Library Progress International *Print version ISSN 0970 1052*
Vol.44, No.4, Jul-Dec 2024: P. 1455-1463 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

An Enhanced YOLOv8 Approach for Trapped Individual Detection Based On Deep Learning

Yuhong Yang [1], Zhuo Song *[2], Jeffrey S. Ingosan [3]

[1] Student, CITCS, University of the Cordilleras /University, Baguio, Philippines.
[2] Student, CITCS, University of the Cordilleras /University, Baguio, Philippines.
[3] Dean, CITCS, University of the Cordilleras /University, Baguio, Philippines.

How to cite this article: Yuhong Yang , Zhuo Song , Jeffrey S. Ingosan (2024). An Enhanced YOLOv8 Approach for Trapped Individual Detection Based On Deep Learning. Library Progress International, 44(4), 1455-1463

ABSTRACT:

The Philippines faces significant economic losses annually due to natural disasters, impacting the lives of tens of millions of people. The application of deep learning-based object detection technologies can help to enhance the efficiency of rescue operations by enabling the rapid identification of trapped individuals, thereby mitigating economic and human losses. The demand for lightweight, accurate, and portable models suitable for deployment on unmanned aerial vehicles (UAVs) has emerged as a critical research problem. This study introduces an enhanced You Only Look Once version 8 (YOLOv8) model aimed at improving the detection of trapped individuals in UAV-captured images. The C2f layers in the neck and head networks are replaced with the Cross Stage Partial with Focus-Faster (C2f-Faster) module. This adjustment significantly reduces the model's size, facilitating its integration into UAV systems. Furthermore, a 160x160 output head will be added to improve the accuracy of small object detection. A novel loss function combining Scaled Intersection over Union (SIOU) and Complete Intersection over Union (CIOU) is proposed to address issues related to aspect ratio ambiguity and sample imbalance in bounding box regression. The proposed model shows a 25.94% improvement in detection accuracy and a 29.01% increase in inference speed compared to the baseline YOLOv8 model. This study applies a machine learning model to analyze UAV imagery for faster identification of trapped individuals. The findings indicate that this approach can substantially reduce government expenditures on disaster relief, optimize the efficiency of rescue operations, and save more lives.

KEYWORDS:
People Detection, C2f, C2f-Faster, SIOU, CIOU

Introduction:

With its extensive coastline and countless islands, the Philippines is particularly susceptible to natural disasters. [1].There are 6 types of natural disasters currently suffered by the Philippines or the world are: typhoons, storms, floods, landslides, earthquakes, and volcanic eruptions. Millions of dollars in losses each year are not a small amount. The Philippines suffered the greatest economic losses in 2015, reaching 2.3 million of dollars. Every year, tens of millions of people are affected, and these natural disasters have seriously affected people's lives. And every year there are a large number of abandoned children in the Philippines. A large part of the reasons why these children are abandoned are natural disasters [2].

According to a report released by the United Nations Office for Disaster Prevention and Reduction (UNDRR) and the Center for Research on Epidemiology of Disasters (CRED) in 2020, the Philippines ranked fourth in the number of natural disasters worldwide during the period 2000-2019 [3]. In order to reduce disaster losses and improve disaster resilience, this paper applies artificial intelligence technology to personnel search and rescue activities.

There are still some challenges for Yolo model. Pang Chao et al. proposed an improved rice disease recognition and detection model based on the YOLOv8n model and they also point that the size of the model is still very large, and how to carry the model on mobile devices to more effectively complete monitoring tasks is still a problem that scientists need to study [4]. Pedestrian detection in smart community scenarios requires accurate identification of pedestrians to cope with various situations. However, in the case of occlusion and long-distance pedestrians, existing detectors may miss detections, make false detections, and have problems with large models that are difficult to deploy [5]. In the process of mobile phone screen defect detection, there are often problems such as low detection accuracy, high small target defect missed detection rate, and slow detection speed [6]. In general, using the Yolo model for person detection often encounters problems such as high object overlap, small targets, and targets occluding each other, resulting in low model accuracy and slow speed.

