PSRR-MaxpoolNMS++: Fast Non-Maximum Suppression with Discretization and Pooling

Tianyi Zhang, Chunyun Chen, Yun Liu, Xue Geng, Mohamed M. Sabry Aly, Jie Lin

Abstract—Non-maximum suppression (NMS) is an essential post-processing step for object detection. The de-facto standard for NMS, namely GreedyNMS, is not parallelizable and could thus be the performance bottleneck in object detection pipelines. MaxpoolNMS is introduced as a fast and parallelizable alternative to GreedyNMS. However, MaxpoolNMS is only capable of replacing the GreedyNMS at the first stage of two-stage detectors like Faster R-CNN. To address this issue, we observe that MaxpoolNMS employs the process of *box coordinate discretization* followed by *local score argmax calculation*, to discard the nested-loop pipeline in GreedyNMS to enable parallelizable implementations. In this paper, we introduce a simple *Relationship Recovery* module and a *Pyramid Shifted MaxpoolNMS* module to improve the above two stages, respectively. With these two modules, our **PSRR-MaxpoolNMS** is a generic and parallelizable approach, which can completely replace GreedyNMS at all stages in all detectors. Furthermore, we extend PSRR-MaxpoolNMS to the more powerful **PSRR-MaxpoolNMS++**. As for *box coordinate discretization*, we propose *Density-based Discretization* for better adherence to the target density of the suppression. As for *local score argmax calculation*, we propose an *Adjacent Scale Pooling* scheme for mining out the duplicated box pairs more accurately and efficiently. Extensive experiments demonstrate that both our PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ outperform MaxpoolNMS by a large margin. Additionally, PSRR-MaxpoolNMS++ not only surpasses PSRR-MaxpoolNMS but also attains competitive accuracy and much better efficiency when compared with GreedyNMS. Therefore, PSRR-MaxpoolNMS++ is a parallelizable NMS solution that can effectively replace GreedyNMS at all stages in all detectors.

Index Terms—Non-maximum suppression, object detection, fast NMS, parallelizable NMS

1 Introduction

Object detection is one of the most fundamental tasks in computer vision, with the objective of localizing and classifying objects in a scene. In the last decade, deep neural networks have emerged as the champion of object detection [2]–[4]. Deep-learning-based object detectors can be broadly grouped into either one-stage detectors like SSD [4] and YOLO [5] or two-stage detectors like Faster R-CNN [3] and R-FCN [6], in which neural networks often account for the majority of computing operations. On the other side, significant progress has been made towards better-performing dedicated hardware for accelerating the basic network operations (*e.g.*, convolution, fully-connected layer, and pooling) by exploiting their inherent parallelism, such as GPUs and Google TPUs [7]. As a result, the execution time spent on network operations is decreasing rapidly, *e.g.*, at milliseconds.

Non-maximum suppression (NMS), as a must-have post-processing technique in deep-learning-based object detectors, is likely to become the performance bottleneck in object detection pipelines [8]. The de-facto standard for NMS, namely GreedyNMS, mainly follows a nested-loop pipeline which is inefficient to implement. Such a nested-loop pipeline greedily picks out the box candidate with the highest confidence score and removes the boxes heavily overlapped with the picked one. Each iteration picks out only one box. More importantly, it is difficult to parallelize and accelerate the nested-loop pipeline with parallelism-friendly hardware like GPUs. Thus, GreedyNMS would gradually *dominate* the execution time of deep-learning-based object detectors [8], as neural networks run faster thanks to the increasing parallelism on dedicated hardware (*e.g.*, from P100 to V100 GPUs).

