Better Small Object Detection with SOTA modifications on YOLOv8

Pinze Yu, Wenluetao Wu
New York University
{py2050, ww2342}@nyu.edu

Abstract

YOLOv8 [4] has been widely accepted as an improved version of YOLO offering unparalleled performance in terms of speed and accuracy. However, the accuracy is relatively low on small object detection. In our project, we modified some structures in YOLOv8 to improve the performance of small object detection without a significant increase in the number of parameters and layers. In particular, we modified the YOLOv8 structure in three main scopes: the loss function, the convolutional layer, and the additional attention layer. Tested on a customized infrared dataset [2] with small objects, the overall mAP50 accuracy improved by a noticeable amount. The code repository is available here: https://github.com/Pinze-Yu/CV_Final_Project.

1. Introduction

Small object detection has been an important topic in detection for a long time. In the COCO dataset [5], small objects are defined as objects whose bounding box is 32 $\times$ 32 pixels or less in a 480 $\times$ 640 image. In reality, it also has applications in various fields, including video surveillance, traffic flow analysis, and object tracking. While many groundbreaking works were done, most of them were either fast or accurate, yet it was hard to obtain both a fast and accurate model.

There are a few possible reasons that bring difficulties to small object detection. First, most state-of-the-art objection detection algorithms, including YOLOv8, pass the input image through convolution layers to extract features. Though it will help the model learn abstract features, the size of the target object gets reduced as well. Thus, the small object may vanish after convolution and become undetectable. Secondly, as the small object contains fewer pixels, the probability that it suffers from overlapping increases.

In this project, we use the mAP50 accuracy of the original YOLOv8 as the baseline of our small object detection task. We then apply modification in different aspects of YOLOv8 one at a time, which includes Normalized Wasserstein Distance Loss [13], Dynamic Snake Convolution [7], and Triplet Attention [6], and we compare their performances with the baseline performance from YOLOv8. Overall, the mAP50 performance was improved by 5% to 9%, and further progress is also expected for any possible combinations of the listed SOTA modifications.

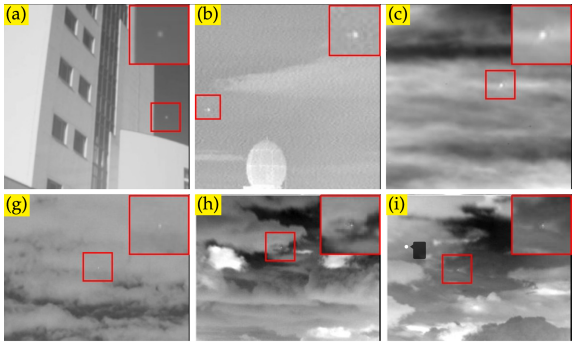
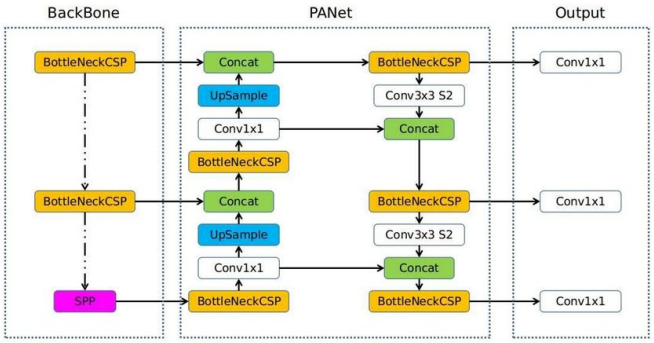


Figure 1. Sample frames from the SIRST dataset. Image from [2]