The overall objective of this research is to solve these problems and develop an improved YOLO V8 model tailored for aerial person detection in disaster scenarios [7][8]. The improved model can help find trapped people faster, reduce casualties and help the government reduce rescue costs.

1) Related Work:

Various methods have been introduced to enhance feature extraction in models. The YOLO model integrates deformable convolution and varifocal loss to increase the accuracy of pedestrian detection. Experimental findings indicate that the improved model reaches an mAP50 of 0.897, marking a 0.196 increase over the baseline algorithm YOLOv5s [9]. Ajantha Vijayakumar discusses the performance metrics for object detection, along with post-processing approaches, dataset accessibility, detection techniques, and the architectural design across different YOLO versions. In future research work, the author will integrate the ConvNeXt architecture into the YOLOv8 backbone to improve feature extraction performance of YOLO [10].

The high missed detection rate of small targets and covered objects also leads to low model accuracy. Yolo is a function for real-time detection of human behavior, but the model has low accuracy for small targets and high missed detection rate [11][12]. To achieve this, the authors initially strengthen the backbone network using an attention mechanism, followed by the addition of a small-object detection layer to improve key point detection for smaller targets. Finally, they substitute the convolution and SPPF with regionally dynamic-aware deep separable convolution (DR-DP-Conv) and atrous spatial pyramid pooling (ASPP), respectively[13]. Experimental findings demonstrate that the proposed model can reliably and accurately detect anomalies in real-world escalator settings, achieving an mAP50-95 of 0.576. To enhance the detection of small aerial targets, Cao Li and colleagues suggest adding a 160×160 small-object output detection head[14][15].

To address current limitations in the YOLO model, this paper proposes replacing the C2f modules in the neck and head networks with a c2f-Faster layer, aiming to enhance model accuracy through improved feature extraction. Additionally, a 160x160 output head will be incorporated to boost detection rates for smaller targets. The model will also implement a combined SIOU and CIOU loss function to accelerate convergence, thereby increasing training efficiency.

2) Methods and Methodology:

This section mainly introduces the improvement methods of the dataset and model. Specifically, it is necessary to introduce C2f-faster module in the neck and head network, add a output head with size of 160 x160, and use the combined loss function of CIOU and SIOU.

A. The Introduction of Dataset

The datasets used in this article include tiny_set_v2, airPersonDetect and some disaster-affected pictures on the Internet. There are 7,500 pictures in the tiny_set_v2 dataset, 3,000 pictures in the airPersonDetect dataset, and 1,000 pictures on the Internet, so there are a total of 11,500 pictures of the dataset, and the training, validation, and test sets are split in a 70:20:10 ratio. The data are summarized in the following Table 1.

Table 1. The description of dataset

Parameters	Number
Number of tiny_set_v2 dataset	7500
Number of airPersonDetect dataset	3000
Number of pictures on the Internet	1000
Picture number of training set	8050
Picture number of valid set	2300
Picture number of test set	1150

B. Replacing the C2f Module in the Neck and Head Network with the C2f-Faster Module

Data model's performance is inherently tied to its ability to accurately capture relevant features. In this context, the C2f layer constitutes a pivotal module within the YOLO architecture, designed to enhance both the feature extraction efficiency and overall model capability. By incorporating partial cross-layer connections, the C2f layer enables the fusion of features across different stages of the network, thereby retaining more comprehensive information during forward propagation [16]. This mechanism greatly enhances the model's capacity for extracting detailed features and capturing essential details, particularly in complex and dynamic scenes. Furthermore, the C2f layer addresses computational efficiency by reducing redundant information transfer between layers, effectively lowering computational overhead while preserving model performance. This improvement is particularly critical for the deployment of large models in resource-constrained environments, such as embedded systems or unmanned aerial vehicles (UAVs). Additionally, the enhanced multi-scale feature fusion offered by the C2f layer is essential for enhancing the accuracy of small object detection, which is particularly advantageous in applications like crowd detection during disaster response or small object recognition in complex environments.