To address this problem, MaxpoolNMS [8] is proposed as a fast and parallelizable alternative to GreedyNMS, which discards the nested-loop pipeline. It is inspired by the observation that bounding boxes with high confidence scores correlate to peak values on the so-called *confidence score maps*, in which the spatial relationships among anchor boxes are preserved. Therefore, NMS can be designed as simple max-pooling on the confidence score maps, which is efficient and inherently parallel. However, MaxpoolNMS [8] is dedicated only to replacing GreedyNMS at the first stage of two-stage detectors, *e.g.*, the GreedyNMS after the Region Proposal Network (RPN) in Faster R-CNN [3]. There is a significant drop in detection accuracy when directly applying MaxpoolNMS to the final predicted bounding boxes. This lowers the value of MaxpoolNMS because it cannot be used to replace GreedyNMS at all stages in all detectors.

The objective of this paper is a fast and parallelizable NMS approach that can be applied to all stages of two-stage object detectors as well as one-stage detectors. For this goal, we propose **PSRR-MaxpoolNMS**, an alternative to MaxpoolNMS [8] to completely replace GreedyNMS at all stages of all detectors, instead

- *This work was supported by the National Natural Science Foundation of China (Grant No. 62202024), the Fundamental Research Funds for the Central Universities, and the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021).*
- *T. Zhang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China. (E-mail: zhang_tianyi@buaa.edu.cn)*
- *C. Chen and M. Aly are with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, 639798. (E-mail: chunyun001@e.ntu.edu.sg; msabry@ntu.edu.sg)*
- *Y. Liu is with the College of Computer Science, Nankai University, Tianjin 300350, China. (E-mail: vagrantlyun@gmail.com)*
- *X. Geng and J. Lin are with the Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore, 138632. (E-mail: geng_xue@i2r.a-star.edu.sg; jie.dellinger@gmail.com)*
- *Corresponding author: Yun Liu (E-mail: vagrantlyun@gmail.com).*
- *A preliminary version of this work has been published on CVPR 2021 [1].*

Manuscript received May 12, 2023; revised Sep 19, 2024.

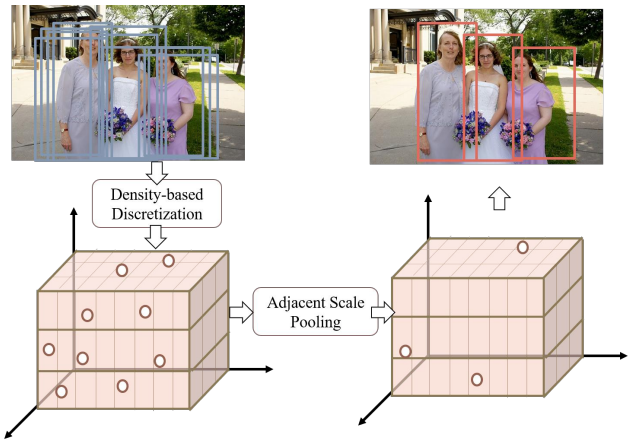


Fig. 1. **Pipeline of our PSRR-MaxpoolNMS++ for object detection.** *Density-based Discretization* allocates bounding boxes into discrete centers, followed by *Adjacent Scale Pooling* to eliminate overlapped boxes effectively and efficiently.

of only at the first stage of two-stage detectors like MaxpoolNMS [8]. The proposed PSRR-MaxpoolNMS follows the scheme of *box coordinate discretization* and then *local score argmax calculation* to discard the nested-loop pipeline in GreedyNMS for acceleration. Box coordinate discretization refers to allocating boxes into discrete centers based on the box locations, scales, and shapes to form confidence score maps. Local score argmax calculation refers to picking out the box with the maximum confidence score within certain neighboring ranges of box coordinates (*e.g.*, max-pooling keeps the box with the maximum score within each pooling window). Inspired by MaxpoolNMS [8], our PSRR-MaxpoolNMS proposes ***Relationship Recovery*** and ***Pyramid Shifted MaxpoolNMS*** to improve the above two stages, respectively. Specifically, Relationship Recovery resolves the mismatch problem in confidence score maps caused by the fact that MaxpoolNMS [8] projects anchor boxes to score maps without consideration of box regression. Pyramid Shifted MaxpoolNMS resolves the difficulty of maximizing the score map sparsity caused by the fact that MaxpoolNMS [8] only uses a single-scan max-pooling on the confidence score maps. PSRR-MaxpoolNMS has been published in our preliminary conference paper [1].

