YOLOv911: An Improvement to YOLOv7 for Airborne Object Detection Task

Dion Andreas Solang, Reza Fuad Rachmadi, I Ketut Eddy Purnama
Department of Computer Engineering
Faculty of Intelligent Electrical and Information Technology
Institut Teknologi Sepuluh Nopember (ITS)

***Abstract*—In this research, we present some approach to improve the detection capability of YOLOv7 for airborne objects. Airborne objects appear very small in camera images when they are located at a considerable distance from the camera. However, due to their high speed of movement, it is crucial to detect them while they are still far away. Therefore, to effectively detect these objects with YOLOv7, its small object detection capability must be enhanced. To address this challenge, we proposed several modifications that include changes in the architecture (adding an extra detection head, modifying the feature-map source, and replacing the detection head with a detached anchor-free head), application of bag-of-freebies techniques (anchor recalculation and mosaic augmentation), and change in the inference process (partitioning the image and performing inference on each partition). Through comprehensive experimentation, we have discovered that the combination of replacing the detection head with a detached anchor-free head, and performing inference on partitions yields the most promising results, with a significant increase in mean average precision (mAP) of 46.18% while still maintaining real-time inference speed (greater than 10 FPS). This improvement is notably higher compared to the unmodified plain YOLOv7, which achieved an mAP score of 0%.**

***Index Terms*—Small Object Detection, YOLOv7, Architecture Modification, Bag-of-Freebies Modification, Airborne Object**

Fig. 1: Example Image from AOT Dataset (From [3])

I. Introduction

Autonomous Aerial Vehicle (AAV) has the potential to significantly impact industries, particularly in commercial delivery [1]. To realize this potential, a reliable and efficient Sense and Avoid (SAA) system is needed. While the airspace in which AAVs operate may be relatively sparse, there are still risks of encountering static obstacles or airborne objects such as birds or other drones. With an effective SAA system, we can mitigate these risks and safeguard both the AAV and the valuable cargo it carries during commercial delivery.

As most AAV uses camera as its sensor due to its smaller weight and lower price, there is a need for camera based object detection system for SAA purpose. One problem is that, airborne objects can appear unexpectedly and approach rapidly from long distances. For this reason, airborne objects must be detected as early as possible, which means that they must be detected when their image on the camera were still very small. Thus, the object detector for SAA system must be able to detect small objects.

In this research, we attempt to optimize YOLOv7 to be able to perform airborne object detection. YOLOv7 was chosen due to its ability to perform real-time object detection accurately even under complex outdoor environment. At the time of execution of this research, YOLOv7 had the highest mAP score amongst all published real-time object detector [2]. This research is not about finding the ultimate solution for airborne object detection, but rather an exploration of methods that can be used to enhance YOLOv7 on detecting small objects, which extends to airborne objects.

To optimize YOLOv7, we defined a set of modification to be applied to YOLOv7 that includes modification to its neural network architecture, some applications of bag-of-freebies, and modification on the way the neural network perform inference. To find the best probably optimal solution, we will try combinations of the modification within the set, and choose the one with the highest mAP score.

We will be using The Airborne Object Tracking (AOT) Dataset [3] to train and test the modification combinations. AOT Dataset consist of aerial vehicle flights footage. In this dataset, the resolution of each image are 2048×2448 px, meanwhile the size of the airborne object bounding box can be as small as 4 px (0.00008% of resolution). An example of the dataset can be seen on Fig. 1.

II. Related Works

Several attempts have been made in the past to increase YOLO architecture ability to detect small objects. Here we present some of them.

A. YOLO-Z

YOLO-Z is a modification of YOLOv5 to optimize its ability to detect cone for autonomous racing purpose [4]. YOLO-Z modify the backbone of YOLOv5r5.0 to downscaled DenseNet, while the neck was changed to PANet. These modification results in an increase of accuracy to detect cone that are far away while still being able to do detection in real-time.