Figure 1 is the structure of the C2f layer. The right part of the image demonstrates how the C2f structure is formed. It consists of the following parts:

1. Initial Convolution: The initial processing of input data involves passing it through a convolutional layer to obtain basic feature extraction.
2. Split: The feature map is then split into two parts. This splitting is a key part of the Cross Stage Partial (CSP) concept, which limits the amount of information that flows through the network, improving computational efficiency while maintaining sufficient feature diversity.
3. Bottleneck Stacking: One of the split feature maps goes through a series of Bottleneck layers. In the diagram, two bottlenecks are shown, each contributing to further feature compression and transformation.
4. Concatenation: After the bottleneck layers, the output is concatenated (merged) with the feature map that bypassed the bottlenecks, allowing the model to retain both transformed and original features.
5. Final Convolution: Finally, the concatenated feature map is processed by another convolutional layer to combine the information and produce the output.

The left portion of the image shows the architecture of a Bottleneck block. The bottleneck block consists of two consecutive 1x1 convolution layers (Conv 1x1) that reduce the number of channels, allowing for more efficient computation by compressing the input features.

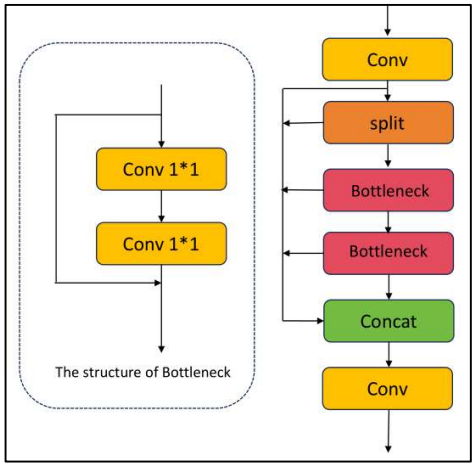

Figure 1. The structure of C2f

C2f-Faster is similar to the C2f structure but incorporates FasterNet blocks. FasterNet Block Stacking improves the network's processing speed while retaining important feature information. The right part of the diagram shows how FasterNet blocks are incorporated into the C2f-Faster structure. Compared with traditional C2f's bottleneck, it has an additional Partial Convolution 3x3 module that likely refers to a PConv layer with a 3x3 kernel. Partial convolutions are designed to improve the efficiency of convolutional operations by only applying convolutions to valid pixels (or data). This makes them more computationally efficient. Figure2 shows the structure of C2f-Faster.

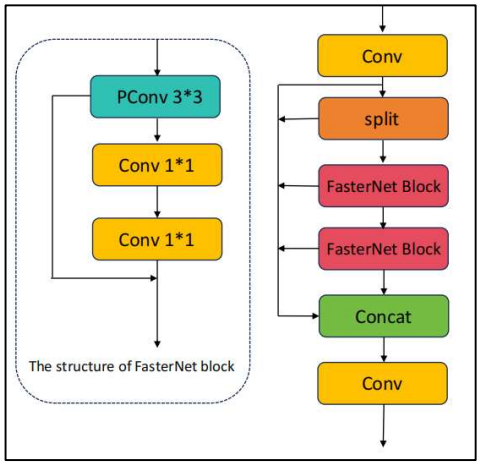

Figure2. The structure of C2f-Faster

Compared with C2f, the main difference of C2f-Faster is that it uses partial convolution (PConv), which only performs convolution operations on valid pixels, reducing unnecessary calculations. Therefore, compared with C2f, C2f-Faster has significantly improved computing efficiency, inference speed, feature extraction capabilities, computing burden, and small target detection. It is particularly suitable for application scenarios that require real-time processing and efficient computing. While maintaining high-precision detection, it significantly reduces dependence on hardware resources. Therefore, C2f-Faster is more suitable for the detection scenarios of small targets and limited computing resources in this paper.