Although PSRR-MaxpoolNMS has been generalized to have almost the same application capability as GreedyNMS, it still has potential drawbacks in detection accuracy and efficiency. First, the discretization stage, Relationship Recovery, is anchor-based discretization, which means the discrete scale/ratio centers are inherited from the anchor box settings in detectors. It only relies on the empirical scale/ratio settings of anchor boxes, which cannot directly adapt to the target density. Second, the local argmax calculation stage sequentially performs multiple types of max-pooling, which does not consider the distribution of candidate boxes and is thus redundant and ineffective. In this paper, we make plentiful extensions over our preliminary conference version PSRR-MaxpoolNMS. Our new approach is also a generic NMS acceleration approach for all stages of all object detectors, dubbed as **PSRR-MaxpoolNMS++**. The overall pipeline of our new approach is illustrated in Fig. 1. PSRR-MaxpoolNMS++ improves both the box coordinate discretization and local score argmax calculation for better detection accuracy and efficiency. Besides, we extend PSRR-MaxpoolNMS++ to SoftNMS [9] to show its generic acceleration effect. The improvement of PSRR-MaxpoolNMS++ over PSRR-MaxpoolNMS can be summarized as follows:

- Box coordinate discretization – PSRR-MaxpoolNMS++ introduces ***Density-based Discretization*** to calculate discrete centers based on the target density that the suppression process aims to achieve.
- Local score argmax calculation – PSRR-MaxpoolNMS++ presents ***Adjacent Scale Pooling*** to effectively and efficiently identify the local ranges of local score argmax calculation.
- Extension to SoftNMS – We extend our discretization and local argmax calculation to SoftNMS [9] to show its generic acceleration effect.

PSRR-MaxpoolNMS++ significantly improves the performance of PSRR-MaxpoolNMS. It achieves comparable detection accuracy to GreedyNMS while being parallelizable. Both PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ avoid the nested-loop pipeline in GreedyNMS. PSRR-MaxpoolNMS++ has much fewer rounds of iterations than PSRR-MaxpoolNMS, leading to faster speed. The detailed comparisons of various NMS approaches are listed in Table 1. Our approaches are generic to all stages of various object detectors with much-improved efficiency.

2 Related Works

2.1 One-stage and Two-stage Object Detectors

Deep-learning-based object detection frameworks can be roughly classified into one-stage detectors and two-stage detectors. Two-stage detectors [2], [3], [10]–[13], like Faster R-CNN [3] and Mask R-CNN [10], are based on the class-agnostic region proposals. The region proposals are the candidate bounding boxes that potentially enclose target objects. R-CNN [13] and Fast R-CNN [2] employ hand-crafted region proposal generation [14], [15]. Instead, Faster R-CNN [3] generates region proposals by training a built-in RPN. The features of object proposals are fed into the subsequent detection network to predict the final box coordinates and class-specific probabilities for each proposal. Some works [11], [12] strive to enhance the detection performance by leveraging contextual cues and regularization strategies. One-stage detectors, like SSD [4] and YOLO [5], skip the region proposal stage in two-stage detectors and directly detect objects over a dense sampling of anchor locations. One-stage detectors usually have faster inference speed than two-stage detectors, possibly sacrificing detection accuracy.

2.2 Non-maximum Suppression

The final goal of object detectors is to output exactly one bounding box to tightly enclose each target object. However, most deep-learning-based object detection pipelines tend to generate redundant highly-overlapped bounding boxes to enclose an object, thus introducing a large number of false positives. Non-maximum suppression (NMS) is an essential step to suppress the redundant bounding boxes. NMS is usually applied either to pre-filtering the class-agnostic proposals to increase the efficiency or post-processing the final class-specific predictions to improve the detection accuracy.