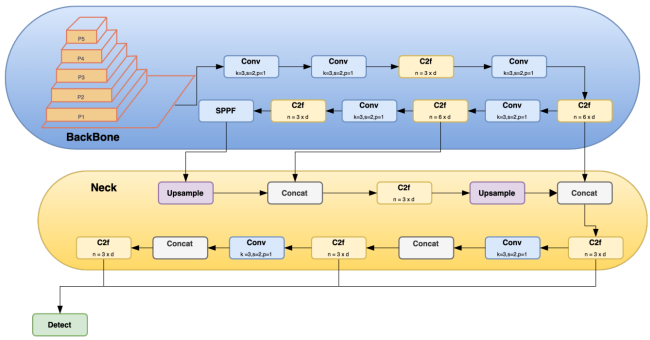
2. Dataset

In this project, we used the Single-frame InfraRed Small Target **(SIRST)** dataset provided by Y. Dai *et al*. [2]. SIRST dataset includes 427 infrared pictures selected from hundreds of infrared sequences from different scenarios. From Figure 1, we can see that the frames are mostly distant flying objects in the sky that only take several pixels, and some of them are even hard to recognize for humans.

Due to the limited amount of frames we have, it is reasonable to do proper data augmentation to introduce more randomness to the dataset. In particular, we apply rotation, cropping, shifting, and horizontal and vertical flipping to the original frames. With five different types of augmentation randomly applied, we obtained 2562 images in total, and we split them into train, test, and validation data with a ratio of 8:1:1. The final dataset has 2049 frames in the training set and 128 frames in the test and validation set respectively.



(a) Illustration of YOLOv5 model structure. Image from [3]



(b) Visualization of YOLOv8 model structure. Image from [11]

Figure 2. Comparison of YOLOv5 & YOLOv8 model architectures.

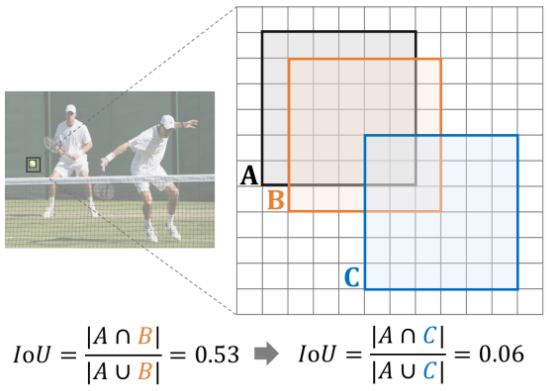


Figure 3. The bad performance of IoU on small objects with tiny deviation. Box A denotes the ground truth bounding box, and boxes B and C denote the predicted bounding box with 1 pixel and 4 pixels diagonal deviation respectively. Image from [13]

3. Methods

3.1. YOLOv8

The YOLO (You Only Look Once) series is a revolutionary approach in object detection, began with YOLOv1 in 2015 by J. Redmon *et al*. [9], YOLOv1 introduced a novel concept: treating object detection as a single regression problem, directly mapping image pixels to bounding box coordinates and class probabilities. One influential successor was YOLOv5, which introduced several architectural changes, most notably the standardized practice of structuring the model into the backbone, neck, and head [3, 12]. Although it was not an official release from the original YOLO creators, it still gained popularity due to its ease of use and deployment which was attributed to Ultralytics.

Compared with YOLOv5, YOLOv8 introduced the C2f module to replace to C3 module in YOLOv5 and changed the dimension of the first convolutional layer in the backbone and bottleneck [8]. Also, YOLOv8 used a new neural network architecture that concatenated both Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) [10]. The main idea of FPN is to gradually reduce the spatial resolution of the input image while increasing the number of feature channels, which generates feature maps that can detect objects with different scales and resolutions. The PAN architecture, on the other hand, uses skip connections to aggregate features from different layers of the network. The application of PAN is also beneficial for capturing features at multiple scales and resolutions, which is crucial for accurately detecting objects of different sizes and shapes [10, 12].

3.2. Normalized Wasserstein Distance Loss

The Normalized Wasserstein Distance (NWD) loss is a new metric proposed by J. Wang *et al*. [13]. The core invention of the work is that they first model the bounding boxes as 2D Gaussian distributions, and then use NWD loss to compute the similarity between the boxes by their corresponding Gaussian distributions. One key motivation for using NWD loss instead of conventional IoU-based metrics is that IoU-based metrics are very sensitive to the location deviation of the tiny objects. As shown in Figure 3, the IoU for the ground truth bounding box and deviated bounding box has dropped drastically with a tiny move of 3 pixels.