C. Add 160 x 160 Output Head

YOLOv8 has three detection heads by default, which are used for targets of different scales (e.g., 80x80, 40x40, and 20x20)[17][18]. This paper incorporates a 160x160 detection head into the YOLOv8 model to enhance the accuracy of small target detection. In YOLOv8, the typical output layers are P3, P4, and P5. This paper will extract a 160x160 feature map from the P2 layer to form the fourth output head. Specifically, the data features from the P3 layer will be upsampled and combined with those from the P2 layer to create the output for the fourth detection head. The code structure for the enhanced head layer is illustrated in Figure 3.

```
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f_Faster, [512]] # 12

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f_Faster, [256]] # 15 (P3/8-small)

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 2], 1, Concat, [1]] # cat backbone P2
  - [-1, 3, C2f_Faster, [128]] # 18 (P2/4-xsmall)

  - [-1, 1, Conv, [128, 3, 2]]
  - [[-1, 15], 1, Concat, [1]] # cat head P3
  - [-1, 3, C2f_Faster, [256]] # 21 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f_Faster, [512]] # 24 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2f_Faster, [1024]] # 27 (P5/32-large)

  - [[18, 21, 24, 27], 1, Detect, [nc]] # Detect(P2, P3, P4, P5)
```

Figure 3. The code structure of the improved head

D. The Combination of SIOU and CIOU Loss Function

CIOU and SIOU are loss functions for bounding box regression tasks. They make the model more accurate when predicting bounding boxes by considering multiple factors such as overlapping area, distance to the center point, and aspect ratio. This paper emphasizes the use of the combined SIOU and CIOU loss function to enhance the bounding box regression loss for this model.

CIOU takes into account not only the overlap between the predicted and ground truth boxes but also factors such as the distance between their center points and the aspect ratio, allowing for a more precise assessment of bounding box matching. CIOU is particularly suitable for handling overlapping or close objects, and can improve the bounding box prediction accuracy of the model. In this paper, there are a lot of objects that cover each other, so CIOU's good performance in overlapping or close objects can help us deal with the occlusion problem well. The formula for calculating CIOU is given in equation (1). The b and b^g denotes the center points of the predicted box and the target box, respectively. The $\rho^2(b,b^g)$ represents the Euclidean distance between the two center points. The c refers to the diagonal length of the smallest enclosing area that contains both the predicted box and the target box. The width and height of the prediction box are represented by the w and h, and w^g and h^g are the width and height of the target box.

$CIOU_{loss}$ =1-(A∩B)/(A∪B)+(ρ^2(b,b^g))/c^2 +α(4/π^2 *(arctan(w^g/h^g) − arctan (w/h))^2) (1)

SIOU considers the scale (aspect ratio) and direction (angle) of the bounding box when calculating the loss. Because the scale information is taken into account, SIOU is more suitable for small target detection tasks. There are many small target detection situations in our database, Thus, this paper proposes incorporating SIOU alongside the original CIOU bounding box regression loss function of the model. The calculation formula for SIOU is presented in equation (2). Δ can represent the center distance between the predicted box and the true box. Ω is related to the aspect ratio or shape difference between the predicted box and the true box. A and B represent the areas of the predicted box and the true box, respectively.