The most widely used NMS method is GreedyNMS [16]. GreedyNMS first sorts the boxes by their confidence scores in descending order, then iteratively selects the most confident predictions from the remaining boxes and eliminates all the other

TABLE 1
Comparison of different NMS approaches. The comparisons are made in aspects of the overall pipeline, application scenarios, complexity, and technical details. SC: Single-Channel MaxpoolNMS; CR: Cross-Ratio MaxpoolNMS; CS: Cross-Scale MaxpoolNMS; All: Cross-all-Channel MaxpoolNMS; ASP: Adjacent Scale Pooling; SF: Shifted MaxpoolNMS.

Methods		GreedyNMS	MaxpoolNMS [8]	PSRR-MaxpoolNMS	PSRR-MaxpoolNMS++
General pipeline	Nested loop	✓	×	×	×
	Parallelizable	×	✓	✓	✓
	Arbitrary detectors/stages	✓	×	✓	✓
Discretization	Anchor-based discre.	-	✓	✓	×
	Density-based discre.	-	×	×	✓
Local-argmax calculation	Pooling types	-	SC/CR/CS	SC (+SF) +CR (+SF) +CS (+SF) +All (+SF)	ASP (+SF)
	Scan types	Until exhausted	1	8	2
Extension to SoftNMS		✓	-	-	✓

boxes that have a large overlap with the selected ones. This iteration terminates after the left boxes are exhausted or we have picked out a sufficient number of boxes. There are some variants of NMS to increase the detection accuracy [9], [17]–[20]. SoftNMS [9] decreases the scores of the boxes to be suppressed, instead of deleting these boxes by hard thresholding. Adaptive NMS [17] learns to adaptively set the box selection threshold according to the object density. Hosang *et al.* [18] reformulates NMS as a *convnet* that can be trained end to end. Visibility Guided NMS [19] leverages the detection of the whole objects as well as the detection of the visible parts to tackle the problem of highly occluded object detection. FeatureNMS [20] leverages the feature embedding distance to determine whether to suppress or keep the candidate boxes. However, these NMS methods need to form a nested-loop iteration, which is hard for parallel implementation when considering real-world deployment.

When it comes to NMS acceleration for practical deployment, it has been less explored. MaxpoolNMS [8] switches GreedyNMS to simple max-pooling on the score maps which encode confidence scores, scales, ratios, and spatial locations of anchor boxes. After max-pooling, only boxes with peak scores are kept and the others are suppressed. MaxpoolNMS is a fast and parallelizable NMS approach. However, it is only confined to the first stage RPN of the two-stage detectors. When applied to one-stage detectors or the second stage of the two-stage detectors, it would lead to performance degradation. We review MaxpoolNMS in detail in §3.1 and discuss its limitations in §3.2. Hash-NMS [21] maps each box into different hash cells and removes the non-maximum boxes within each hash cell. It aims to perform pre-filtering before the GreedyNMS to accelerate the detection on the crowded data. In this paper, we focus on NMS acceleration. Different from MaxpoolNMS [8] and Hash-NMS [21], our proposed approach could be applied to the final detection stage for both one-stage and two-stage detectors to directly filter the detection results.

2.3 End-to-end Object Detection

Recently, end-to-end object detection approaches have achieved competitive performance. These approaches directly output the sparsely aligned bounding-box predictions without the need of NMS for duplicate removal. DETR [22] totally discards NMS based on the vision transformer (self-attention) mechanism as well as the bipartite matching between box predictions and ground truths. However, it suffers from the failure of small object detection and slow training convergence. To tackle these problems, many variants of DETR are proposed to enhance the learning of one-to-one matching. Deformable DETR [23] narrows the search range of each object query to a small set of feature points. DN-DETR [24] and DINO [25] demonstrate that the slow convergence comes from the one-to-one matching and thus propose denoising techniques to improve the performance. Group DETR [26] constructs group-wise one-to-many label assignments to utilize multiple positive object queries. Co-DETR [27] utilizes the one-to-many detectors to guide the training of end-to-end detectors.