Compared with IoU, NWD has two major advantages for detecting small objects: First, the smoothness w.r.t. location deviation, and second, the capability of measuring the similarity between both non-overlapping and mutually inclusive bounding boxes. Figure 4 is the result of an experiment showing the advantages mentioned above. As we can see in the curves, the curve of NWD is much more smooth and it can consistently reflect the similarity between A and B even if they are mutually inclusive ($|A \cap B| = A$ or B) or non-overlapping ($|A \cap B| = 0$).

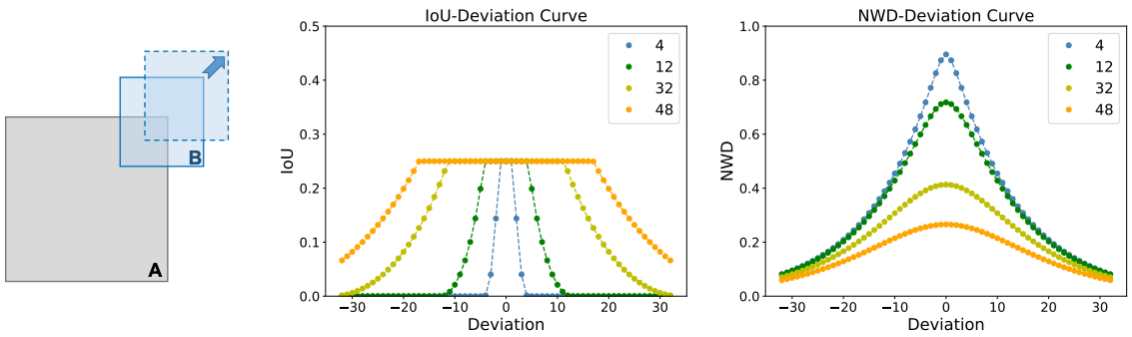


Figure 4. Comparison between IoU-Deviation Curve and NWD-Deviation Curve. The side length of B is half of the side length of A, and we move away B from A along the diagonal of A. The abscissa value denotes the number of pixels deviation between the center points of A and B, and the ordinate value denotes the corresponding metric value. Image from [13]

3.3. Dynamic Snake Convolution

Dynamic Snake Convolution (DSC) is a novel convolution technique proposed by Y. Qi *et al*. [7]. Inspired by the deformable convolutional networks (DCN) [1], the basic idea of DSC is to adapt the convolutional kernel to the geometric deformations of different targets. However, unlike DCN which let the network learn geometrical changes completely freely and leads to perceptual regions wandering on **thin tubular structures**, DSC applies constraints to the learning process so that the convolutional kernel will be able to consider the tubular morphology.

In Figure 5, the comparison between standard, dilated, deformable, and the novel snake-shaped convolutional kernel has shown why the DSC is more suitable for topological tubular structures. The standard convolutional kernel is designed to extract local features, and the dilated kernel is used to expand the receptive field. Based on these, the deformable kernel is designed to solve the problem of limited geometric transformation in ConvNets. However, due to its structural limit, it still failed to catch thin and tubular local structures. The DSC managed to adaptively focus on the slender local features and realize a more accurate segmentation of tubular structures.

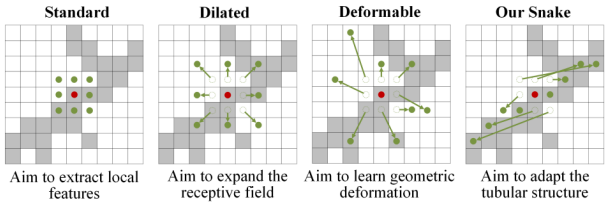


Figure 5. Visualized demonstration of different convolutional kernel's performance on tubular structures (marked as grey pixels). Image from [7]

