# Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation

Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, Wangmeng Zuo



***Abstract*—Deep learning-based object detection and instance segmentation have achieved unprecedented progress. In this paper, we propose Complete-IoU (CIoU) loss and Cluster-NMS for enhancing geometric factors in both bounding box regression and Non-Maximum Suppression (NMS), leading to notable gains of average precision (AP) and average recall (AR), without the sacrifice of inference efficiency. In particular, we consider three geometric factors, i.e., overlap area, normalized central point distance and aspect ratio, which are crucial for measuring bounding box regression in object detection and instance segmentation. The three geometric factors are then incorporated into CIoU loss for better distinguishing difficult regression cases. The training of deep models using CIoU loss results in consistent AP and AR improvements in comparison to widely adopted $\ell_n$-norm loss and IoU-based loss. Furthermore, we propose Cluster-NMS, where NMS during inference is done by implicitly clustering detected boxes and usually requires less iterations. Cluster-NMS is very efficient due to its pure GPU implementation, and geometric factors can be incorporated to improve both AP and AR. In the experiments, CIoU loss and Cluster-NMS have been applied to state-of-the-art instance segmentation (e.g., YOLACT), and object detection (e.g., YOLO v3, SSD and Faster R-CNN) models. Taking YOLACT on MS COCO as an example, our method achieves performance gains as +1.7 AP and +6.2 $AR_{100}$ for object detection, and +0.9 AP and +3.5 $AR_{100}$ for instance segmentation, with 27.1 FPS on one NVIDIA GTX 1080Ti GPU. All the source code and trained models are available at https://github.com/Zzh-tju/CIoU.**

***Index Terms*—Instance segmentation, object detection, bounding box regression, non-maximum suppression.**

## I. Introduction

OBJECT detection and instance segmentation have received overwhelming research attention due to their practical applications in video surveillance, visual tracking, face detection and inverse synthetic aperture radar detection [1]–[5]. Since Deformable Part Model [6], bounding box regression has been widely adopted for localization in object detection. Driven by the success of deep learning, prosperous deep models based on bounding box regression have been studied, including one-stage [7]–[12], two-stage [13], [14], and multi-stage detectors [15], [16]. Instance segmentation is a more challenging task [17]–[19], where instance mask is further required for accurate segmentation of individuals. Recent state-of-the-art methods suggest to add an instance mask branch to existing object detection models, e.g., Mask R-CNN [20] based on Faster R-CNN [14] and YOLACT [19] based on RetinaNet [11]. In object detection and instance segmentation, dense boxes are usually regressed [9], [14], [19], [20]. As shown in Fig. 1, existing loss functions are limited in distinguishing difficult regression cases during training, and it takes tremendous cost to suppress redundant boxes during inference. In this paper, we suggest to handle this issue by enhancing geometric factors of bounding box regression into the learning and inference of deep models for object detection and instance segmentation.

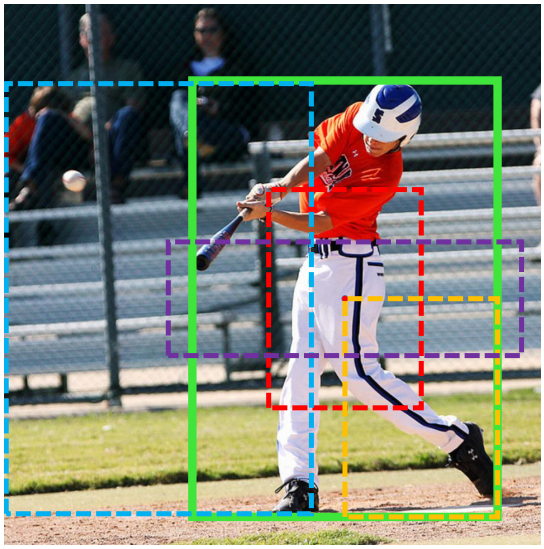


Fig. 1. Diversity of bounding box regression, where green box is the ground-truth box. First, albeit different ways of overlaps, these regression cases have the same $\ell_1$ loss and IoU loss. We propose CIoU loss by considering three geometric factors to distinguish them. Second, albeit NMS is a simple post-processing step, it is the bottleneck for suppressing redundant boxes in terms of both accuracy and inference efficiency. We then propose Cluster-NMS incorporating with geometric factors for improving AP and AR while maintaining high inference efficiency.

In training phase, a bounding box $\mathcal{B} = [x, y, w, h]^{\mathrm{T}}$ is forced to approach its ground-truth box $\mathcal{B}^{gt} = [x^{gt}, y^{gt}, w^{gt}, h^{gt}]^{\mathrm{T}}$ by minimizing loss function $\mathcal{L}$,

$$\min_{\Theta} \sum_{\mathcal{B}^{gt} \in \mathbb{B}^{gt}} \mathcal{L}(\mathcal{B}, \mathcal{B}^{gt} | \Theta), \tag{1}$$

Z. Zheng, P. Wang and R. Ye are with the School of Mathematics, Tianjin University, Tianjin, 300350, China. (Email: zh_zheng@tju.edu.cn, wang_ping@tju.edu.cn, ementon@tju.edu.cn).

D. Ren, W. Liu and Q. Hu are with the Tianjin Key Laboratory of Machine Learning, College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China. (Email: rendongweihit@gmail.com, lewiswestbrook95@gmail.com, huqinghua@tju.edu.cn)

W. Zuo is with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China. (Email: cswmzuo@gmail.com)

Corresponding author: Dongwei Ren

where $\mathbb{B}^{gt}$ is the set of ground-truth boxes, and $\Theta$ is the parameter of deep model for regression. A typical form of $\mathcal{L}$ is $\ell_n$-norm, e.g., Mean-Square Error (MSE) loss and Smooth-$\ell_1$ loss [21], which have been widely adopted in object detection [22], pedestrian detection [23], [24], text detection [25]–[27], 3D detection [28], [29], pose estimation [30], [31], and instance segmentation [17], [19]. However, recent works suggest that $\ell_n$-norm based loss functions are not consistent with the evaluation metric, i.e., Interaction over Union (IoU), and instead propose IoU-based loss functions [32]–[34]. For training state-of-the-art object detection models, e.g., YOLO v3 and Faster R-CNN, Generalized IoU (GIoU) loss achieves better precision than $\ell_n$-norm based losses. However, GIoU loss only tries to maximize overlap area of two boxes, and still performs limited due to only considering overlap areas (refer to the simulation experiments in Sec. III-A). As shown in Fig. 2, GIoU loss tends to increase the size of predicted box, while the predicted box moves towards the target box very slowly. Consequently, GIoU loss empirically needs more iterations to converge, especially for bounding boxes at horizontal and vertical orientations (see Fig. 4).

In testing phase, the inference of deep model is often efficient to predict dense boxes, which are left to Non-Maximum Suppression (NMS) for suppressing redundant boxes. NMS is an essential post-processing step in many detectors [7]–[9], [11], [13], [14], [19], [35], [36]. In original NMS, a box is suppressed only if it has overlap exceeding a threshold with the box having the highest classification score, which is likely to be not friendly to occlusion cases. Other NMS improvements, e.g., Soft-NMS [37] and Weighted-NMS [38], can contribute to better detection precision. However, these improved NMS methods are time-consuming, severely limiting their real-time inference. Some accelerated NMS methods [19], [39] have been developed for real-time inference, e.g., Fast NMS [19]. Unfortunately, Fast NMS yields performance drop due to that many boxes are likely to be over-suppressed.

In this paper, we propose to enhance geometric factors in both training and testing phases, where Complete-IoU (CIoU) loss aims to better distinguish difficult regression cases and Cluster-NMS can improve AP and AR without the sacrifice of inference time. As for CIoU loss, three geometric factors, i.e., overlap area, normalized central point distance and aspect ratio, are formulated as invariant to regression scale. Benefiting from complete geometric factors, CIoU loss can be deployed to improve average precision (AP) and average recall (AR) when training deep models in object detection and instance segmentation. From Fig. 2, CIoU loss converges much faster than GIoU loss, and the incorporation of geometric factors leads to much better match of two boxes.

We further propose Cluster-NMS, by which NMS can be done by implicitly clustering detected boxes and geometric factors can be easily incorporated, while maintaining high inference efficiency. First, in Cluster-NMS, redundant detected boxes can be suppressed by grouping them implicitly into clusters. Cluster-NMS usually requires less iterations, and its suppression operations can be purely implemented on GPU, benefiting from parallel acceleration. Cluster-NMS can guarantee exactly the same result with original NMS, while it is very efficient. Then, geometric factors, such as overlap-based score penalty, overlap-based weighted coordinates and normalized central point distance, can be easily assembled into Cluster-NMS. Benefiting from geometric factors, Cluster-NMS achieves significant gains in both AP and AR, while maintaining high inference efficiency. Moreover, Cluster-NMS can also run in batch-mode, further boosting the inference speed.

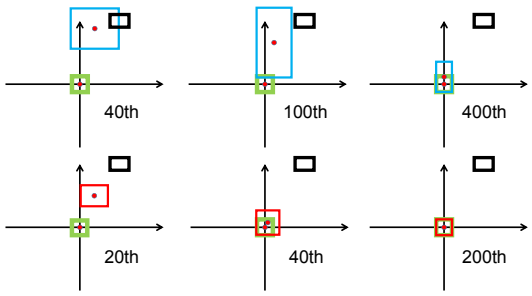



Fig. 2. Updating of predicted boxes after different iterations optimized by GIoU loss (first row) and CIoU loss (second row). Green and black denote target box and anchor box, respectively. Blue and red denote predicted boxes for GIoU loss and CIoU loss, respectively. GIoU loss only considers overlap area, and tends to increase the GIoU by enlarging the size of predicted box. Benefiting from all the three geometric factors, the minimization of normalized central point distance in CIoU loss gives rise to fast convergence and the consistency of overlap area and aspect ratio contributes to better match of two boxes.

In the experiments, CIoU loss and Cluster-NMS have been applied to several state-of-the-art instance segmentation (e.g., YOLACT) and object detection (e.g., YOLO v3, SSD and Faster R-CNN) models. Experimental results demonstrate that CIoU loss can lead to consistent gains in AP and AR against $\ell_n$-norm based and IoU-based losses for object detection and instance segmentation. Cluster-NMS contributes to notable gains in AP and AR, and guarantees real-time inference.