B. exYOLO

exYOLO used YOLOv3 as the basis for modification [5]. exYOLO modified the neck of YOLOv3 by adding Receptive Field Block before combining the feature maps. These modifications produce a higher mAP score than plain YOLOv3 on PASCAL VOC2007 dataset.

C. Barunastra ITS' Object Detection System 2022

In [6], a variant of YOLOv4-tiny was applied for the object detection system in an Autonomous Surface Vehicle. This variant of YOLOv4-tiny added an additional detection layer to the architecture, making the originally 2-head architecture of YOLOv4 tiny become 3-head. This change in the architecture improved the detection capability, especially for small object, without sacrificing latency too much.

III. Experimental Setup

A. Computational Resource and Basis Model

To conduct the experiment, we utilized Nvidia RTX 2080 Ti GPU which has 11 GB of VRAM. We performed a pilot test with this hardware and found that with this limited amount of memory, training large YOLOv7 model such as W6, E6, and E2E is infeasible. Thus, we chose normal YOLOv7 with input size of 1600 and training batch-size of 1 as the basis model for modification as it was the model with the largest input size that can be trained with enough free space in the 11 GB VRAM for minor architectural modifications.

B. Dataset

The AOT dataset, consist of more than 11 TB images of drone camera footage. Amongst these 11 TB of data, there are images taken from planned and unplanned encounters with airborne objects. In this research, we sample the data from planned encounters. There are millions of image in the planned encounters. To obtain 400 images for training as explained in section III-A and some more for validation and testing, we use the sampling strategy described in Table I.

TABLE I: Dataset Sampling Strategy

Splits	Total Images	Classes Percentage				
		Airplane	Helicopter	Bird	Drone	Negative
Training	400	23.75%	23.75%	23.75%	23.75%	5%
Validation	100	20%	20%	20%	20%	20%
Test	200	20%	20%	20%	20%	20%

C. Training Setup and Model Selection

To train the YOLOv7 modified according to modification candidates in III-D, we used images sampled as explained in III-B. Each modification was trained with 300 epochs at batch-size 1. For validation, the experiment was repeated with a resampled data thrice, and have the average reported.

We define the best model as the model with the highest mAP@50 score. However, to preserve the real-time capability of the selected model, models that will be considered are only the model that can perform inference with speed more than 10 FPS.

mAP@50 was chosen instead of mAP@50:95 because we don't expect for the neural network to have a tightly fit bounding box prediction. We consider a loose 50% IoU coverage as good enough for a detection.

D. Modifications

Here we present the proposed modification candidates for YOLOv7

1) Mosaic Augmentation: Mosaic Augmentation was reported to increase the mAP score of the model in [7]yolov5. This augmentation is simple to implement. Therefore, we included mosaic augmentation in our set of modifications combine.

2) Anchor Recalculation: Anchor points that were available on the implementation code of YOLOv7 are optimized for COCO2017 dataset. As the AOT dataset distribution are heavily skewed, the anchors need to be adjusted to match the data distribution. For this reason, anchor recalculation is included in our set of modifications.

3) EIoU localization loss: The localization loss used in YOLOv7 is CIoU. Both EIoU and CIoU are designed to solve the vanishing gradient problem of the standard IoU. The advantage EIoU has over CIoU is that when the bounding boxes intersect, EIoU behaves like the standard IoU while CIoU doesn't. The metrics used to evaluate the models like mAP are based on the standard IoU, thus it's better for the loss function to mimic the metric [8]. [8] reported that EIoU performs better on Faster-RCNN than CIoU, making this modification a good candidate to be tested on YOLOv7.

4) Reroute Neck Feature-map Source: The source feature map that was fed on the feature pyramid can be rerouted to a shallower layer of the backbone (look at Fig. 2). By using the shallower layer, the input data will have a shorter propagation through the neural network. This way, less information will be lost through propagation layer-by-layer.