3.4. Triplet Attention

Attention mechanisms have been widely used in various computer vision tasks. Triplet attention is a light-weight but effective attention mechanism proposed by Diganta Misra *et al*. [6]. In traditional attention mechanisms, to obtain the attention weights of different channels, spatial attention is generated by reducing the input tensor to a single-channel output utilizing Global Average Pooling (GAP) and Global Max Pooling (GMP), which causes loss in spatial information. A later released model, CBAM [14], focusing on both spatial and channel attention, although reduced the loss in spacial information, still separated the spatial and channel information. The core contribution of Triplet Attention is proposing the cross dimension interaction and calculating this interaction using a three-branch structure. As shown in Figure 6, Triplet Attention consists of three separate branches, each tasked with the role of capturing the inter-

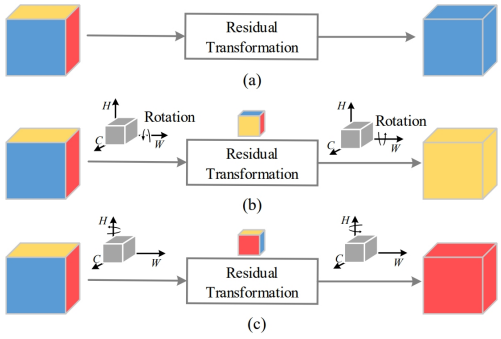


Figure 6. Abstract representation of Triplet Attention with a three-branch structure. Given the input tensor, Triplet Attention captures inter-dimensional dependencies by rotating the input tensor followed by residual transformation. Image from [6]

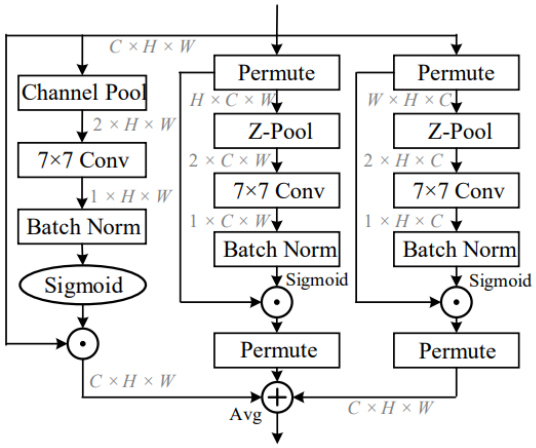


Figure 7. The structure of Triplet Attention. The symbol $\otimes$ represents matrix multiplication, and $\oplus$ denotes broadcast element-wise addition. Image from [6]

dimensional dependencies by applying rotation operation and residual transformations, and encodes inter-channel and spatial information with negligible computational overhead. Consider an input tensor that has the dimensions of channels (C), height (H), and width (W), and two branches out of three will focus on combining features across one spatial dimension(either height(H) or width(W)) with the channel dimension. This is achieved by using a *Z-pool* plus a convolutional layer with kernal size of $k \times k$ after first permuting all the input tensors of each branch. The other branch functions similarly to CBAM [14], which constructs the spatial attention. All three branches then apply a sigmoid function to generate weights, and permute the results back to the original shape. The final attention weight would be the average of the outputs of all three branches (See Figure 7 for interpretation).

4. Experiments

4.1. YOLOv8

In this project, we used the mAP50 accuracy of YOLOv8 on the augmented SIRST dataset as the baseline. Due to the limited computational power we have, we used the nano version of YOLOv8 (YOLOv8n) which contains 3005843 parameters. Although other sizes of YOLOv8 may provide higher accuracy, the number of parameters is also surging.

To make the result reproducible, we kept the most training parameters as the default value provided by Ultralytics [4] with only a few modifications. First, all our experiments run 200 epochs. Second, we conducted an experiment on YOLOv8 to find an appropriate batch size for our future experiments. Table 1 reflects the result of our experiment, and we decide to use a batch size of 32 based on the result.

Batch Size	mAP50	mAP50-95
16	0.522	0.250
32	**0.553**	**0.259**
64	0.527	0.249

Table 1. Original YOLOv8 accuracy comparison for different batch sizes. A batch size of 32 gives the best result and is thus set as our default training batch size.