This paper is a substantial extension of our pioneer work [40]. In this work, CIoU loss is better analyzed and is further applied to instance segmentation. The newly proposed Cluster-NMS achieves much better performance and can guarantee real-time efficiency for object detection and instance segmentation. We summarize the contributions from three aspects:

- A Complete IoU loss, i.e., CIoU loss, is proposed by taking three geometric factors, i.e., overlap area, normalized central point distance and aspect ratio, into account, and results in consistent performance gains for training deep models of bounding box regression.
- We propose Cluster-NMS, in which geometric factors can be further exploited for improving AP and AR while maintaining high inference efficiency.
- CIoU loss and Cluster-NMS have been applied to several state-of-the-art instance segmentation (e.g., YOLACT) and object detection (e.g., YOLO v3, SSD and Faster R-CNN) models. Experimental results clearly validate the effectiveness and efficiency of our methods.

The remainder is organized as follows: Sec. II briefly reviews related works, Sec. III proposes CIoU loss by taking complete geometric factors into account, Sec. IV presents Cluster-NMS along with its variants by incorporating geomet-

ric factors, Sec. V gives experimental results and Sec. VI ends this paper with concluding remarks.

## II. Related Work

### A. Object Detection and Instance Segmentation

For a long time bounding box regression has been adopted as an essential component in many representative object detection frameworks [6]. In deep models for object detection, R-CNN series [14], [16], [20] adopt two or three bounding box regression modules to obtain higher location accuracy, while YOLO series [7], [8], [36] and SSD series [9], [10], [41] adopt one for faster inference speed. Recently, in RepPoints [42], a rectangular box is formed by predicting several points. FCOS [12] locates an object by predicting the distances from the sampling points to the top, bottom, left and right sides of the ground-truth box. PolarMask [18] predicts the length of $n$ rays from the sampling point to the edge of the object in $n$ directions to segment an instance. There are other detectors such as RRPN [25] and R$^2$CNN [43] adding rotation angle regression to detect arbitrary orientated objects for remote sensing detection and scene text detection. For instance segmentation, Mask R-CNN [20] adds an extra instance mask branch on Faster R-CNN, while the recent state-of-the-art YOLACT [19] does the same thing on RetinaNet [11]. To sum up, bounding box regression is one key component of state-of-the-art deep models for object detection and instance segmentation.

### B. Loss Function for Bounding Box Regression

Albeit the architectures of deep models have been well studied, loss function for bounding box regression also plays a critical role in object detection. While $\ell_n$-norm loss functions are usually adopted in bounding box regression, they are sensitive to varying scales. In YOLO v1 [36], square roots for $w$ and $h$ are adopted to mitigate this effect, while YOLO v3 [8] uses $2 - wh$. In Fast R-CNN, Huber loss is adopted to obtain more robust training. Meyer [44] suggested to connect Huber loss with the KL divergence of Laplace distributions, and further proposed a new loss function to eliminate the transition points between $\ell_1$-norm and $\ell_2$-norm in the Huber loss. Libra R-CNN [45] studies the imbalance issues and proposes Balanced-$\ell_1$ loss. In GHM [46], the authors proposed a gradient harmonizing mechanism for bounding box regression loss that rectifies the gradient contributions of samples. IoU loss is also used since Unitbox [32], which is invariant to the scale. To ameliorate the training stability, Bounded-IoU loss [33] introduces the upper bound of IoU. GIoU [34] loss is proposed to tackle the issues of gradient vanishing for non-overlapping cases, but still suffers from the problems of slow convergence and inaccurate regression. Nonetheless, geometric factors of bounding box regression are actually not fully exploited in existing loss functions. Therefore, we propose CIoU loss by taking three geometric factors into account for better training deep models of object detection and instance segmentation.

### C. Non-Maximum Suppression

NMS is a simple post-processing step in the pipelines of object detection and instance segmentation, but it is the key bottleneck for detection accuracy and inference efficiency. As for improving detection accuracy, Soft-NMS [37] penalizes the detection score of neighbors by a continuous function w.r.t. IoU, yielding softer and more robust suppression than original NMS. IoU-Net [47] introduces a new network branch to predict the localization confidence to guide NMS. Weighted-NMS [38] outputs weighted combination of the cluster based on their scores and IoU. Recently, Adaptive NMS [48] and Softer-NMS [49] are proposed to respectively study proper threshold and weighted average strategies. As for improving inference efficiency, boolean matrix [39] is adopted to represent IoU relationship of detected boxes, for facilitating GPU acceleration. A CUDA implementation of original NMS by Faster R-CNN [14] uses logic operations to check the boolean matrix line by line. Recently, Fast NMS [19] is proposed to improve inference efficiency, but it inevitably brings a drop of performance due to the over-suppression of boxes. In this work, we propose efficient Cluster-NMS, and geometric factors can be readily exploited to obtain significant improvements in both precision and recall.

## III. Complete-IoU Loss

For training deep models in object detection, IoU-based losses are suggested to be more consistent with IoU metric than $\ell_n$-norm losses [32]–[34]. The original IoU loss can be formulated as [34],

$$\mathcal{L}_{IoU} = 1 - IoU. \tag{2}$$

However, it fails in distinguishing the cases that two boxes do not overlap. Then, GIoU [34] loss is proposed,

$$\mathcal{L}_{GIoU} = 1 - IoU + \frac{|\mathcal{C} - \mathcal{B} \cup \mathcal{B}^{gt}|}{|\mathcal{C}|}, \tag{3}$$

where $\mathcal{C}$ is the smallest box covering $\mathcal{B}$ and $\mathcal{B}^{gt}$, and $|\mathcal{C}|$ is the area of box $\mathcal{C}$. Due to the introduction of penalty term in GIoU loss, the predicted box will move towards the target box in non-overlapping cases. GIoU loss has been applied to train state-of-the-art object detectors, e.g., YOLO v3 and Faster R-CNN, and achieves better precision than MSE loss and IoU loss.

### A. Analysis to IoU and GIoU Losses

To begin with, we analyze the limitations of original IoU loss and GIoU loss. However, it is very difficult to analyze the procedure of bounding box regression simply from the detection results, where the regression cases in uncontrolled benchmarks are often not comprehensive, e.g., different distances, different scales and different aspect ratios. Instead, we suggest conducting simulation experiments, where the regression cases should be comprehensively considered, and then the issues of a given loss function can be easily analyzed.

**Algorithm 1** Simulation Experiment

**Input:** Loss $\mathcal{L}$ is a continuous bounded function defined on $\mathbb{R}^4_+$.
$\mathbb{B} = \{\{\mathcal{B}_{n,s}\}_{s=1}^{S}\}_{n=1}^{N}$ is the set of anchor boxes at $N = 5,000$ uniformly scattered points within the circular region with center $(10, 10)$ and radius 3, and $S = 7\times7$ covers 7 scales and 7 aspect ratios of anchor boxes.
$\mathbb{B}^{gt} = \{\mathcal{B}_i^{gt}\}_{i=1}^{7}$ is the set of target boxes that are fixed at $(10, 10)$ with area 1, and have 7 aspect ratios.
**Output:** Regression error $\boldsymbol{E} \in \mathbb{R}^{T\times N}$

Initialize $\boldsymbol{E} = \mathbf{0}$ and maximum iteration $T$.
Do bounding box regression:
**for** $n = 1$ to $N$ **do**
  **for** $s = 1$ to $S$ **do**
    **for** $i = 1$ to $7$ **do**
      **for** $t = 1$ to $T$ **do**

$$\eta = \begin{cases} 0.1 & \text{if} \quad t <= 0.8T \\ 0.01 & \text{if} \quad 0.8T < t <= 0.9T \\ 0.001 & \text{if} \quad t > 0.9T \end{cases}$$

        $\nabla\mathcal{B}_{n,s}^{t-1}$ is gradient of $\mathcal{L}(\mathcal{B}_{n,s}^{t-1}, \mathcal{B}_i^{gt})$ w.r.t. $\mathcal{B}_{n,s}^{t-1}$
        $\mathcal{B}_{n,s}^{t} = \mathcal{B}_{n,s}^{t-1} + \eta(2 - IoU_{n,s}^{t-1})\nabla\mathcal{B}_{n,s}^{t-1}$
        $\boldsymbol{E}(t, n) = \boldsymbol{E}(t, n) + |\mathcal{B}_{n,s}^{t} - \mathcal{B}_i^{gt}|$
      **end for**
    **end for**
  **end for**
**end for**
**return** $\boldsymbol{E}$

*1) Simulation Experiment:* In the simulation experiments, we try to cover most of the relationships between bounding boxes by considering geometric factors including distance, scale and aspect ratio, as shown in Fig. 3(a). In particular, we choose 7 unit boxes (i.e., the area of each box is 1) with different aspect ratios (i.e., 1:4, 1:3, 1:2, 1:1, 2:1, 3:1 and 4:1) as target boxes. Without loss of generality, the central points of the 7 target boxes are fixed at $(10, 10)$. The anchor boxes are uniformly scattered at 5,000 points. (***i***) Distance: In the circular region centered at $(10, 10)$ with radius 3, 5,000 points are uniformly chosen to place anchor boxes with 7 scales and 7 aspect ratios. In these cases, overlapping and non-overlapping boxes are included. (***ii***) Scale: For each point, the areas of anchor boxes are set as 0.5, 0.67, 0.75, 1, 1.33, 1.5 and 2. (***iii***) Aspect ratio: For a given point and scale, 7 aspect ratios are adopted, i.e., following the same setting with target boxes (i.e., 1:4, 1:3, 1:2, 1:1, 2:1, 3:1 and 4:1). All the $5,000\times7\times7$ anchor boxes should be fitted to each target box. To sum up, there are totally $1,715,000 = 7 \times 7 \times 7 \times 5,000$ regression cases.

Then given a loss function $\mathcal{L}$, we can simulate the procedure of bounding box regression for each case using stochastic gradient descent algorithm. For predicted box $\mathcal{B}_i$, the current prediction can be obtained by

$$\mathcal{B}_i^t = \mathcal{B}_i^{t-1} + \eta(2 - IoU_i^{t-1})\nabla\mathcal{B}_i^{t-1}, \tag{4}$$

where $\mathcal{B}_i^t$ is the predicted box at iteration $t$, $\nabla\mathcal{B}_i^{t-1}$ denotes the gradient of loss $\mathcal{L}$ w.r.t. $\mathcal{B}_i^{t-1}$ at iteration $t-1$, and $\eta$ is the learning rate. It is worth noting that in our implementation, the gradient is multiplied by $2 - IoU_i^{t-1}$ to accelerate the convergence. The performance of bounding box regression is evaluated using $\ell_1$-norm. For each loss function, the simulation experiment is terminated when reaching iteration $T = 200$,

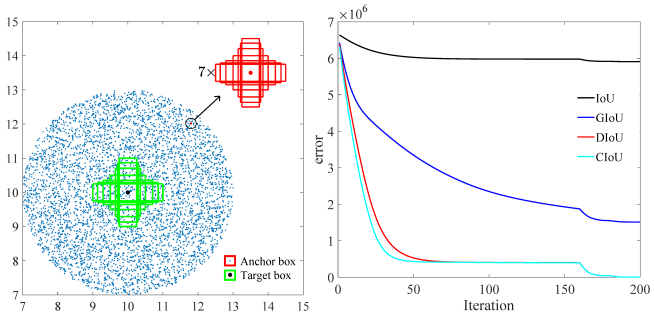



Fig. 3. Simulation experiments: (a) 1,715,000 regression cases are adopted by considering different distances, scales and aspect ratios, (b) regression error sum (i.e., $\sum_n \boldsymbol{E}(t, n)$) curves of different loss functions at iteration $t$.

and the error curves are shown in Fig. 3(b).

