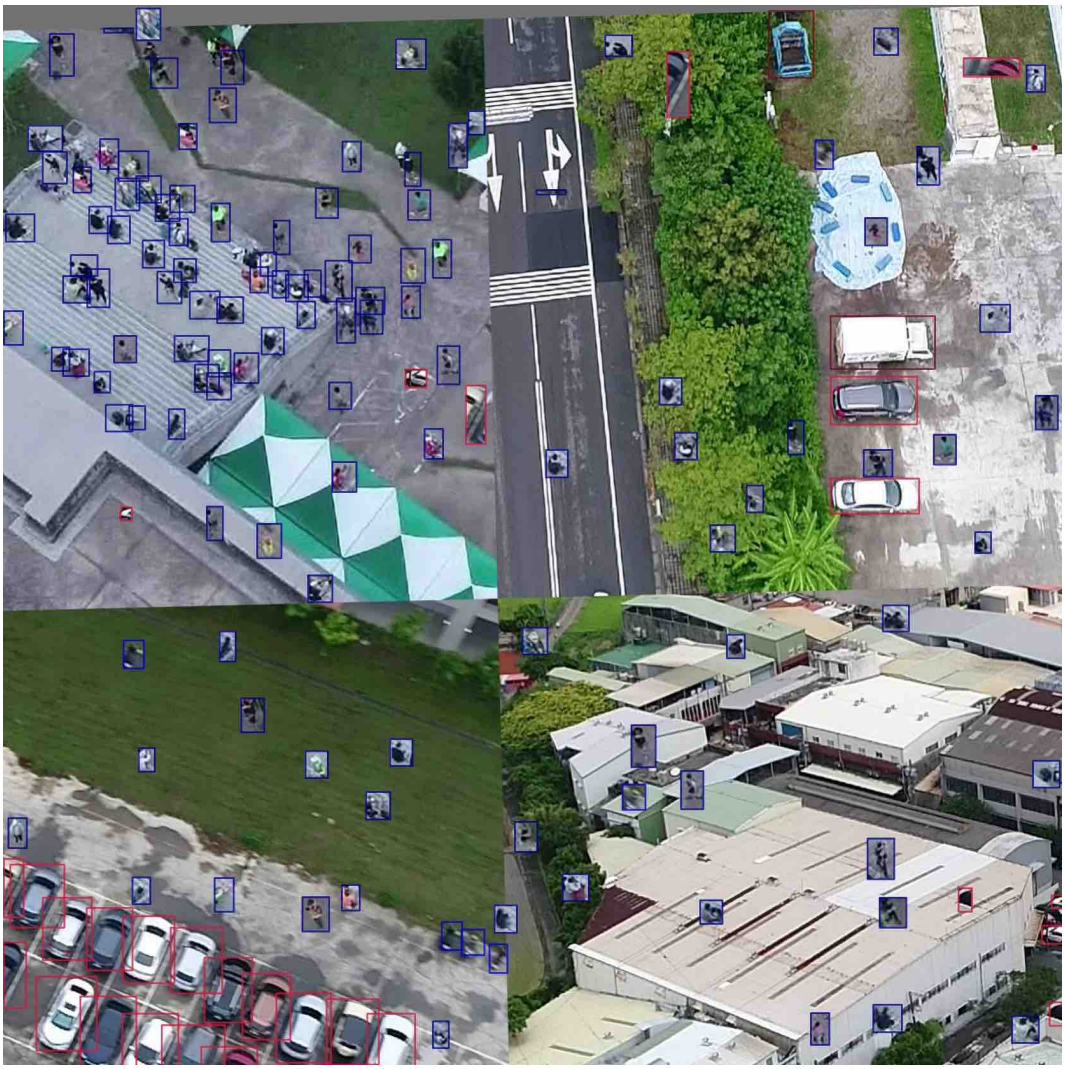


Figure A.1: SOA img0001_3-2.

Figure A.2: SOA img0001_4-1.

Figure A.3: SOA img0009_6-2.