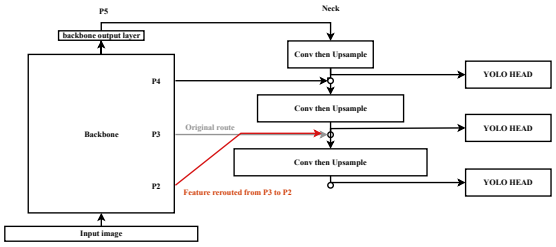


Fig. 2: Rerouting Neck Feature-map Source

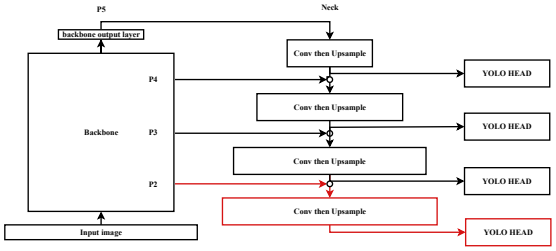


Fig. 3: Adding More Detection Layer

5) Additional Detection Layer: An additional detection layer enable YOLOv7 to detect at more scales. With more scales, the detection layers can be more specialized to specific cluster of data. This approach has been tried in [6] by increasing the number of detection layer from 2 to 3 in YOLOv4-tiny. In this experiment, we will increase the number of detection layer from 3 to 4.

6) Replacing Detection Layer to Decoupled Anchor-Free Head: One of the advantage of anchor-free model to anchor-based model is that it decreases the amount of heuristic tuning parameter as we don't have to define the anchors. [9] [10] reported that using anchor-free head increases the accuracy and decreases the number of parameters in the model, resulting in a faster and more accurate model.

7) Partitioning Image for Inference: Partitioning the image has the potential to improve the accuracy of small object detection. By partitioning an image into multiple smaller images as shown in Fig. 4, the down-scaling factor of the original image to neural network input size is greatly reduced, thereby reducing the loss of information. This approach allows the finer detail like small objects in the data.

One challenge with partitioning is the latency. As the neural network has to perform detection on each partition, the total inference time is multiplied by the number of partitions.

IV. Result

A. Initial Performance

At first, we evaluate the performance of a plain YOLOv7 without all the modifications proposed in section III-D. With 300 epochs and 400 data sample, we find the model was unable to detect anything in the test set (mAP = 0). For the purpose of comparison with other modification combination, we will call this model as `YOLOv7-plain`.

Fig. 4: Image Partitioning

B. Mosaic Augmentation and Anchor Recalculation

In this section, we will compare 3 modification combination

- YOLOv7-plain + Mosaic
- YOLOv7-plain + Anchor Recalculation
- YOLOv7-plain + Mosaic + Anchor Recalculation

We named these models as `YOLOv7+M`, `YOLOv7+AR`, and `YOLOv7+MAR`. We calculated the anchors using k-means clustering algorithm with log-distance function on the training dataset. The result of the recalculation can be seen on Fig. 5. As can be seen on the figure, 8 out of 9 of the old anchor points are placed in the first quadrant of the median line. This means that those 8 anchors responsible only for 25% of the dataset, which is very ineffective. Compare that to the recalculated anchor. Every quadrant has at least one anchor point responsible to it.

TABLE II: Mosaic Augmentation and Anchor Recalculation Performance

No	Model	mAP@50
0	`YOLOv7-plain`	0%
1	`YOLOv7-M`	0%
2	`YOLOv7-AR`	0%
3	**`YOLOv7-MAR`**	**11.2%**
	Improvement	**+11.2%**

Table II shows that the model was only able to detect something on the test dataset after being applied both mosaic augmentation and anchor recalculation. For that reason, this model will be used as the basis for further modification, as it was the only model that could detect objects in test dataset. Therefore, it should be assumed that the modification presented in further sections is composed of mosaic augmentation and anchor recalculation, in addition to the respective modification that was applied, unless explicitly stated otherwise.