*2) Limitations of IoU and GIoU Losses:* In Fig. 4, we visualize the final regression errors at iteration $T$ for 5,000 scattered points. From Fig. 4(a), it is easy to see that IoU loss only works for the cases of overlapping with target boxes. The anchor boxes without overlap will not move due to that the gradient is always 0.

By adding a penalty term as Eqn. (3), GIoU loss can better relieve the issues of non-overlapping cases. From Fig. 4(b), GIoU loss significantly enlarges the basin, i.e., the area that GIoU works. But the cases with extreme aspect ratios are likely to still have large errors. This is because that the penalty term in GIoU loss is used to minimize $|C - A \cup B|$, but the area of $C - A \cup B$ is often small or 0 (when two boxes have inclusion relationships), and then GIoU almost degrades to IoU loss. GIoU loss would converge to good solution as long as running sufficient iterations with proper learning rates, but the convergence rate is indeed very slow. Geometrically speaking, from the regression steps as shown in Fig. 2, one can see that GIoU actually increases the predicted box size to overlap with target box, and then the IoU term will make the predicted box match with the target box, yielding a very slow convergence.

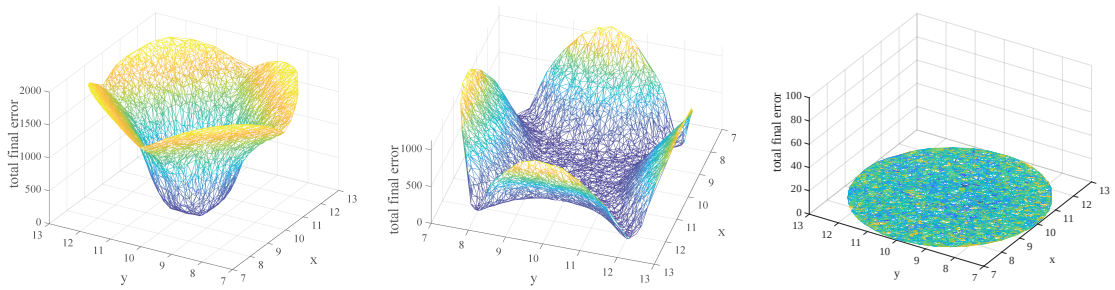
To sum up, IoU-based losses only aim to maximize the overlap area of two boxes. Original IoU loss converges to bad solutions for non-overlapping cases, while GIoU loss is with slow convergence especially for the boxes with extreme aspect ratios. And when incorporating into object detection or instance segmentation pipeline, both IoU and GIoU losses cannot guarantee the accuracy of regression. We in this paper suggest that a good loss function for bounding box regression should enhance more geometric factors besides overlap area.

### B. CIoU Loss

Considering the geometric factors for modeling regression relationships in Simulation Experiment, we suggest that a loss function should take three geometric factors into account, i.e., overlap area, distance and aspect ratio. Generally, a complete loss can be defined as,

$$\mathcal{L} = S(\mathcal{B}, \mathcal{B}^{gt}) + D(\mathcal{B}, \mathcal{B}^{gt}) + V(\mathcal{B}, \mathcal{B}^{gt}) \tag{5}$$

where $S$, $D$, $V$ denote the overlap area, distance and aspect ratio, respectively. One can see that IoU and GIoU losses only




(a) $\mathcal{L}_{IoU}$ (b) $\mathcal{L}_{GIoU}$ (c) $\mathcal{L}_{CIoU}$

Fig. 4. Visualization of regression errors of IoU, GIoU and CIoU losses at the final iteration $T$, i.e., $\boldsymbol{E}(T, n)$ for every coordinate $n$. We note that the basins in (a) and (b) correspond to good regression cases. One can see that IoU loss has large errors for non-overlapping cases, GIoU loss has large errors for horizontal and vertical cases, and our CIoU loss leads to very small regression errors everywhere.

consider the overlap area. In the complete loss, IoU is only a good choice for $S$,

$$S = 1 - IoU. \tag{6}$$

Similar to IoU, we want to make both $D$ and $V$ be also invariant to regression scale. In particular, we adopt normalized central point distance to measure the distance of two boxes,

$$D = \frac{\rho^2(\boldsymbol{p}, \boldsymbol{p}^{gt})}{c^2}, \tag{7}$$

where $\boldsymbol{p} = [x, y]^{\mathrm{T}}$ and $\boldsymbol{p}^{gt} = [x^{gt}, y^{gt}]^{\mathrm{T}}$ are the central points of boxes $\mathcal{B}$ and $\mathcal{B}^{gt}$, $c$ is the diagonal length of box $\mathcal{C}$, and $\rho$ is specified as Euclidean distance, as shown in Fig. 5. And

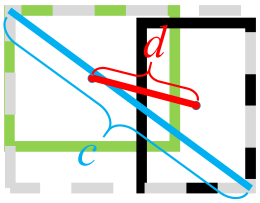



Fig. 5. Normalized central point distance. $c$ is the diagonal length of the smallest enclosing box covering two boxes, and $d = \rho(\boldsymbol{p}, \boldsymbol{p}^{gt})$ is the distance of central points of two boxes.

the consistency of aspect ratio is implemented as

$$V = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2. \tag{8}$$

Finally, we obtain the Complete-IoU (CIoU) loss,

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\boldsymbol{p}, \boldsymbol{p}^{gt})}{c^2} + \alpha V. \tag{9}$$

It is easy to see that $S$, $D$ and $V$ are invariant to regression scale and are normalized to $[0, 1]$. Here, we only introduce one trade-off parameter $\alpha$, which is defined as

$$\alpha = \begin{cases} 0, & \text{if } IoU < 0.5, \\ \frac{V}{(1-IoU)+V}, & \text{if } IoU \geq 0.5. \end{cases} \tag{10}$$

One can see that our CIoU loss will degrade to DIoU loss in our pioneer work [40] when $IoU < 0.5$. It is reasonable that when two boxes are not well matched, the consistency of aspect ratio is less important. And when $IoU \geq 0.5$, the consistency of aspect ratio becomes necessary.

The proposed CIoU loss inherits some properties from IoU and GIoU losses. (i) CIoU loss is still invariant to the scale of regression problem. (ii) Analogous to GIoU loss, CIoU loss can provide moving directions for bounding boxes when non-overlapping with target box. Furthermore, our CIoU loss has two merits over IoU and GIoU losses, which can be evaluated by simulation experiment. (i) As shown in Fig. 2 and Fig. 3, CIoU loss can rapidly minimize the distance of two boxes, and thus converges much faster than GIoU loss. (ii) For the cases with inclusion of two boxes, or with extreme aspect ratios, CIoU loss can make regression very fast, while GIoU loss has almost degraded to IoU loss, i.e., $|\mathcal{C} - \mathcal{B} \cup \mathcal{B}^{gt}| \to 0$.

## IV. Cluster-NMS

Most state-of-the-art object detection [8], [9], [14] and instance segmentation [19] adopt the strategy to place more anchor boxes to detect difficult and small objects for improving detection accuracy. Moreover, NMS for suppressing redundant boxes is facing tremendous pressure during inference.

Let $\boldsymbol{B} = [\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_N]^{\mathrm{T}}$ [1] is an $N \times 4$ matrix storing $N$ detected boxes. These boxes have been sorted according to the non-ascending classification scores, i.e., $s_1 \geq s_2 \geq \cdots \geq s_N$. Original NMS is to suppress redundant boxes by sequentially traversing $N$ boxes. Specifically, for the box $\mathcal{M}$ with the current highest score, original NMS can be formally defined as,

$$s_j = \begin{cases} s_j, & \text{if } IoU(\mathcal{M}, \mathcal{B}_j) < \varepsilon, \\ 0, & \text{if } IoU(\mathcal{M}, \mathcal{B}_j) \geq \varepsilon, \end{cases} \tag{11}$$

where $\varepsilon$ is a threshold. Original NMS is very time-consuming. And several improved NMS, e.g., Soft-NMS [37] and non-maximum weighted (Weighted-NMS) [38], can further improve the precision and recall, but they are much more inefficient. We propose Cluster-NMS, where NMS can be parallelly done on implicit clusters of detected boxes, usually requiring less iterations. Besides, Cluster-NMS can be purely implemented on GPU, and is much more efficient than original NMS. Then, we incorporate the geometric factors into Cluster-NMS for further improving both precision and recall, while maintaining high inference efficiency.

**Algorithm 2** Cluster-NMS

**Input:** $N$ detected boxes $\boldsymbol{B} = [\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_N]^{\mathrm{T}}$ with non-ascending sorting by classification score, i.e., $s_1 \geq \cdots \geq s_N$.
**Output:** $\boldsymbol{b} = \{b_i\}_{1\times N}, b_i \in \{0, 1\}$ encodes final detection result, where 1 denotes reservation and 0 denotes suppression.

Initialize $T = N$, $t = 1$ and $\boldsymbol{b}^0 = \mathbf{1}$
Compute IoU matrix $\boldsymbol{X} = \{x_{ij}\}_{N\times N}$ with $x_{ij} = IoU(\mathcal{B}_i, \mathcal{B}_j)$.
$\boldsymbol{X} = \texttt{triu}(\boldsymbol{X})$ ▷ Upper triangular matrix with $x_{ii} = 0, \forall i$
**while** $t \leq T$ **do**
  $\boldsymbol{A}^t = \texttt{diag}(\boldsymbol{b}^{t-1})$
  $\boldsymbol{C}^t = \boldsymbol{A}^t \times \boldsymbol{X}$
  $\boldsymbol{g} \leftarrow \max_j \boldsymbol{C}^t$ ▷ Find maximum for each column $j$
  $\boldsymbol{b}^t \leftarrow \texttt{find}(\boldsymbol{g} < \varepsilon)$ ▷ $\begin{cases} b_j = 1, & \text{if } g_j < \varepsilon \\ b_j = 0, & \text{if } g_j \geq \varepsilon \end{cases}$
  **if** $\boldsymbol{b}^t == \boldsymbol{b}^{t-1}$ **then**
    $t^* = t$, break
  **end if**
  $t = t + 1$
**end while**
**return** $\boldsymbol{b}^{t^*}$

### A. Cluster-NMS

We first compute the IoU matrix $\boldsymbol{X} = \{x_{ij}\}_{N\times N}$, where $x_{ij} = IoU(\mathcal{B}_i, \mathcal{B}_j)$. We note that $\boldsymbol{X}$ is a symmetric matrix, and we only need the upper triangular matrix of $\boldsymbol{X}$, i.e., $\boldsymbol{X} = \texttt{triu}(\boldsymbol{X})$ with $x_{ii} = 0, \forall i$. In Fast NMS [19], the suppression is directly performed on the matrix $\boldsymbol{X}$, i.e., a box $\mathcal{B}_j$ would be suppressed as long as $\exists\ x_{ij} > \varepsilon$. Fast NMS is indeed efficient, but it suppresses too many boxes. Considering $\exists \mathcal{B}_i$ has been suppressed, the box $\mathcal{B}_i$ should be excluded when making the rule for suppressing $\mathcal{B}_j$ even if $x_{ij} > \varepsilon$. But in Fast NMS, $\mathcal{B}_i$ is actually taken into account, thereby being likely to over-suppress more boxes.