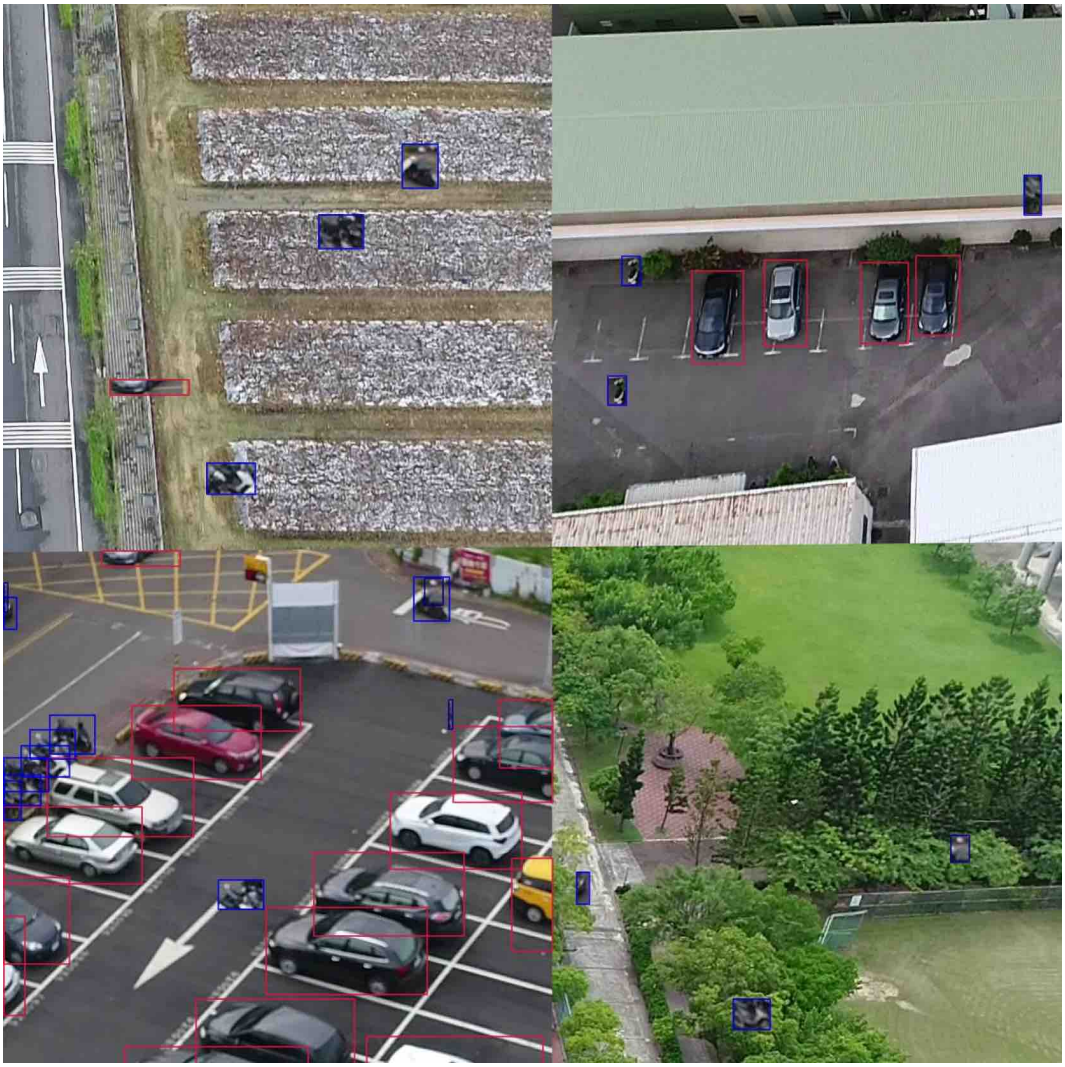


Figure A.4: SOA img0011_4-3.

Figure A.5: SOA img0011_6-1.