$SIOU_{loss}$=1- (A∩B)/(A∪B)+ （Δ + Ω）/2 (2)

High object coverage and high small target missed detection rate are two common difficulties encountered in human detection models. CIOU has a relatively good performance when the object overlap is high, and SIOU is good at small target detection. Therefore, this study combines the advantages of the two loss functions and continuously adjusts their respective ratios to make the model have a higher accuracy. The final loss function formula of the model is as follows:

Loss=α$SIOU_{loss}$+β$CIOU_{loss}$ （α+β=1） (3)

E. The Whole Structure of Yolov8

The main improvements to the YoloV8 model in this paper are as follows. The C2f-Faster module replaces the C2f module. Compared with the C2f module, the C2f-Faster module can improve the calculation speed while ensuring a high feature extraction rate. In addition, to enhance the detection rate of small targets, this paper also adds a 160x160 small target output head to the original output head. Finally, the combined loss function of SIOU+CIOU is used to speed up the model's convergence, thereby enhancing the model's training speed. Figure 4 below illustrates the detailed model structure of the enhanced YOLOv8.

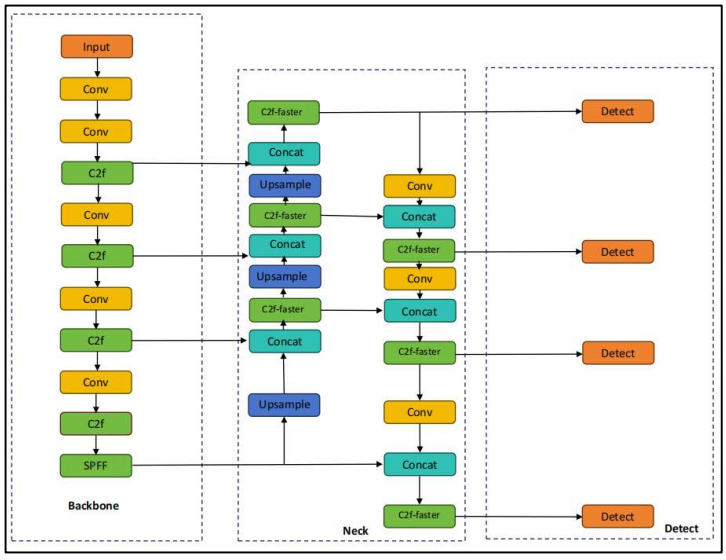

Figure 4. The final specific improved YoloV8 structure

3) Experiments and Results:

This chapter mainly introduces the experimental environment, the experimental outcomes of both the original and improved models, along with a comparison of their results.

A. Experimental environment

Graphics training requires a high-performance GPU, so the GPU used for training the model in this paper is Tesla T4, with a GPU memory capacity of 15110 MiB, about 15 GB of video memory. The memory is generally determined by the size of the dataset image, and 64 GB of memory was purchased. In addition, the basic YOLOv8 model version used in this paper is v8.2.38, The Python interpreter used is version 3.9.19, and the deep learning framework employed is PyTorch, version 1.9.0, which is compatible with CUDA 10.2 (cu102).

The maximum epoch in this study is 700, because sometimes the best results are not achieved after 500 epochs during the model training process. Furthermore, the YOLO model uses an early stopping mechanism to prevent overfitting. Once the model accuracy no longer improves, it will automatically stop training. This strategy also ensures that computing resources are effectively utilized. In addition, the batch size is configured to 8 to optimize the GPU's computational power and memory capacity. When the batch size is too large, the model computing resources are not enough and the model training will be terminated prematurely. The image size is 640. The optimizer used is Adam. Table 2 shows the specific environment configuration.

Tabel 2. The specific environment configuration

Experimental Configuration	Parameters
YOLOv8 version	v8.2.38
PyTorch version	1.9.0
GPU's video memory	15110 MiB
epoch	700
Batch size	8
Image size	640
CPU	64

B. The results of the original model

The original YOLOv8 neural network model contains 218 layers and has a total of 25,840,918 trainable parameters. The model requires 78.7 GFLOPs (giga floating point operations) to complete one forward propagation. Figures 5 and 6 present the performance results of the original YOLOv8 model in the current dataset and the designated experimental environment. As shown in Figure 5 and Figure 6, the model was trained for a total of 370 epochs, and the best performance was observed at the 270th training epoch. The training process was stopped when the mean average precision (mAP) did not improve significantly after 100 detection rounds, indicating convergence. The model achieved a best mAP50 of 0.716, a best mAP50-95 of 0.324, a maximum precision of 0.759, and a maximum recall of 0.632. The trend of the loss curve and the continuous improvement of the precision and recall confirm the effectiveness of the training process. Finally, the training process spanned 370 epochs and took a total of 45.097 hours, with each epoch lasting 7.31 minutes on average.