C. EIoU Localization Loss

In addition to just using EIoU, we also tested EIoU with its convexication technique mentioned in [8]. The result can be seen on Table III It turns out that EIoU only worsen the performance of YOLOv7 in AOT dataset. One possible reason for this is that: although CIoU have residue of regularization

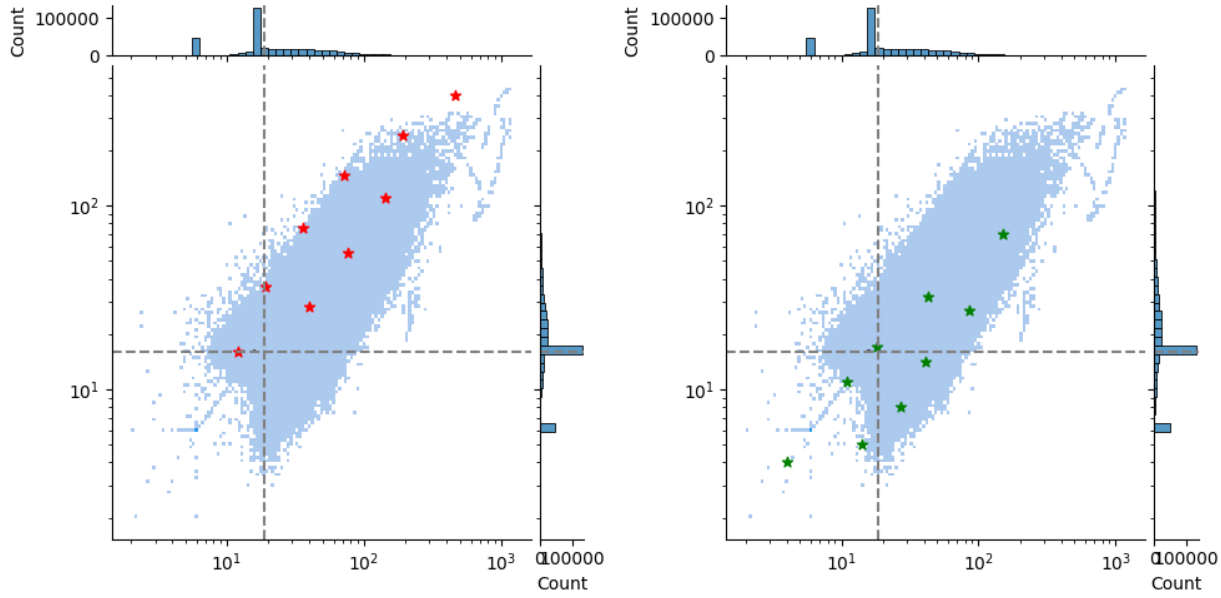


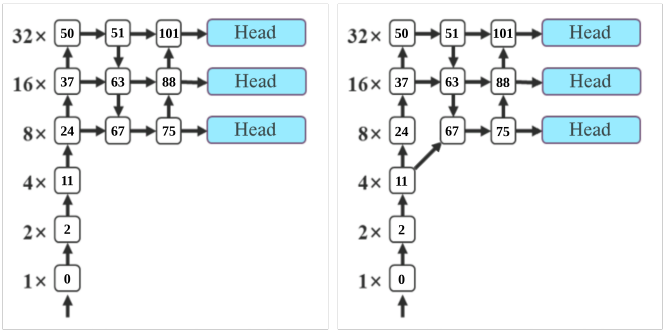
Fig. 5: Anchor Points in Dataset Distribution. Left: Original Anchors. Right: Recalculated Anchors

terms when the bounding boxes intersect, these regularization helped the predicted box to fit faster. EIoU has a weaker values when the boxes not intersect. Perhaps, on a longer training epoch, the EIoU can outperform CIoU.

TABLE III: EIoU Localization Loss Performance

No	Modification	mAP@50
0	**YOLOv7-MAR +CIoU** (original)	**11.2%**
1	YOLOv7-MAR + EIoU	0%
2	YOLOv7-MAR + EIoU + Convexication	4.92%
	Improvement	**-6.28%**

D. Rerouting Neck Feature-map Source



(a) Before Rerouting (b) After Rerouting

Fig. 6: Moving Feature-map Source

We rerouted connection of the first pyramid from scale 8 to scale 4 of the backbone. That is from layer 24 to layer 11 of the configuration file. The moving of this connection is illustrated in Fig. 6. The performance of this modification can be seen on Table IV. This modification managed to increase the mAP score by 2.98%.