4.2. Normalized Wasserstein Distance Loss

To implement the normalized Wasserstein distance loss, we first introduce the original NWD loss function [13] into the YOLOv8 loss functions. One noticeable trick we used to align the Wasserstein is that we still make use of the YOLOv8 default Complete IoU (CIoU) by scaling and concatenating the loss from CIoU and NWD loss together. With this trick, we can directly apply NWD loss to the original training process pre-implemented by Ultralytics [4]. The experiment uses 200 epochs, a batch size of 32, and all other parameters being the default value.

4.3. Dynamic Snake Convolution

The implementation of Dynamic Snake Convolution (DSC) into YOLOv8 focuses on changing C2f layers in the head of YOLOv8. In particular, in addition to the DSC structure, we also need to implement a new C2f and bottleneck structure for the DSC to align with the dimensions and structures in the original YOLOv8. The experiment uses 200 epochs, a batch size of 32, and all other parameters being the default value.

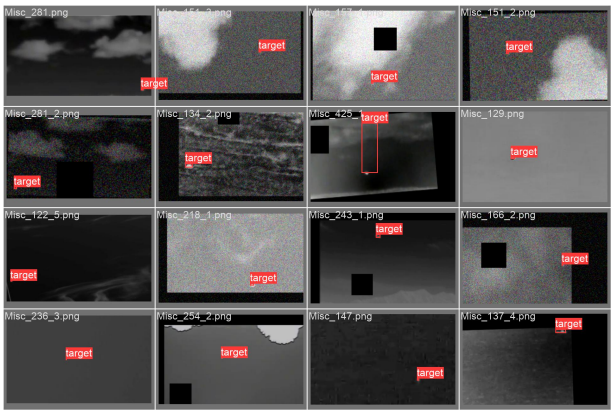
4.4. Triplet Attention

To incorporate Triplet Attention into YOLOv8, the essential step is to add the new attention block into the head of YOLOv8. To achieve this, we implemented the *TripletAttention* class together with the *Z-pool*, *BasicConv*, and *AttentionGate* it needs to utilize in the modules of YOLOv8, based on Triplet Attention's structure [6]. By adding the new attention block to the head of YOLOv8, the experiment could be done using 200 epochs, a batch size of 32, and all other parameters being the default value.

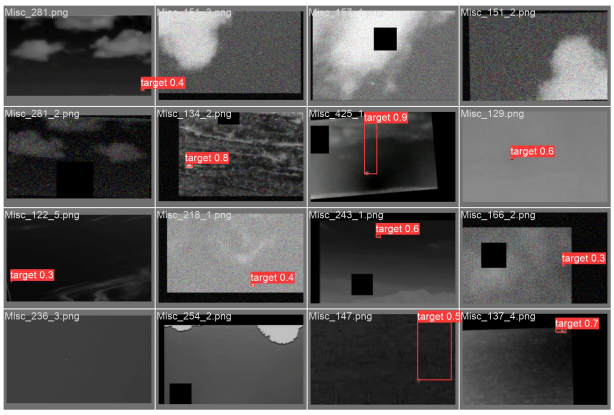
5. Results

5.1. YOLOv8

As the baseline of our project, the original YOLOv8 model reaches a decent mAP50 score of 0.553. Although it is not super accurate, the potential of the original YOLOv8 is still considering that we are only using the nano (smallest scale) version of the model, and YOLO itself prioritizes detection speed over precision. Figure 9a gives the Precision-Recall curve of our baseline model.

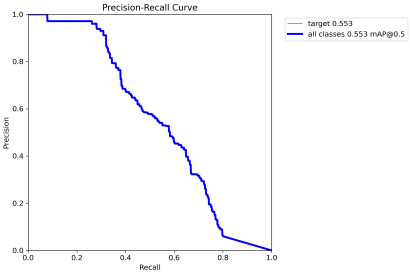


(a) Ground truth labels for the sample images.