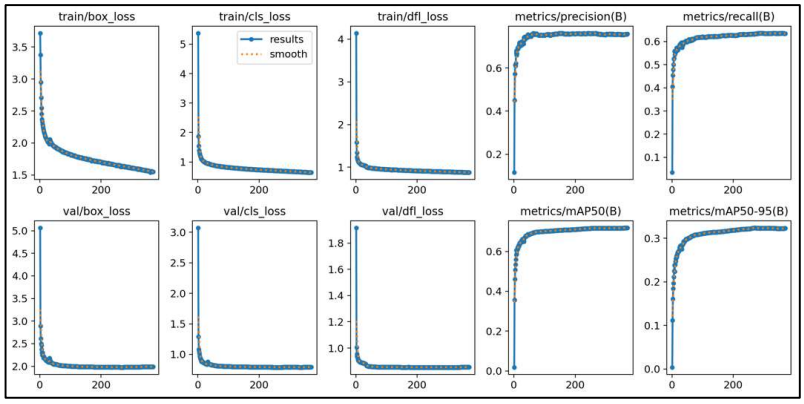

Figure 5. The loss of original YOLOV8 model.

```
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 270, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.
Ultralytics YOLOv8.2.38 🚀 Python-3.9.19 torch-1.9.0+cu102 CUDA:0 (Tesla T4, 15110MiB)
YOLOv8 summary: 315 layers, 23560963 parameters, 23560947 gradients, 77.2 GFLOPs
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████|
             all       1150      42399      0.759      0.632      0.716      0.324
370 epochs completed in 45.097 hours.
```

Figure 6. The running result of original YOLOV8 model.

C. The results of the enhanced model

This paper improves the convergence speed of the model by integrating SIOU and CIOU loss functions, thereby reducing the training time. Initially, the weight factors of SIOU and CIOU were set to 0.5 respectively. Through repeated experiments and adjusting these ratios, it was determined that the weight factor of SIOU was 0.42 and the weight factor of CIOU was 0.58 to obtain the highest mAP50-95 accuracy. Table 3 presents the model accuracy results changing with the adjustment of SIOU and CIOU ratios.

Table 3. MAP variation with adjustment in loss function ratios

CIOU: SIOU	mAP50	mAP50-95
0.5 : 0.5	0.59	0.39
0.6 : 0.4	0.58	0.38
0.4 : 0.6	0.53	0.37
0.58 : 0.42	0.647	0.418
0.7 : 0.3	0.52	0.36

The improved YOLOv8 neural network model contains 232 layers and a total of 8,274,900 trainable parameters. The model requires 31.7 GFLOPs (giga floating point operations) to complete a forward propagation. Figure 7 and Figure 8 show the change curves of the loss function and evaluation indicators of the improved YOLOv8 model during training and verification, including important indicators such as box loss, classification loss, DFL (Distribution Focal Loss), Precision, Recall, mAP50, and mAP50-95 for training and verification. As shown in Figure 7 and Figure 8, the model was trained 500 epochs in total. The model achieved the best mAP50 of 0.647, the best mAP50-95 of 0.418, the maximum precision of 0.786, and the highest recall of 0.526. As training advances, the different loss functions of the model in the target detection task decrease, and indicators such as precision and recall rate are significantly improved. In particular, key performance indicators such as mAP50 and mAP50-95 are also stable, indicating that the model has good convergence and significant performance improvement. Finally, the training process spans 500 epochs, requiring a total of 33.4 hours, with an average of 4.008 minutes per epoch.