Let $\boldsymbol{b} = \{b_i\}_{1\times N}, b_i \in \{0, 1\}$ be a binary vector to indicate the NMS result. We introduce an iterative strategy, where current suppressed boxes with $b_i = 0$ would not affect the results. Specifically, for iteration $t$, we have the NMS result $\boldsymbol{b}^{t-1}$, and introduce two matrices,

$$\begin{aligned} \boldsymbol{A}^t &= \texttt{diag}(\boldsymbol{b}^{t-1}), \\ \boldsymbol{C}^t &= \boldsymbol{A}^t \times \boldsymbol{X}. \end{aligned} \tag{12}$$

Then the suppression is performed on the matrix $\boldsymbol{C}$. The details can be found in Alg. 2. All these operations can be implemented on GPU, and thus Cluster-NMS is very efficient.

When $T = 1$, Cluster-NMS degrades to Fast NMS, and when $T = N$, the result of Cluster-NMS is totally same with original NMS. Due to the pure operation on matrix $\boldsymbol{A}$ and $\boldsymbol{X}$, the predicted boxes without overlaps are implicitly grouped into different clusters, and the suppression is performed in parallel between clusters. In the following, we first prove that Cluster-NMS with $T = N$ iterations can achieve the same suppression result with original NMS, and then discuss that Cluster-NMS usually requires less iterations.

**Proposition 1.** *Let $\boldsymbol{b}^T$ be the result of Cluster-NMS after $T$ iterations, $\boldsymbol{b}^T$ is also the final result of original NMS.*

[1]Actually, $\boldsymbol{B}$ is a tensor with size $C \times N \times 4$, which contains $C$ classes. Since different classes share the same NMS operation, we omit this channel for simplicity.

*Proof.* Let $\boldsymbol{X}_k$ be the square block matrix of $\boldsymbol{X}$ containing the first $k \times k$ partition, while let $\boldsymbol{X}_{N-k}$ be the square block matrix of $\boldsymbol{X}$ containing the last $(N-k)\times(N-k)$ partition. The matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ share the same definition. Let $\boldsymbol{b}_k^k$ be the subvector of $\boldsymbol{b}$ containing the first $k$ elements after $k$ iterations in Cluster-NMS. And it is straightforward that $\boldsymbol{A}_k = \texttt{diag}(\boldsymbol{b}_k^k)$. Besides, we binarize the upper triangular IoU matrix $\boldsymbol{X} = \{x_{ij}\}_{N\times N}$ by threshold $\varepsilon$,

$$\begin{cases} x_{ij} = 0, & \text{if } x_{ij} < \varepsilon, \\ x_{ij} = 1, & \text{if } x_{ij} \geq \varepsilon, \end{cases} \tag{13}$$

which does not affect the result by Cluster-NMS in Alg. 2, but makes the proof easier to understand.

(***i***) When $t = 1$, $\boldsymbol{b}_1^1 = [b_1]^{\mathrm{T}} = [1]^{\mathrm{T}}$ is same to the result by original NMS, since the first box with the largest score is kept definitely. When $t = 2$, we have

$$\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{X} = \begin{pmatrix} \boldsymbol{A}_2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_{N-2} \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_2 & \boldsymbol{X}_{other} \\ \mathbf{0} & \boldsymbol{X}_{N-2} \end{pmatrix}. \tag{14}$$

Since the result $\boldsymbol{b}_2^2$ is not affected by the last $N-k$ boxes, we only consider

$$\boldsymbol{C}_2 = \boldsymbol{A}_2 \times \boldsymbol{X}_2 = \begin{pmatrix} 0 & b_1 x_{1,2} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & x_{1,2} \\ 0 & 0 \end{pmatrix}, \tag{15}$$

where $x_{1,2}$ is a binary value. And thus $\boldsymbol{b}_2^2 = [b_1, b_2]^{\mathrm{T}}$, where $b_2 = \neg x_{1,2}$ is exactly the output of original NMS at iteration $t = 2$.

(***ii***) Then we assume that when $t = k$, $\boldsymbol{b}_k^k$ determined by $\boldsymbol{C}_k$ is same with the result of original NMS after iteration $t = k$. When $t = k + 1$, we have

$$\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{X} = \begin{pmatrix} \boldsymbol{A}_{k+1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_{N-k-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_{k+1} & \boldsymbol{X}_{other} \\ \mathbf{0} & \boldsymbol{X}_{N-k-1} \end{pmatrix}. \tag{16}$$

Analogously, we do not care these block matrices $\boldsymbol{A}_{N-k-1}, \boldsymbol{X}_{other}$ and $\boldsymbol{X}_{N-k-1}$, since they do not affect $\boldsymbol{C}_{k+1}$,

$$\begin{aligned} \boldsymbol{C}_{k+1} = \boldsymbol{A}_{k+1} \times \boldsymbol{X}_{k+1} &= \begin{pmatrix} \boldsymbol{A}_k & 0 \\ \mathbf{0} & b \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_k & \boldsymbol{X}_{:,k+1} \\ \mathbf{0} & 0 \end{pmatrix}, \\ &= \begin{pmatrix} \boldsymbol{C}_k & \boldsymbol{A}_k \times \boldsymbol{X}_{:,k+1} \\ \mathbf{0} & 0 \end{pmatrix}, \end{aligned} \tag{17}$$

where $\boldsymbol{X}_{:,k+1}$ is the $k$-th column of $\boldsymbol{X}_k$ by excluding the last 0. Then it is easy to see that $b_{k+1}$ can be determined by

$$b_{k+1} = \neg \max(\boldsymbol{A}_k \times \boldsymbol{X}_{:,k+1}), \tag{18}$$

which is the output of Original NMS at iteration $t = k + 1$. And thus $\boldsymbol{b}_{k+1}^{k+1} = [\boldsymbol{b}_k^k; b_{k+1}]$ is same with the result of original NMS after iteration $t = k + 1$.

Combining (***i***) and (***ii***), it can be deduced that $\boldsymbol{b}^T$ after $T$ iterations in Cluster-NMS is exactly the final result of original NMS. □

***Discussion***: Cluster-NMS usually requires less iterations. Let $\mathbb{B}^* = \{\mathcal{B}_{j_1}, \mathcal{B}_{j_2}, \cdots, \mathcal{B}_{j_M}\}$ be the largest cluster containing $M$ boxes, where a box $\mathcal{B}_j \in \mathbb{B}^*$ if and only if $\exists i \in \{j_1, j_2, \cdots, j_M\}, IoU(\mathcal{B}_i, \mathcal{B}_j) \geq \varepsilon$, and $IoU(\mathcal{B}_j, \mathcal{B}_u) < \varepsilon, \forall j \in \{j_1, j_2, \cdots, j_M\}$ and $\forall u \notin \{j_1, j_2, \cdots, j_M\}$. Thus for

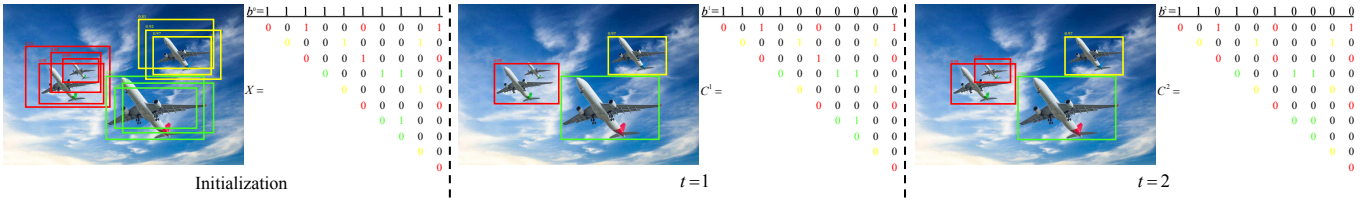



Fig. 6. An example of Cluster-NMS, where 10 detected boxes are implicitly grouped into 3 clusters. The IoU matrix $\boldsymbol{X}$ has been binarized by threshold $\varepsilon = 0.5$. When $t = 1$, Cluster-NMS is equivalent to Fast NMS [19], where the boxes are over-suppressed. Theoretically, Cluster-NMS will stop after at most 4 iterations, since the largest cluster (red boxes) has 4 boxes. But $\boldsymbol{b}$ after only 2 iterations is exactly the final result of original NMS, indicating that Cluster-NMS usually require less iterations.

the binarized IoU matrix $\boldsymbol{X}$, $x_{u,j} = 0, \forall j \in \{j_1, j_2, \cdots, j_M\}$ and $\forall u \notin \{j_1, j_2, \cdots, j_M\}$. That is to say $\mathbb{B}^*$ is actually processed without considering boxes in other clusters, and after $M$ iterations, $\boldsymbol{b}$ definitely will not change. Different clusters are processed in parallel by Cluster-NMS. We present an example in Fig. 6, where 10 detected boxes can be divided into 3 clusters and the largest cluster contains 4 boxes. Thus, the maximum iteration $t^*$ of Cluster-NMS in Alg. 2 is 4. But in Fig. 6, $\boldsymbol{b}$ after only 2 iterations is exactly the final result of original NMS. In practical, Cluster-NMS usually requires less iterations.

We also note that original NMS has been implemented as CUDA NMS in Faster R-CNN [14], which is recently included into TorchVision 0.3. TorchVision NMS is faster than Fast NMS and our Cluster-NMS (see Table VI), due to engineering accelerations. Our Cluster-NMS requires less iterations and can also be further accelerated by adopting these engineering tricks, e.g., logic operations on binarized $\boldsymbol{X}$ as in Proposition 1. But this is beyond the scope of this paper. Furthermore, Cluster-NMS can run in batch mode, which makes it faster than any other existing NMS including TorchVision NMS. Moreover, the main contribution of Cluster-NMS is that geometric factors can be easily incorporated into Cluster-NMS, while maintaining high efficiency.