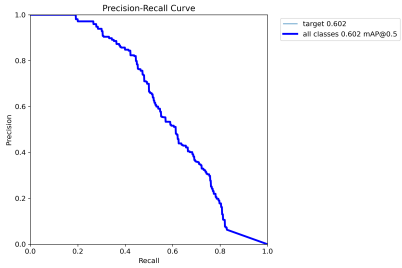


(b) Predictions of the sample images of YOLOv8 model with NWD loss

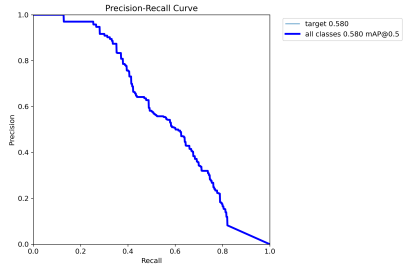
Figure 8. Sample images from the validation set. Model used for prediction is YOLOv8 with NWD loss.



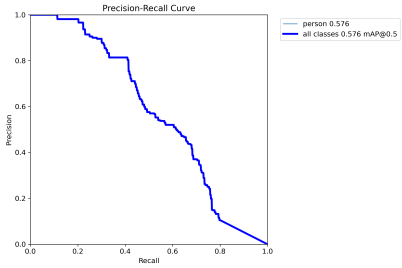
(a) PR curve for original YOLOv8n structure. mAP50 = 0.553



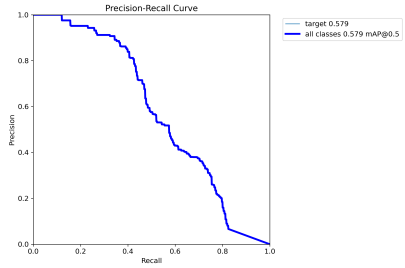
(b) PR curve for YOLOv8n with NWD loss. mAP50 = 0.602



(c) PR curve for YOLOv8n with DSC-C2f layers and bottlenecks. mAP50 = 0.580



(d) PR curve for YOLOv8n with Triplet attention layer PR curve. mAP50 = 0.576



(e) Future work: PR curve for YOLOv8n with NWD loss and DSC-C2f layers. mAP50 = 0.579

Figure 9. Results of the experiments. We use the Precision-Recall (PR) curve and mAP50 metrics to compare the performance.

5.2. Normalized Wasserstein Distance Loss

As the NWD loss is designed for small objection detection, it is not surprising that the mAP50 metric has an improvement of 8.86% from 0.553 to 0.602. One advantage of the modification of the loss function is that the number of parameters, as well as the whole model structure, remain unchanged (See Table 2). Considering the fact that changing the loss function will not affect the skeleton of the model, the improvement in the performance is almost free of charge. As a result, such modifications are more preferable especially when the computation power is limited.

5.3. Dynamic Snake Convolution

Overall, the improvement of DSC-C2f modifications is not as large as the modifications of loss functions. The mAP50 metric increased from 0.553 to 0.580 which is about 4.88%. Although there is still a decent improvement, it is

Modification(s)	mAP50	mAP50-95	# Parameters	# Layers
Original YOLOv8n	0.553	0.259	3005843	168
NWD Loss	0.602	0.269	3005843	168
DSC-C2f layers	0.580	0.294	3356287	229
Triplet Attention	0.576	0.275	3006443	201
NWD & DSC	0.579	0.271	3356287	168

Table 2. Summary of the project results, including mAP50, mAP50-95, number of parameters, and number of layers for the modifications on YOLOv8 structure. All of them improved from the baseline performance. However, the combined modification of NWD loss and DSC-C2f layers did not outperform either of the single modifications itself.

worth noting that, unlike the NWD loss modification, the implementation of DSC naturally increases the number of parameters and layers in the model. As shown in Table 2, using DSC-C2f will lead to 61 more layers and 11.66% more parameters. In addition, even though there are a few tubular targets in the frames of the SIRST dataset (See Figure 8), most targets' shapes are still regular or dotted. Thus, DSC may not perform effectively on the normal-shaped targets. Taking all these into account, DSC is not very suitable for usage in our task.