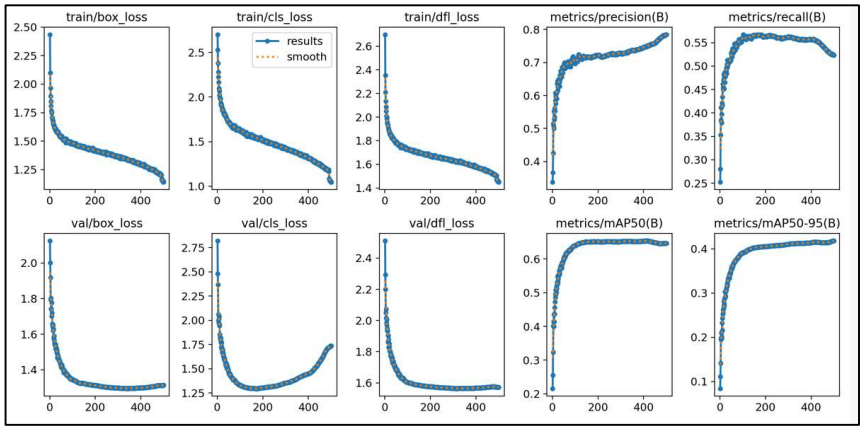

Figure 7. The loss of improved YOLOV8 model

```
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 400, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.
Ultralytics YOLOv8.2.38 🚀 Python-3.9.19 torch-1.9.0+cu102 CUDA:0 (Tesla T4, 15110MiB)
YOLOv8 summary: 315 layers, 23560963 parameters, 23560947 gradients, 77.2 GFLOPs
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████|
                 all       1150      42399      0.786      0.526      0.647      0.418
500 epochs completed in 33.4 hours.
```

Figure 8. The performance results of the enhanced YOLOv8 model

D. Comparison of Model Results

This paper mainly improves the neck and head network of the model and introduces a new C2f-Faster layer, which enhances the model's training speed while maintaining the accuracy of the experimental results. In addition, the loss function of the model is improved and one 160 x 160 output head is added to further improve the model's accuracy in detecting small targets and speed up its convergence. The experimental results show that the mAP50-95 of the enhanced

model is improved by 29.01% ((0.418-0.324)/0.324=0.2901), and the model convergence speed is improved by 25.94% ((45.097-33.4)/45.097=0.2594). Table 4 shows the specific comparison results.

Table 4. Comparison of model results

comparison items	original YOLO	improved YOLO
mAP50-95	0.324	0.418
mAP50	0.716	0.647
best model epoch	270	500
epoch time duration	7.31 (minutes)	4.008 (minutes)
Total time	45.097 (hours)	33.4 (minutes)

4) Conclusion

This study aims to improve both the accuracy and training efficiency of the YOLOv8 model by enhancing the network architecture of its neck and head layers. Specifically, the C2f module has been replaced with the C2f-Faster module, which incorporates the FasterNet block instead of the conventional bottleneck structure, enabling faster training while preserving model accuracy. Given that the dataset comprises drone-captured images—characterized by small object sizes and significant object overlap due to distance and angle a 160×160 output head has been added to enhance the model's detection capabilities for small objects, thus further improving accuracy. Additionally, a combined CIOU and SIOU loss function has been employed to accelerate model convergence, thereby improving the overall training efficiency. Experimental results show a 25.94% increase in training speed and a 29.01% improvement in mAP50-95 accuracy. And the experimental results also show that when the ratio of SIOU and CIOU loss functions is 0.42 and 0.58, the model has the best mAP50-95 value. The enhanced model significantly enhances the efficiency of rescue operations by enabling rapid identification of trapped individuals, thereby reducing overall rescue time. Its application in disaster response automates the processing and filtering of vast amounts of data, alleviating the need for continuous manual monitoring. This approach can significantly reduce government expenditure. Future work will focus on refining model's backbone network to further boost feature extraction and overall model performance.