TABLE IV: Rerouting Feature-map Source Performance

No	Modification	mAP@50
0	YOLOv7-MAR	11.2%
1	**YOLOv7-MAR + rerouting**	**14.09%**
	Improvement	**+2.98%**

E. Adding Extra Detection Layer

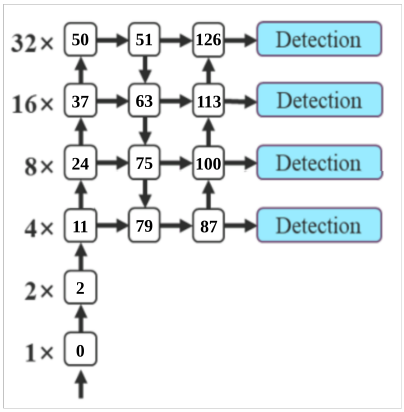


Fig. 7: Adding More Detection Layer

We add an extra feature pyramid stage connected to scale 4 of the backend, and put a detection head on it. This modification is illustrated in Fig. 7. We find that this modification perform worse than rerouting as seen on Table V.

TABLE V: Performance of Adding Extra Detection Layer

No	Modification	mAP@50
0	YOLOv7-MAR	**11.2%**
1	YOLOv7-MAR + more head	5.19%
	Improvement	**-6%**

F. Decoupled Anchor-free Head

We changed the head of the model to anchor-free head with Task Aligned Labelling like in [10]. After some fixing numerical instability due to division by very small object by clipping the values, we found a dramatic increase in mAP as seen on Table VI.

TABLE VI: Anchor-free Head Performance

No	Model	mAP@50
0	YOLOv7-MAR (anchor head)	11.2%
1	YOLOv7-MAR + anchor-free head	0%
1	**YOLOv7-MAR + anchor-free head** *numerically stabilized	**32.98%**
	Improvement	**+21.78%**

G. Image Partitioning

We partitioned the images into 4 equal-sized images. For training data, to reduce the number of negative samples that can impact the recall ability of the neural network, we cropped the images in a way that the objects is within the cropping area for positive sample images and randomly cropping for negative sample images. This cropping strategy was only done in the training set. For the test set, the cropping is done by dividing the image into four equal quadrants.

Since the partitioned image has smaller dimension, we experimented with smaller neural network input size. The performance is shown on Table VII. We can see that YOLOv7+MAR + anchor-free with input size 960 produced the greatest mAP score amongst other model when the images are partitioned into 4.

TABLE VII: Modification Performance on Partitioned Images

No	Model	Input Size	Partition4 mAP@50
0	YOLOv7-AR	960	2.9%
1	YOLOv7-MAR	640	18.46%
3	YOLOv7-MAR	960	37.69%
4	YOLOv7-MAR + rerouting	960	30.04%
5	YOLOv7-MAR + more head	960	10.53%
6	YOLOv7-M + anchor-free	640	37.57%
7	**YOLOv7-M + anchor-free**	960	**46.18%**

H. Latency

Using Nvidia RTX 2080 Ti, we can obtain the speed of inference for each modification in Table VIII We can see that every modified model can perform more than 10 inference per second. Therefore, all the modifications can be considered for model selection.

As an additional information, the floating point operations for an inference of each models can be seen on table IX

V. Conclusion

From the conducted experiment, we found that the best modification amongst the proposed modification candidates are the combination of mosaic augmentation, anchor-free head, and image partition. The resulting modified model was able to produce mAP@50 score of 46.18% on the test set while still able to perform detection faster than 10 FPS on consumer GPU Nvidia RTX 2080 Ti.