5.4. Triplet Attention

With the incorporation of Triplet Attention, the mAP50 metric increased 4.16%, from 0.553 to 0.576. Although the performance of Triplet Attention is not as good as previous mechanisms, it successfully limits the increase in parameters, which is only 600 more than the original YOLOv8(See Table 2). Thus, Triplet Attention does provide a good solution in terms of attention mechanisms, for its being lightweight as well as effectively boosting the accuracy of small object detection. This feature would provide it with compatibility with other structures and methodologies. As a result, we consider Triplet Attention suitable for our task.

6. Future Works

6.1. Combination of Modifications

After several trials of single modifications, it is natural to think about concatenating some of them together to further improve the performance. As a pilot test, we tried applying both NWD loss and DSC-C2f together to the YOLOv8 model since there is no problem aligning intermediate dimensions between modified layers. However, as shown in Table 2, the accuracy does not outperform compared to either of the single modifications. As a result, we believe a lot of work can be done in this regard in the future, including further combinations of modifications on different layers.

6.2. Deformable Convolutional Networks

As the outcome of DSC-C2f modification is not as good as expected because of the lack of tubular targets, maybe its inspiration, the Deformable Convolutional Networks (DCN) [1], could be more effective. However, the official implementation of DCN uses both C++ and Python, and the wheels provided required a lower version of PyTorch and CUDA than what we obtained on NYU HPC. As a consequence, we were not able to implement DCNv3 modifications onto YOLOv8 yet. However, it is indeed a possible structure that could bring improvements to small object detection.

6.3. Theoretical Analysis

As shown in our results, the NWD loss improved the overall performance the most. We have given a general introduction to the algorithm, but a closer theoretical and mathematical analysis of the novel loss function would help explain the principles behind it. Even though most findings in the machine learning field nowadays are empirical, it is always a good idea to try to find the principles of an experimentally successful algorithm.

7. Conclusion

In this project, we examined the performance of YOLOv8 on small object detection with state-of-the-art modifications, including Normalized Wasserstein Distance Loss [13], Dynamic Snake Convolution [7], and Triplet Attention [6]. By improving the model from different perspectives (loss function, convolutional layers, attention layers), we conclude that all these modifications have more or less improved the accuracy of detecting small objects. In the future, there are possibly more effective ways of combining the modifications, and the performance of YOLOv8 is subject to further improvements.

Acknowledgements

We would like to thank the NYU HPC team for providing high-performance GPU clusters. We also sincerely appreciate Professor Rob Fergus for all the informative lectures and invaluable guidance given throughout the semester.

References

[1] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017. 3, 6

[2] Yimian Dai, Yiquan Wu, Fei Zhou, and Kobus Barnard. Attentional Local Contrast Networks for Infrared Small Target Detection. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–12, 2021. 1

[3] Daniel Dlužnevskij, Pavel Stefanovič, and Simona Ramanauskaitė. Investigation of yolov5 efficiency in iphone supported systems. *Baltic Journal of Modern Computing*, 9, 01 2021. 2

[4] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, Jan. 2023. 1, 4

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 1

[6] Diganta Misra, Trikay Nalamada, Ajay Uppili Arasanipalai, and Qibin Hou. Rotate to attend: Convolutional triplet attention module, 2020. 1, 3, 4, 6

[7] Yaolei Qi, Yuting He, Xiaoming Qi, Yuan Zhang, and Guanyu Yang. Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation, 2023. 1, 3, 6

[8] RangeKing. Brief summary of yolov8 model structure, 2023. 2

[9] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2

[10] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8, 2023. 2

[11] Nabin Sharma, Sushish Baral, May Paing, and Rathachai Chawuthai. Parking time violation tracking using yolov8 and tracking algorithms. *Sensors*, 23:5843, 06 2023. 2

[12] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond, 2023. 2

[13] Jinwang Wang, Chang Xu, Wen Yang, and Lei Yu. A normalized gaussian wasserstein distance for tiny object detection, 2022. 1, 2, 3, 4, 6

[14] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018. 3, 4