Acknowledgement

I wish to express my heartfelt thanks to my supervisors for their invaluable guidance and patience during my research. I am also grateful to my friends for their encouragement and constructive feedback during challenging times. Finally, I wish to thank my family for their unwavering support and understanding.

References

[1] Global Disaster Data System. (n.d.). Retrieved from https://www.gddat.cn/newGlobalWeb/#/home

[2] Why so many orphans. (n.d.). Retrieved from https://www.filipino-orphans.org/blog/why-so-many-orphans-part-1/?gad_source=1&gclid=CjwKCAjwm_SzBhAsEiwAXE2Cv2ToBttUH0b-OGvmx8KAVcUJX2H9TKCitABc9j-NZUMimFxGDKM80xoCDDoQAvD_BwE

[3] Yan, J., Gao, N., & Liu, J. (2021). Study on the Philippines natural disaster response plan. Progress in Earthquake Sciences, 51(8), 379-384. https://doi.org/10.3969/j.issn.2096-7780.2021.08.007

[4] Pang, C., Wang, C., Su, Y., & Zhang, Z. (2024). Rice disease detection method based on improved YOLOv8. Journal of Inner Mongolia Agricultural University (Natural Science Edition). Retrieved from https://link.cnki.net/urlid/15.1209.S.20240423.1234.006

[5] Tang, J., Lai, H., & Wang, T. (2024). Improved YOLOv8 pedestrian detection algorithm in long-distance situations. Computer Engineering. https://doi.org/10.19678/j.issn.1000-3428.0068897

[6] Zhou, S., Xu, H., Zhu, X., Huang, X., Sheng, K., Cao, Y., & Chen, C. (2024). Mobile phone screen defect detection algorithm based on improved YOLOv8n: PGS-YOLO. Computer Engineering. https://doi.org/10.19678/j.issn.1000-3428.0069259

[7] Zhou, Z., et al. (2023). Enhancing small object detection with improved YOLO architectures. IEEE Transactions on Image Processing.

[8] Singh, P., et al. (2022). Application of YOLO for drone-based surveillance. Remote Sensing.

[9] Li, J., & Abdullah, M. I. (2024). DP-YOLO: Enhancing pedestrian detection in crowd scenes with deformable convolution and varifocal loss. ISBN 979-8-4007-1821-2/24/03. https://doi.org/10.1145/3672919.3672962

[10] Ajantha, V., & Vairavasundaram, S. (2024). YOLO-based object detection models: A review and its applications. https://doi.org/10.1007/s11042-024-18872-y

[11] Shaffie, A., et al. (2021). YOLO-based tumor detection in medical imaging. Biomedical Signal Processing and Control.

[12] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.

[13] Liu, S., Li, C., Liu, Y., & Wang, Y. (2024). SH-YOLO: Small target high-performance YOLO for abnormal behavior detection in escalator scenes. https://doi.org/10.1587/transinf.2024EDL8011

[14] Cao, L., Xu, H., Xie, G., Li, Y., Huang, X., Chen, H., & Zhu, X. (2024). ASOD-YOLO: An improved aerial small target detection algorithm based on YOLOv8n. Computer Engineering and Science. Retrieved from https://link.cnki.net/urlid/43.1258.tp.20240926.1558.002

[15] Meituan. (2022). YOLOv6: A single-stage object detection framework. arXiv preprint arXiv:2209.02976.

[16] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[17] Ultralytics. (2023). YOLOv8. GitHub Repository.

[18] Sulaiman, N. S., Ibrahim, I. A., & Lee, C. P. (2021). Real-time object detection using YOLOv4-tiny in embedded systems. Journal of Ambient Intelligence and Humanized Computing.