### B. Incorporating Geometric Factors into Cluster-NMS

Geometric factors for measuring bounding box regression can be introduced into Cluster-NMS for improving precision and recall.

*1) Score Penalty Mechanism into Cluster-NMS:* Instead of the hard threshold in original NMS Eqn. (11), we introduce a Score Penalty Mechanism based on the overlap areas into Cluster-NMS, analogous to Soft-NMS. Specifically, in Cluster-NMS$_S$, we adopt the score penalty following "Gaussian" Soft-NMS [37], and the score $s_j$ is re-weighted as

$$s_j = s_j \prod_i e^{-\frac{(\boldsymbol{A}\times\boldsymbol{X})_{ij}^2}{\sigma}}, \tag{19}$$

where $\sigma = 0.2$ in this work. It is worth noting that our Cluster-NMS$_S$ is not completely same with Soft-NMS. In Cluster-NMS$_S$, $s_j$ is only penalized by those boxes with higher scores than $s_j$, since $\boldsymbol{A} \times \boldsymbol{X}$ is an upper triangular matrix.

*2) Normalized Central Point Distance into Cluster-NMS:* As suggested in our pioneer work [40], normalized central point distance can be included into NMS to benefit the cases with occlusions. By simply introducing the normalized central point distance $D$ in Eqn. (7) into IoU matrix $\boldsymbol{X}$, Cluster-NMS$_D$ is actually equivalent with DIoU-NMS in our pioneer work [40]. Moreover, we can incorporate the normalized central point distance into Cluster-NMS$_S$, forming Cluster-NMS$_{S+D}$. Specifically, the score $s_j$ is penalized as

$$s_j = s_j \prod_i \min\{e^{-\frac{(\boldsymbol{A}\times\boldsymbol{X})_{ij}^2}{\sigma}} + D^\beta, 1\}, \tag{20}$$

where $\beta = 0.6$ is a trade-off parameter for balancing the precision and recall (see Fig. 8).

*3) Weighted Coordinates into Cluster-NMS:* Weighted-NMS [38] is a variant of original NMS. Instead of deleting redundant boxes, Weighted-NMS creates new box by merging box coordinates according to the weighted combination of the boxes based on their scores and overlap ares. The formulation of Weighted-NMS is as follow,

$$\mathcal{B} = \frac{1}{\sum_j w_j} \sum_{\mathcal{B}_j \in \Lambda} w_j \mathcal{B}_j, \tag{21}$$

where $\Lambda = \{\mathcal{B}_j \mid x_{ij} \geq \varepsilon, \forall i\}$ is a set of boxes, weight $w_j = s_j IoU(\mathcal{B}, \mathcal{B}_j)$, and $\mathcal{B}$ is the newly created box. However, Weighted-NMS is very inefficient because of the sequential operations on every box.

Analogously, such weighted strategy can be included into our Cluster-NMS. In particular, given the matrix $\boldsymbol{C}$, we multiply its every column using the score vector $\boldsymbol{s} = [s_1, s_2, \cdots, s_N]^{\mathrm{T}}$ in the entry-by-entry manner, resulting in $\boldsymbol{C}'$ to contain both the classification score and IoU. Then for the $N$ detected boxes $\boldsymbol{B} = [\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_N]^{\mathrm{T}}$, their coordinates can be updated by

$$\boldsymbol{B} = \frac{\boldsymbol{C}' \times \boldsymbol{B}}{\sum_j \boldsymbol{C}'}. \tag{22}$$

The Cluster-NMS$_W$ shares the same output form with Cluster-NMS, but the coordinates of boxes have been updated. Moreover, the normalized central point distance can be easily assembled into Cluster-NMS$_W$, resulting in Cluster-NMS$_{W+D}$, where the IoU matrix $\boldsymbol{X}$ is computed by considering both overlap area and distance as DIoU-NMS [40]. Cluster-NMS$_W$ has the same result with Weighted-NMS as well as 6.1 times faster efficiency, and Cluster-NMS$_{W+D}$ contributes to consistent improvements in both AP and AR (see Table VI).

### C. Batch-mode Weighted Cluster-NMS

Batch-mode inference is a common strategy in video detection to improve the inference speed. We further introduce a batch-mode Weighted Cluster-NMS, which means it can run across multiple images. We note that TorchVision NMS has the fastest speed when processing on a single image, but it fails to run in batch-mode. When processing a batch containing $P$ images, TorchVision NMS needs a loop with $P$ times to process images one by one. In contrast, Cluster-NMS can run in batch-mode, i.e., processing $P$ images at one shot. Considering a batch containing $P$ images, the output of inference model is with the dimension $[P, N, 5]$, where $N$ denotes the number of boxes, and $5$ denotes the box coordinates $x, y, w, h$ and classification score $s$. Then we filter out the low score boxes by a threshold, e.g., 0.001. In order to run in batch-mode, we introduce an integer $U$, which denotes the maximum number of processing boxes. Finally, batch-mode Weighted Cluster-NMS processes on a tensor with shape of $[P, U, 5]$. It is easy to see that batch-mode Weighted Cluster-NMS is a generalization of Weighted Cluster-NMS. When $P = 1$, it degrades to Weighted Cluster-NMS. And when $P > 1$, the only change of batch-mode Weighted Cluster-NMS is that all operations in Alg. 2 have risen by one dimension. In practical, we suggest setting $U = 1,500$ when test-time augmentation (TTA) is turned on, and $U = 1,000$ when test-time augmentation (TTA) is turned off for balancing inference speed and accuracy.

## V. Experimental Results

In this section, we evaluate our proposed CIoU loss and Cluster-NMS for state-of-the-art instance segmentation YOLACT [19] and object detection YOLO v3 [8], SSD [9] and Faster R-CNN [14] on popular benchmark datasets, e.g., PASCAL VOC [50] and MS COCO [51]. CIoU loss is implemented in C/C++ and Pytorch, and Cluster-NMS is implemented in Pytorch. For the threshold $\varepsilon$, we adopt their default settings. The source code and trained models have been made publicly available.

### A. Instance Segmentation

YOLACT [19] is the most recent real-time instance segmentation method, in which Smooth-$\ell_1$ loss is adopted for training, and Fast NMS is used for real-time inference. To make a fair comparison, we train two YOLACT[2] (ResNet-101-FPN) models using Smooth-$\ell_1$ loss and CIoU loss, respectively. The training is carried out on NVIDIA GTX 1080Ti GPU with batchsize 4 per GPU. The training dataset is MS COCO 2017 train set, and the testing dataset is MS COCO 2017 validation set. The evaluation metrics include AP, AR, inference time (ms) and FPS. The details of different settings of AP and AR can be found in [9].

Tables I and II report the results of object detection and instance segmentation of two YOLACT models. By adopting the same NMS strategy, i.e., Fast NMS, one can see that CIoU loss can make a consistent improvements in AP and AR evaluation metrics for both object detection and instance segmentation, validating the contribution of geometric factors in CIoU loss. Then on the YOLACT model trained using our CIoU loss, we evaluate the effectiveness of Cluster-NMS by assembling different geometric factors. One can see that: (i) In comparison to original NMS, Fast NMS is efficient but yields notable drops in AP and AR, while our Cluster-NMS can guarantee exactly the same results with original NMS and its efficiency is comparable with Fast NMS. (ii) By enhancing score penalty based on overlap areas, Cluster-NMS$_S$ achieves notable gains in both AP and AR. Especially, for large objects, Cluster-NMS$_S$ performs much better in AP$_L$ and AR$_L$. The hard threshold strategy in original NMS is very likely to treat large objects with occlusion as redundancy, while our Cluster-NMS$_S$ is friendly to large objects. (iii) By further incorporating distance, Cluster-NMS$_{S+D}$ achieves higher AR metrics, albeit its AP metrics are only on par with Cluster-NMS$_S$. From Fig. 8, one can see that distance is a crucial factor for balancing precision and recall, and we choose $\beta = 0.6$ in our experiments. (iv) Incorporating geometric factors into Cluster-NMS takes only a little more inference time, and ~28 FPS on one GTX 1080Ti GPU can guarantee real-time inference. To sum up, our Cluster-NMS with geometric factors contributes to significant performance gains, while maintaining high inference efficiency.

One may notice that our re-trained YOLACT model using Smooth-$\ell_1$ loss is a little inferior to the results reported in their original paper [19]. This is because batchsize in [19] is set as 8, which causes out of memory on our GTX 1080Ti GPU. Then, we also evaluate Cluster-NMS on their released models trained using Smooth-$\ell_1$ loss. From Tables III and IV, one can draw the consistent conclusion that Cluster-NMS with geometric factors contributes to AP and AR improvements. Finally, we present the results of detection and segmentation in Fig. 7, from which one can easily find the more accurate detected boxes and segmentation masks by our CIoU loss and Cluster-NMS$_{S+D}$.

### B. Object Detection

For object detection, YOLO v3, SSD and Faster R-CNN are adopted for evaluation.

*1) YOLO v3 [8]:* First, we evaluate CIoU loss in comparison to MSE loss, IoU loss and GIoU loss on PASCAL VOC 2007 test set [50]. YOLO v3 is trained on PASCAL VOC 07+12 (the union of VOC 2007 trainval and VOC 2012 trainval). The backbone network is Darknet608. We follow exactly the GDarknet[3] training protocol released from [34]. Original NMS is adopted during inference. The performance for each loss has been reported in Table V. Besides AP metrics based on IoU, we also report the evaluation results using AP metrics based on GIoU. As shown in Table V, GIoU as a generalized version of IoU indeed achieves a certain degree of performance improvement. DIoU loss only includes distance in our pioneer work [40], and can improve the performance with gains of 3.29% AP and 6.02% AP75 using IoU as evaluation metric. CIoU loss takes the three

[2] https://github.com/dbolya/yolact

[3] https://github.com/generalized-iou/g-darknet

TABLE I
OBJECT DETECTION RESULTS OF YOLACT [19]. THE MODELS ARE RE-TRAINED USING SMOOTH-$\ell_1$ LOSS AND CIOU LOSS BY US, AND THE RESULTS ARE REPORTED ON MS COCO 2017 VALIDATION SET [51].