VI. Discussion

We have explored different ways to improve the detection of airborne objects in YOLOv7. However, there are still more things we can try in the future to make it even better. One idea is to use attention mechanisms, which help the model focus on important features related to airborne objects. We could also investigate transfer learning and domain adaptation to leverage existing knowledge and adapt the model to specific situations. Overall, there are still many opportunities for further research and enhancements in airborne object detection using YOLOv7.

References

[1] Amazon. (2022) Amazon prime air prepares for drone deliveries. [Online]. Available: https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries

[2] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[3] (2021) Airborne object tracking dataset. [Online]. Available: https://registry.opendata.aws/airborne-object-tracking

[4] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, "Yolo-z: Improving small object detection in yolov5 for autonomous vehicles," 2021.

[5] J. Xiao, "exYOLO: A small object detector based on YOLOv3 object detector," *Procedia Computer Science*, vol. 188, pp. 18–25, 2021. [Online]. Available: https://doi.org/10.1016/j.procs.2021.05.048

[6] N. Aditya, A. Indaryo, D. Solang, M. Santiung, H. Atmaja, A. Azis, F. Januar, F. Javanica, G. Kautaman, E. Kazaksti, D. Nugraha, R. Permadani, R. Ramadhan, R. Ramadhan, Z. Damayanti, M. Valentia, P. Sundana, F. Shodiq, R. Waisnawa, A. Farhan, A. Adifatama, and R. Dikairono, "Roboboat 2022: Technical design report Barunastra ITS roboboat team," https://robonation.org/app/uploads/sites/3/2022/05/RB22_Institut-Teknologi-Sepuluh-Nopember_TDR.pdf, 2022.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[8] H. Peng and S. Yu, "A systematic iou-related method: Beyond simplified regression for better localization," *CoRR*, vol. abs/2112.01793, 2021. [Online]. Available: https://arxiv.org/abs/2112.01793

[9] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021.

[10] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022.

TABLE VIII: Inference Speed of Modified Models

No	Model	Input Size	Inference Speed	YOLOR Reparameterized
0	YOLOv7-MAR, plain, M, AR, EIoU	1600	29.6 FPS	29.6 FPS
1	YOLOv7-MAR + rerouting	1600	24.28 FPS	24.65 FPS
2	YOLOv7-MAR + more head	1600	24.15 FPS	25.01 FPS
3	YOLOv7-MAR + anchor-free	1600	26.42 FPS	26.01 FPS
4	YOLOv7-MAR	Partition 4, 960	72.45 FPS	73.75 FPS
5	YOLOv7-MAR + rerouting	Partition 4, 960	61.35 FPS	62.64 FPS
6	YOLOv7-MAR + more head	Partition 4, 960	60.64 FPS	61.64 FPS
7	YOLOv7-M + anchor-free	Partition 4, 960	63.17 FPS	64.2 FPS
8	YOLOv7-MAR	Partition 4, 640	69.75 FPS	71.78 FPS
9	YOLOv7-MAR + rerouting	Partition 4, 640	73.61 FPS	74.73 FPS
10	YOLOv7-MAR + more head	Partition 4, 640	60.84 FPS	60.92 FPS
11	YOLOv7-M + anchor-free	Partition 4, 640	65.29 FPS	65.34 FPS

TABLE IX: FLOPs for Architectural Modifications with Different Input Sizes

No	Model	Input Size	GFLOPs ($\tilde{\text{M}}$AC$\times$2)
0	YOLOv7-MAR, plain, M, AR, EIoU	1600	552.11
1	YOLOv7-MAR + rerouting	1600	692.02
2	YOLOv7-MAR + more head	1600	627.25
3	YOLOv7-MAR + anchor-free	1600	1374.47
4	YOLOv7-MAR, plain, M, AR, EIoU	960	205.07
5	YOLOv7-MAR + rerouting	960	257.03
6	YOLOv7-MAR + more head	960	232.98
7	YOLOv7-M + anchor-free	960	510.53
8	YOLOv7-MAR	640	89.39
9	YOLOv7-MAR + rerouting	640	112.04
10	YOLOv7-MAR + more head	640	101.55
11	YOLOv7-M + anchor-free	640	222.53