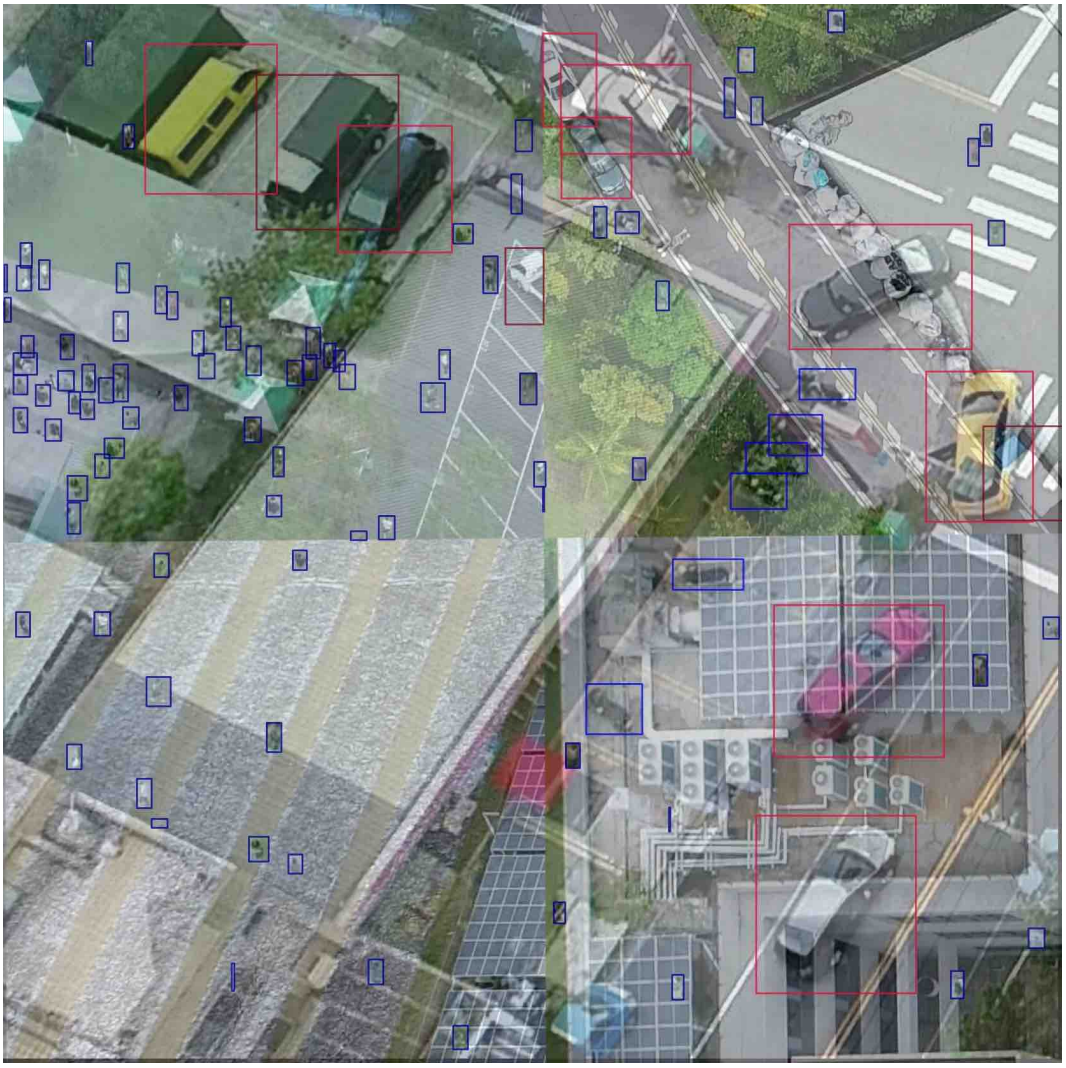


Figure A.6: SOA img0012_3-2.

Figure A.7: SOA img0012_3-2.

Figure A.8: SOA img0012_2-1.

Figure A.9: SOA img0015_5-1.

Figure A.10: SOA img0016_1-1.

Appendix B

Environment Setting

B.1 Operating System

We develop the code on various OS, such as macOS (Ventura 13.0.1) and Linux (Red Hat Enterprise Linux Server release 7.8). All trainings are performed on a server equipped with two NVIDIA Tesla A100 PCIe 40 GB GPUs.

B.2 Third Party Libraries

In this competition, we have utilized the following third-party libraries.

Library	Version
imagesize	
sahi	
scikit-learn	
matplotlib	>= 3.3.2
numpy	>= 1.18.5
opencv-python	>= 4.55.1
Pillow	>= 7.1.2
psutil	
PyYAML	>= 5.3.1
requests	>= 2.23.0
scipy	>= 1.4.1
thop	>= 0.1.1
torch	>= 1.7.0
torchvision	>= 0.8.1
tqdm	>=4.64.0
tensorboard	>=2.4.1
pandas	>=1.1.4
seaborn	>=0.11.0
ensemble_boxes	

Table B.1: Version of libraries

Appendix C

Acknowledgements

We thank the authors of these GitHub repositories, which we use in this competition:

- YOLOv5: https://github.com/ultralytics/yolov5
- MMDetection: https://github.com/open-mmlab/mmdetection
- VisDrone Dataset: https://github.com/VisDrone/VisDrone-Dataset
- Slicing Aided Hyper Inference: https://github.com/obss/sahi
- Weighted Boxes Fusion: https://github.com/ZFTurbo/Weighted-Boxes-Fusion

Appendix D

Contact Information

D.1 Team Information

Team Name	Public Score	Public Rank	Private Score	Private Rank
TEAM_2610	0.739363	5 / 236	0.754987	5 / 236

D.2 Team Member

Name	Institution	Phone	Email
黃靖恩 (Jing-En Huang)	國立臺灣師範大學數學系 (National Taiwan Normal University Mathematics)	0927619756	40840210s@gapps.ntnu.edu.tw
廖家緯 (Jia-Wei Liao)	國立臺灣大學資訊工程研究所 (National Taiwan University Computer Science and Information Engineering)	0939580280	d11922016@csie.ntu.edu.tw

D.3 Advisor

Name	Institution	Position	Email
樂美亨 (Mei-Heng Yueh)	國立臺灣師範大學數學系 (National Taiwan University University Mathematics)	副教授	yue@ntnu.edu.tw