| Method | Loss | NMS Strategy | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLACT-550 [19] R-101-FPN | Smooth-$\ell_1$ | Fast NMS | **30.6** | **32.7** | 31.5 | 51.8 | 33.2 | 14.3 | 33.7 | 46.5 | 28.3 | 41.5 | 43.0 | 23.4 | 46.0 | 60.5 |
| | $\mathcal{L}_{CIoU}$ | Fast NMS | **30.6** | **32.7** | 32.1 | 51.6 | 33.9 | 14.2 | 34.0 | 48.3 | **28.7** | 41.7 | 43.0 | 22.1 | 45.4 | 61.9 |
| | | Original NMS | 11.6 | 86.6 | 32.5 | 52.5 | 34.1 | 14.6 | 34.7 | 48.7 | 28.6 | 42.9 | 45.1 | 23.3 | 48.1 | 64.7 |
| | | Cluster-NMS | 28.8 | 34.7 | 32.5 | 52.5 | 34.1 | 14.6 | 34.7 | 48.6 | 28.6 | 42.8 | 45.2 | 23.3 | 48.1 | 64.6 |
| | | Cluster-$NMS_S$ | 28.6 | 35.0 | 33.1 | **52.9** | **35.2** | **14.9** | 35.2 | 49.7 | **28.7** | **45.3** | 48.8 | **25.3** | 51.7 | 70.4 |
| | | Cluster-$NMS_{S+D}$ | 27.1 | 36.9 | **33.2** | 52.8 | **35.2** | **14.9** | **35.3** | **49.7** | **28.7** | 45.2 | **49.2** | 25.2 | **52.4** | **71.1** |

TABLE II
INSTANCE SEGMENTATION RESULTS OF YOLACT [19]. THE MODELS ARE RE-TRAINED USING SMOOTH-$\ell_1$ LOSS AND CIOU LOSS BY US, AND THE RESULTS ARE REPORTED ON MS COCO 2017 VALIDATION SET [51].

| Method | Loss | NMS Strategy | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLACT-550 [19] R-101-FPN | Smooth-$\ell_1$ | Fast NMS | **30.6** | **32.7** | 29.1 | 47.4 | 30.5 | 9.4 | 32.0 | 48.5 | 27.5 | 39.2 | 40.3 | 18.8 | 44.7 | 59.8 |
| | $\mathcal{L}_{CIoU}$ | Fast NMS | **30.6** | **32.7** | 29.6 | 48.1 | 30.9 | 9.4 | 32.0 | 49.7 | 27.6 | 39.3 | 40.3 | 18.0 | 44.0 | 60.8 |
| | | Original NMS | 11.5 | 86.6 | 29.7 | 48.3 | 31.0 | 9.4 | 32.2 | 49.8 | 27.5 | 40.1 | 41.7 | 18.7 | 45.8 | 62.8 |
| | | Cluster-NMS | 28.8 | 34.7 | 29.7 | 48.3 | 31.0 | 9.4 | 32.2 | 49.7 | 27.5 | 40.1 | 41.7 | 18.8 | 45.8 | 62.8 |
| | | Cluster-$NMS_S$ | 28.6 | 35.0 | **30.3** | **49.1** | **31.7** | **9.7** | **33.0** | **50.8** | **27.7** | **41.4** | 43.6 | **19.7** | 47.7 | 65.9 |
| | | Cluster-$NMS_{S+D}$ | 27.1 | 36.9 | 30.2 | 48.9 | **31.7** | 9.6 | 32.8 | 50.7 | 27.6 | 41.3 | **43.8** | 19.5 | **47.8** | **66.4** |

TABLE III
OBJECT DETECTION RESULTS OF YOLACT [19]. THE MODEL IS BORROWED FROM THE ORIGINAL PAPER [19], AND THE RESULTS ARE REPORTED ON MS COCO 2017 VALIDATION SET [51].

| Method | Backbone | NMS Strategy | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLACT-550 [19] | R-101-FPN | Fast NMS | **30.6** | **32.7** | 32.5 | 53.0 | 34.6 | 15.2 | 34.9 | 47.9 | 28.8 | 42.3 | 43.9 | 23.6 | 47.1 | 61.6 |
| | | Original NMS | 11.9 | 83.8 | 32.9 | 53.7 | 34.8 | 15.6 | 35.5 | 48.2 | 28.7 | 43.3 | 45.8 | 24.9 | 49.3 | 63.7 |
| | | Cluster-NMS | 29.2 | 34.2 | 32.9 | 53.7 | 34.8 | 15.6 | 35.5 | 48.2 | 28.7 | 43.3 | 45.9 | 25.0 | 49.3 | 63.7 |
| | | Cluster-$NMS_S$ | 28.8 | 34.7 | **33.5** | **54.0** | **35.9** | 15.7 | 36.0 | **49.2** | **28.8** | **45.7** | 49.7 | 26.7 | 53.1 | 70.0 |
| | | Cluster-$NMS_{S+D}$ | 27.5 | 36.4 | **33.5** | 53.9 | **35.9** | **15.8** | **36.1** | **49.2** | **28.8** | **45.7** | **50.2** | **26.9** | **53.6** | **71.2** |

TABLE IV
INSTANCE SEGMENTATION RESULTS OF YOLACT [19]. THE MODEL IS BORROWED FROM THE ORIGINAL PAPER [19], AND THE RESULTS ARE REPORTED ON MS COCO 2017 VALIDATION SET [51].

| Method | Backbone | NMS Strategy | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLACT-550 [19] | R-101-FPN | Fast NMS | **30.6** | **32.7** | 29.8 | 48.3 | 31.3 | 10.1 | 32.2 | 50.1 | 27.8 | 39.6 | 40.8 | 18.9 | 44.8 | 61.0 |
| | | Original NMS | 11.9 | 83.8 | 29.9 | 48.4 | 31.4 | 10.0 | 32.3 | 50.3 | 27.7 | 40.4 | 42.1 | 19.5 | 46.4 | 62.8 |
| | | Cluster-NMS | 29.2 | 34.2 | 29.9 | 48.4 | 31.4 | 10.0 | 32.3 | 50.3 | 27.7 | 40.4 | 42.1 | 19.5 | 46.4 | 62.8 |
| | | Cluster-$NMS_S$ | 28.8 | 34.7 | **30.5** | **49.3** | **32.1** | **10.3** | **33.1** | **51.2** | **27.8** | **41.8** | 44.1 | 20.4 | 48.3 | 66.3 |
| | | Cluster-$NMS_{S+D}$ | 27.5 | 36.4 | 30.4 | 49.1 | 32.0 | 10.2 | 32.9 | 51.1 | **27.8** | 41.7 | **44.3** | **20.5** | **48.5** | **66.8** |

important geometric factors into account, which brings an amazing performance gains, i.e., 5.67% AP and 8.95% AP75. From Fig. 9, one can see that detected boxes by CIoU loss are more accurate than those by GIoU loss. Also in terms of GIoU metric, we can draw the same conclusion, validating the effectiveness of CIoU loss.

Since YOLO v3 with GDarknet is implemented using C/C++, it is not suitable for evaluating Cluster-NMS. Then we evaluate Cluster-NMS on a pre-trained YOLO v3 model in Pytorch, where the model YOLOv3-spp-Ultralytics-608[4] achieves much better performance than the original paper of YOLO v3 [8] on MS COCO 2014 validation set. Table VI reports the comparison of different NMS methods. We note that original NMS (TorchVision) is the most efficient due to CUDA implementation and engineering acceleration in TorchVision. Our Cluster-NMS is only a little slower than original NMS (TorchVision) when processing images one by one. By merging coordinates based on overlap areas

[4]https://github.com/ultralytics/yolov3

TABLE V
QUANTITATIVE COMPARISON OF YOLOV3 [8] TRAINED USING DIFFERENT LOSSES. THE RESULTS ARE REPORTED ON THE TEST SET OF PASCAL VOC 2007.

| Loss / Evaluation | AP | | AP75 | |
|---|---|---|---|---|
| | IoU | GIoU | IoU | GIoU |
| MSE | 46.1 | 45.1 | 48.6 | 46.7 |
| $\mathcal{L}_{IoU}$ | 46.6 | 45.8 | 49.8 | 48.8 |
| $\mathcal{L}_{GIoU}$ | 47.7 | 46.9 | 52.2 | 51.1 |
| $\mathcal{L}_{DIoU}$ | 48.1 | 47.4 | 52.8 | 51.9 |
| $\mathcal{L}_{CIoU}$ | **49.2** | **48.4** | **54.3** | **52.9** |

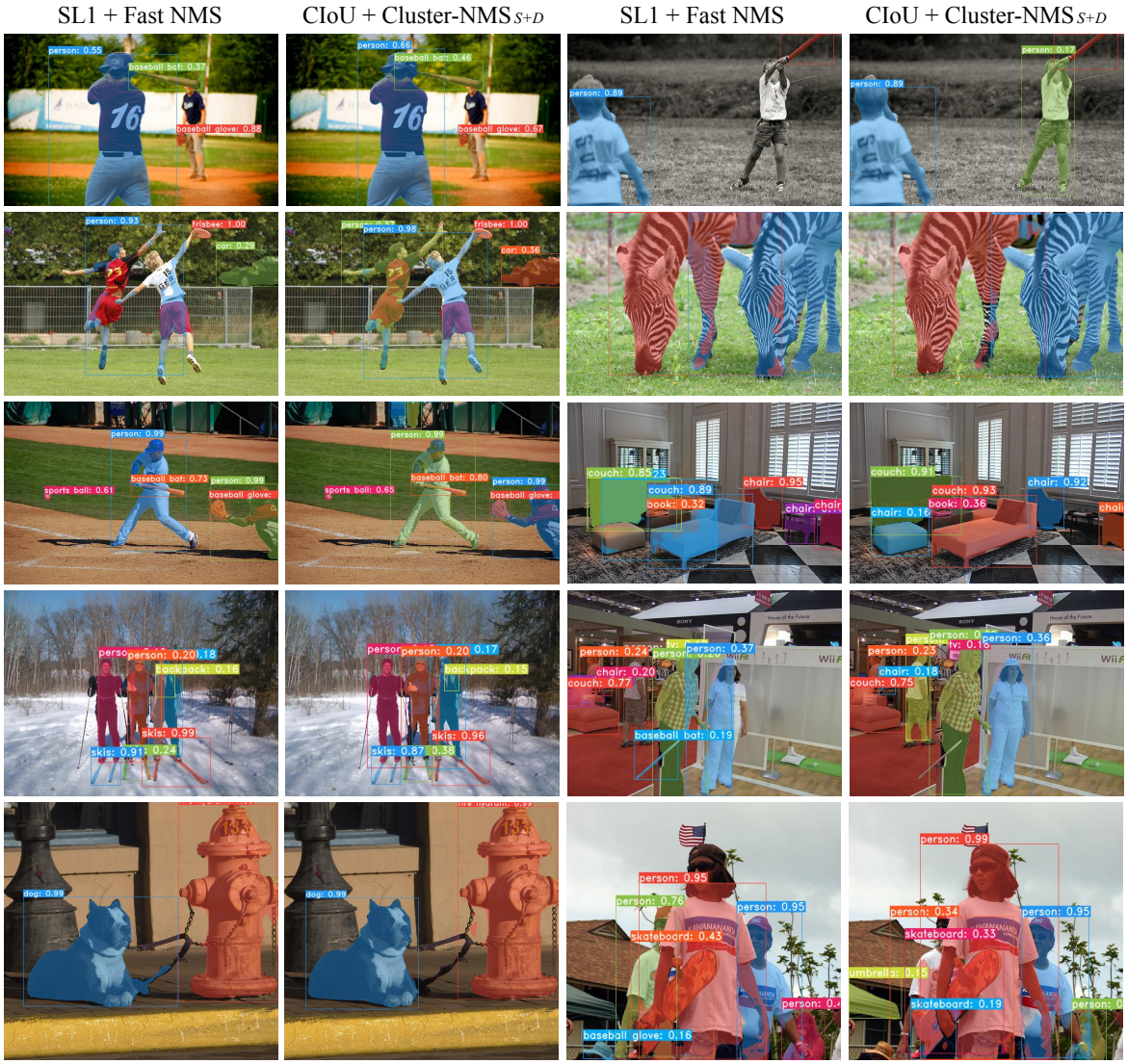



Fig. 7. Detection and segmentation results of YOLACT [19] on MS COCO 2017.

TABLE VI
COMPARISON OF DIFFERENT NMS METHODS ON PRE-TRAINED YOLO V3 MODEL. THE RESULTS ARE REPORTED ON MS COCO 2014 VALIDATION SET.

| Method | NMS Strategy | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLO v3 [8] | Fast NMS | 85.5 | 11.7 | 42.2 | 62.2 | 45.1 | 24.4 | 47.2 | 53.4 | 34.8 | 56.0 | 60.1 | 43.0 | 65.1 | 73.4 |
| | Original NMS | 14.6 | 68.5 | 42.6 | 62.5 | 45.8 | 24.7 | 47.8 | 53.7 | 34.8 | 57.2 | 62.5 | 45.0 | 67.8 | 75.6 |
| | Original NMS (TorchVision) | **95.2** | **10.5** | 42.6 | 62.5 | 45.8 | 24.7 | 47.8 | 53.7 | 34.8 | 57.2 | 62.5 | 45.0 | 67.9 | 75.6 |
| | Weighted-NMS | 11.2 | 89.6 | 42.9 | **62.7** | 46.4 | 25.2 | 48.2 | 53.9 | **35.0** | 57.4 | 62.7 | 45.4 | 68.3 | 75.6 |
| | Cluster-NMS | 82.6 | 12.1 | 42.6 | 62.5 | 45.8 | 24.7 | 47.8 | 53.7 | 34.8 | 57.2 | 62.5 | 45.0 | 67.9 | 75.6 |
| | Cluster-$NMS_W$ | 68.0 | 14.7 | 42.9 | **62.7** | 46.4 | 25.2 | 48.2 | 53.9 | **35.0** | 57.4 | 62.7 | 45.4 | 68.3 | 75.6 |
| | Cluster-$NMS_{W+D}$ | 64.5 | 15.5 | **43.1** | 62.4 | **46.8** | **25.3** | **48.3** | **54.1** | **35.0** | **58.0** | **63.7** | **46.4** | **69.3** | **76.7** |

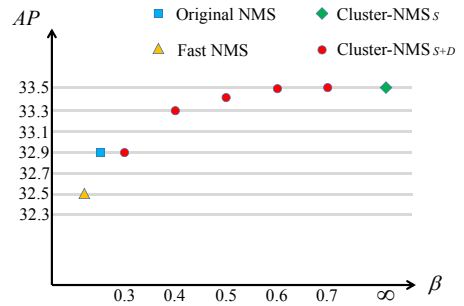
and scores, Weighted-NMS achieves higher AP and AR than original NMS, but it dramatically lowers the inference speed, making it infeasible for real-time application. Our Cluster-$NMS_W$ can guarantee the same results with Weighted-NMS, but is much more efficient. Our Cluster-$NMS_{W+D}$ contributes to further improvements than Cluster-$NMS_W$, especially in terms of AR. Actually, Cluster-NMS can be further accelerated by logic operations on binarized IoU matrix, but it makes infeasible for incorporating other geometric factors. Also our Cluster-NMS with geometric factors can still guarantee real-time inference.

Next, we further evaluate our batch-mode Weighted Cluster-NMS on two pre-trained YOLO v5 models[5] in Pytorch. Table VII reports the result for original NMS (TorchVision), Merge

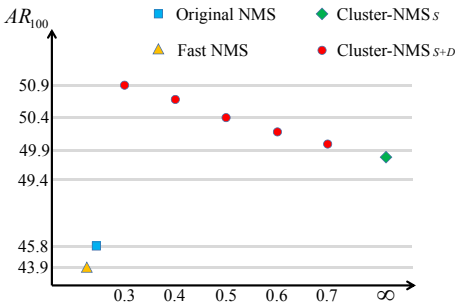
[5]https://github.com/ultralytics/yolov5

TABLE VII
COMPARISON OF DIFFERENT NMS METHODS ON PRE-TRAINED YOLOV5 [52] MODELS. THE RESULTS ARE REPORTED ON MS COCO 2017 VALIDATION SET. TTA DENOTES TEST-TIME AUGMENTATION. BATCH SIZE IS 32.

| Method | NMS Strategy | TTA | $U$ | FPS | Time | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLOv5s [52] | Original NMS (TorchVision) | ✓ | - | 109 | 9.2 | 38.0 | 56.5 | 41.2 | 20.9 | 42.6 | 51.7 |
| | Merge NMS (TorchVision) | ✓ | - | 105 | 9.5 | 38.1 | 56.5 | 41.4 | 21.0 | 42.7 | 51.8 |
| | Batch-mode Weighted Cluster-NMS | ✓ | 1,000 | **185** | **5.4** | 38.0 | 55.7 | 41.6 | 20.5 | 42.8 | 51.9 |
| | Batch-mode Weighted Cluster-NMS | ✓ | 1,500 | 152 | 6.6 | 38.3 | 56.2 | 41.8 | 21.1 | 43.0 | 52.0 |
| | Batch-mode Weighted Cluster-NMS | ✓ | 2,000 | 125 | 8.0 | 38.4 | 56.4 | 41.9 | 21.3 | 43.1 | 52.1 |
| | Original NMS (TorchVision) | | - | **303** | **3.3** | 36.9 | 56.2 | 40.0 | 21.0 | 42.1 | 47.4 |
| | Merge NMS (TorchVision) | | - | 285 | 3.5 | 37.1 | 56.2 | 40.3 | 21.1 | 42.2 | 47.6 |
| | Batch-mode Weighted Cluster-NMS | | 1,000 | 285 | 3.5 | 37.0 | 56.0 | 40.3 | 21.1 | 42.2 | 47.5 |
| YOLOv5m [52] | Original NMS (TorchVision) | ✓ | - | 101 | 9.9 | 45.1 | 63.2 | 49.0 | 27.0 | 50.2 | 60.5 |
| | Merge NMS (TorchVision) | ✓ | - | 98 | 10.2 | 45.2 | 63.3 | 49.1 | 27.0 | 50.3 | 60.5 |
| | Batch-mode Weighted Cluster-NMS | ✓ | 1,000 | **115** | **8.7** | 44.6 | 62.3 | 49.1 | 26.0 | 50.0 | 60.4 |
| | Batch-mode Weighted Cluster-NMS | ✓ | 1,500 | 103 | 9.7 | 45.2 | 62.9 | 49.4 | 26.8 | 50.4 | 60.4 |
| | Original NMS (TorchVision) | | - | **238** | **4.2** | 44.3 | 63.2 | 48.2 | 27.4 | 50.0 | 56.4 |
| | Merge NMS (TorchVision) | | - | 213 | 4.7 | 44.4 | 63.2 | 48.6 | 27.6 | 50.2 | 56.4 |
| | Batch-mode Weighted Cluster-NMS | | 1,000 | 213 | 4.7 | 44.3 | 62.9 | 48.5 | 27.4 | 50.1 | 56.4 |




(a) Average Precision with different $\beta$ values




(b) Average Recall with different $\beta$ values

Fig. 8. Balancing precision and recall by different values of $\beta$ in Cluster-$NMS_{S+D}$. The results are from YOLACT model on MS COCO 2017 validation set.

NMS (TorchVision) and our batch-mode Weighted Cluster-NMS. One can see that when batch size is set to 32 and TTA is turned on, original NMS (TorchVision) and Merge NMS (TorchVision) are slower than our batch-mode Weighted Cluster-NMS. When TTA is turned off, our batch mode Weighted Cluster-NMS has comparable speed with Merge NMS (TorchVision). It should be noticed that the time cost of our batch-mode Weighted Cluster-NMS is only related to the integer $U$ when batch size is fixed, while the time cost of TorchVision NMS is related to the number of boxes output by the model. Therefore, when TTA is turned on, e.g., multi-scale testing or mirror left-right flip, the regression model will predict much more boxes, dramatically slowing down inference speed of TorchVision NMS. On the contrary, our batch-mode Weighted Cluster-NMS is free from time-consuming growth when adopting TTA for obtaining better performance.

TABLE VIII
QUANTITATIVE COMPARISON OF SSD [9] FOR EVALUATING DIFFERENT LOSS FUNCTIONS AND NMS METHODS. THE RESULTS ARE REPORTED ON THE TEST SET OF PASCAL VOC 2007.

| Loss / Evaluation | AP | $AP_{75}$ |
|---|---|---|
| $\mathcal{L}_{IoU}$ | 51.0 | 54.7 |
| $\mathcal{L}_{GIoU}$ | 51.1 | 55.4 |
| $\mathcal{L}_{DIoU}$ | 51.3 | 55.7 |
| $\mathcal{L}_{CIoU}$ | **51.5** | **56.4** |

(a) Comparison of different loss functions.

| NMS Strategy | FPS | Time | AP | $AP_{75}$ |
|---|---|---|---|---|
| Fast NMS | **28.8** | **34.7** | 50.7 | 56.2 |
| Original NMS | 17.8 | 56.1 | 51.5 | 56.4 |
| Cluster-NMS | 28.0 | 35.7 | 51.5 | 56.4 |
| Cluster-$NMS_W$ | 26.8 | 37.3 | 51.9 | 56.3 |
| Cluster-$NMS_{W+D}$ | 26.5 | 37.8 | **52.4** | **57.0** |

(b) Comparison of NMS methods on the model trained by $\mathcal{L}_{CIoU}$.

*2) SSD [9]:* We use another popular one-stage method SSD to further conduct evaluation experiments. The latest PyTorch implementation of SSD[6] is adopted. Both the training set and testing set share the same setting with YOLO v3 on PASCAL VOC. The backbone network is ResNet-50-FPN. And then we train the models using IoU, GIoU, DIoU and CIoU losses. Table VIII gives the quantitative comparison, in which AP metrics based on IoU and evaluation of NMS methods are reported. As for loss function, we can see the consistent improvements of CIoU loss against IoU, GIoU and DIoU losses. As for NMS, Cluster-$NMS_{W+D}$ leads to significant improvements in AP metrics, and its efficiency is still well maintained.

[6]https://github.com/JaryHuang/awesome_SSD_FPN_GIoU

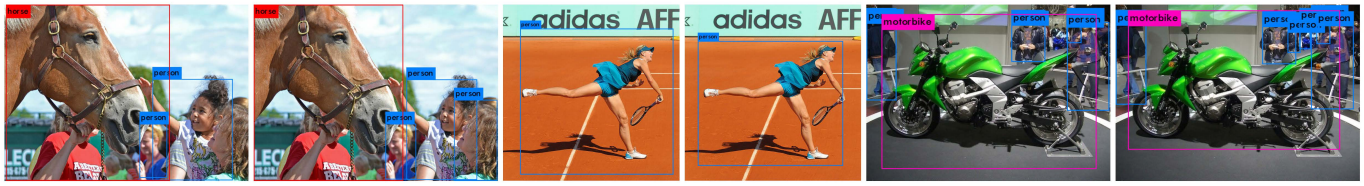



Fig. 9. Detection examples from YOLO v3 [8] trained on PASCAL VOC 07+12.

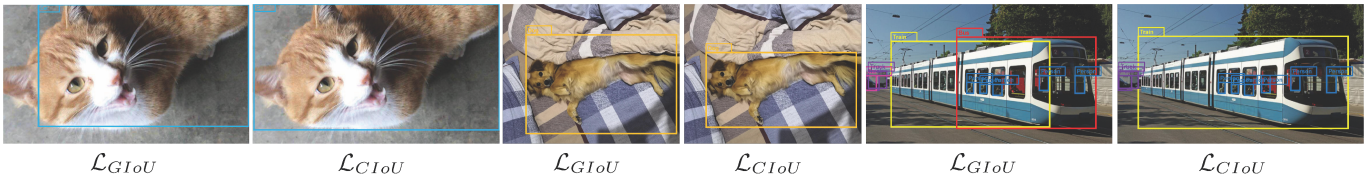



Fig. 10. Detection examples from Faster R-CNN [14] trained on MS COCO 2017.

TABLE IX
QUANTITATIVE COMPARISON OF FASTER R-CNN [14] TRAINED USING $\mathcal{L}_{IoU}$ (BASELINE), $\mathcal{L}_{GIoU}$, $\mathcal{L}_{DIoU}$ AND $\mathcal{L}_{CIoU}$. CLUSTER-NMS$_{W+D}$ IS APPLIED ON THE MODEL TRAINED USING $\mathcal{L}_{CIoU}$, WHILE THE OTHER RESULTS ARE PRODUCED BY ORIGINAL NMS. THE RESULTS ARE REPORTED ON THE VALIDATION SET OF MS COCO 2017.

| Loss / Evaluation | AP | AP$_{75}$ | AP$_S$ | AP$_M$ | AP$_L$ |
|---|---|---|---|---|---|
| $\mathcal{L}_{IoU}$ | 37.9 | 40.8 | 21.6 | 40.8 | 50.1 |
| $\mathcal{L}_{GIoU}$ | 38.0 | 41.1 | 21.5 | 41.1 | 50.2 |
| $\mathcal{L}_{DIoU}$ | 38.1 | 41.1 | **21.7** | 41.2 | 50.3 |
| $\mathcal{L}_{CIoU}$ | 38.7 | 42.0 | 21.3 | 41.9 | 51.5 |
| Cluster-NMS$_{W+D}$ | **39.0** | **42.3** | **21.7** | **42.2** | **52.1** |

*3) Faster R-CNN [14]:* We also evaluate CIoU loss for a two-stage object detection method Faster R-CNN[7] on MS COCO 2017 [51]. Following the same training protocol of [34], we have trained the models using CIoU loss in comparison with IoU, GIoU and DIoU losses. The backbone network is ResNet-50-FPN. Table IX reports the quantitative comparison. The gains of CIoU loss in AP are not as significant as those in YOLO v3 and SSD. It is actually reasonable that the initial boxes filtrated by RPN are likely to have overlaps with ground-truth box, and then all DIoU, GIoU and CIoU losses can make good regression. But due to the complete geometric factors in CIoU loss, the detected bounding boxes will be matched more perfectly, resulting in moderate gains. From Fig. 10, the detected boxes by our CIoU loss are more accurate than GIoU loss. As for suppressing redundant boxes, Cluster-NMS$_{W+D}$ is applied on the model trained by CIoU loss, and it brings further improvements for all the evaluation metrics, validating the effectiveness of assembling geometric factors into Cluster-NMS against Original NMS.

## VI. CONCLUSION

In this paper, we proposed to enhance geometric factors into CIoU loss and Cluster-NMS for object detection and instance segmentation. By simultaneously considering the three geometric factors, CIoU loss is better for measuring bounding box regression when training deep models of object detection and instance segmentation. Cluster-NMS is purely implemented on GPU by implicitly clustering detected boxes, and is much more efficient than original NMS. Geometric factors can then be easily incorporated into Cluster-NMS, resulting in notable improvements in precision and recall, while maintaining high inference efficiency. CIoU loss and Cluster-NMS have been applied to the training and inference of state-of-the-art deep object detection and instance segmentation models. Comprehensive experiments have validated that the proposed methods contribute to consistent improvements of AP and AR, and the high efficiency of Cluster-NMS can guarantee the real-time inference. CIoU loss and Cluster-NMS can be widely extended to other deep models for object detection and instance segmentation.

[7] https://github.com/generalized-iou/Detectron.pytorch

## REFERENCES


[1] X. Wang, M. Wang, and W. Li, "Scene-specific pedestrian detection for static video surveillance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 361–374, 2014. 1

[2] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe, "Siam r-cnn: Visual tracking by re-detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[3] J. Marłn, D. Vzquez, A. M. Lpez, J. Amores, and L. I. Kuncheva, "Occlusion handling via random subspace classifiers for human detection," *IEEE Transactions on Cybernetics*, vol. 44, no. 3, pp. 342–354, 2014. 1

[4] W. Wu, Y. Yin, X. Wang, and D. Xu, "Face detection with different scales based on faster r-cnn," *IEEE Transactions on Cybernetics*, vol. 49, no. 11, pp. 4017–4028, 2019. 1

[5] B. Xue and N. Tong, "Diod: Fast and efficient weakly semi-supervised deep complex isar object detection," *IEEE Transactions on Cybernetics*, vol. 49, no. 11, pp. 3991–4003, 2019. 1

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009. 1, 3

[7] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. 1, 2, 3

[8] J. Redmon and F. Ali, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018. 1, 2, 3, 5, 8, 9, 10, 12

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37. 1, 2, 3, 5, 8, 11

[10] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," *arXiv:1701.06659*, 2017. 1, 3

[11] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020. 1, 2, 3

[12] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 9626–9635. 1, 3

[13] R. Girshick, "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. 1, 2

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017. 1, 2, 3, 5, 7, 8, 12

[15] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1134–1142. 1

[16] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6154–6162. 1, 3

[17] X. Chen, R. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 2061–2069. 1, 2

[18] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," *arXiv preprint arXiv:1909.13226*, 2019. 1, 3

[19] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: real-time instance segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 9157–9166. 1, 2, 3, 5, 6, 7, 8, 9, 10

[20] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask r-cnn," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020. 1, 3

[21] P. J. Huber *et al.*, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964. 2

[22] S.-H. Bae, "Object detection based on region decomposition and assembly," in *The AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8094–8101. 2

[23] G. Brazil, X. Yin, and X. Liu, "Illuminating pedestrians via simultaneous detection & segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4960–4969. 2

[24] C. Zhou, M. Wu, and S.-K. Lam, "Ssa-cnn: Semantic self-attention cnn for pedestrian detection," *arXiv preprint arXiv:1902.09080*, 2019. 2

[25] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3111–3122, 2018. 2, 3

[26] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai, "Rotation-sensitive regression for oriented scene text detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5909–5918. 2

[27] S. Qin, A. Bissacco, M. Raptis, Y. Fujii, and Y. Xiao, "Towards unconstrained end-to-end text spotting," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 4704–4714. 2

[28] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4490–4499. 2

[29] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 770–779. 2

[30] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5693–5703. 2

[31] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 7718–7727. 2

[32] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, "Unitbox: An advanced object detection network," in *Proceedings of the ACM International Conference on Multimedia*, 2016, pp. 516–520. 2, 3

[33] L. Tychsen-Smith and L. Petersson, "Improving object localization with fitness nms and bounded iou loss," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6877–6885. 2, 3

[34] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 658–666. 2, 3, 8, 12

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587. 2

[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. 2, 3

[37] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms ł improving object detection with one line of code," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5562–5570. 2, 3, 5, 7

[38] H. Zhou, Z. Li, C. Ning, and J. Tang, "Cad: Scale invariant framework for real-time object detection," in *The IEEE International Conference on Computer Vision (ICCV Workshop)*, 10 2017, pp. 760–768. 2, 3, 5, 7

[39] D. Oro, C. Fernández, X. Martorell, and J. Hernando, "Work-efficient parallel non-maximum suppression for embedded gpu architectures," in *The IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 1026–1030. 2, 3

[40] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU Loss: Faster and better learning for bounding box regression," in *The AAAI Conference on Artificial Intelligence*, 2020. 2, 5, 7, 8

[41] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu, "Scale-transferrable object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 528–537. 3

[42] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 9656–9665. 3

[43] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2cnn: Rotational region cnn for orientation robust scene text detection," *arXiv preprint arXiv:1706.09579*, 2017. 3

[44] G. P. Meyer, "An alternative probabilistic interpretation of the huber loss," *arXiv preprint arXiv:1911.02088*, 2019. 3

[45] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 821–830. 3

[46] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *The AAAI Conference on Artificial Intelligence*, 2019, pp. 8577–8584. 3

[47] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *The European Conference on Computer Vision (ECCV)*, 2018, pp. 784–799. 3

[48] S. Liu, D. Huang, and Y. Wang, "Adaptive nms: Refining pedestrian detection in a crowd," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6452–6461. 3

[49] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2883–2892. 3

[50] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. 8

[51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *The European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755. 8, 9, 12

[52] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, A. Hogan, lorenzomammana, tkianai, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, J. Poznanski, L. Y. , changyu98, P. Rai, R. Ferriday, T. Sullivan, W. Xinyu, YuriRibeiro, E. R. Claramunt, hopesala, pritul dave, and yzchen, "ultralytics/yolov5: v3.0," aug 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3983579 11