Although end-to-end detectors show promising performance, it is still important to investigate and explore NMS techniques. The current one-to-one assignment approach is not efficient for training, hence a one-to-many approach along with NMS filtering has been proposed to address this issue [28], [29]. One-to-many detectors still outperform end-to-end detectors [30], [31] in terms of model size, inference speed, and accuracy, making them popular in real-world applications. Therefore, this paper focuses on improving the efficiency of object detection by accelerating NMS.

3 PSRR-MaxpoolNMS

In this section, we first briefly review MaxpoolNMS [8] in §3.1 and analyze its limitations in §3.2. Then, we introduce our PSRR-MaxpoolNMS to address the limitations. PSRR-MaxpoolNMS is composed of two steps: Relationship Recovery in §3.3.1, followed by Pyramid Shifted MaxpoolNMS in §3.3.2. *It is worth noting that the spatial index and the channel of the score map refer to the discrete centers in the box coordinate discretization stage.*

3.1 Revisiting MaxpoolNMS

MaxpoolNMS [8] is a fast and parallelizable NMS approach, which is specifically designed for removing the overlapped anchor boxes at the first stage of Faster R-CNN detection pipeline [3], *i.e.*, the RPN. MaxpoolNMS is composed of two modules. First, as illustrated in Fig. 2, it constructs a set of confidence score maps, each of which corresponds to a specific combination of anchor box scale and aspect ratio (*i.e.*, channel c), and each cell on the score map encodes the objectness score (*i.e.*, cell value) and spatial location (*i.e.*, x and y on the map) of an anchor box generated by the RPN. For instance, if we use 4 anchor box scales

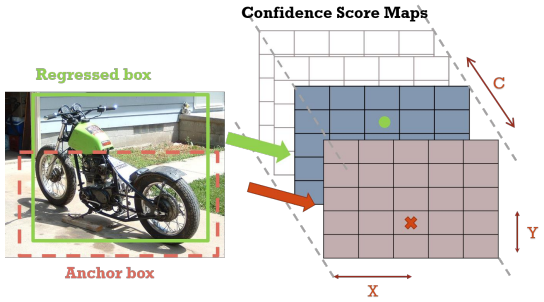


Fig. 2. **Illustration of the *score map mismatch* problem in MaxpoolNMS [8].** MaxpoolNMS projects an anchor box (dashed red) to a score map with an aspect ratio of $1:2$ (in coral), without consideration of box regression. The regressed box (solid green) of the anchor has changed in aspect ratio and is thus projected to another score map with an aspect ratio of $1:1$ (in blue). The projection of the regressed box is correct as it encloses the object more accurately than its corresponding anchor box.

$\{64^2, 128^2, 256^2, 512^2\}$ and 3 aspect ratios $\{1:2, 1:1, 2:1\}$ for an RPN with a downsampling ratio of β (*e.g.*, $\beta = 16$), there are 12 confidence score maps with a width of $\lfloor \frac{W}{\beta} \rceil$ and height of $\lfloor \frac{H}{\beta} \rceil$, where W and H denote the width and height of the input image, respectively. Second, based on the observation that objects correspond to peak scores on the confidence score maps, MaxpoolNMS applies a simple max-pooling on the maps to suppress anchor boxes with low scores and only keep anchor boxes with peak scores.

The max-pooling in MaxpoolNMS has three choices: 1) *Single-Channel MaxpoolNMS* applies max-pooling on each score map (channel) independently; 2) *Cross-Ratio MaxpoolNMS* concatenates score maps at two neighboring aspect ratios for each scale, followed by 3D max-pooling on the concatenated maps; 3) *Cross-Scale MaxpoolNMS* concatenates score maps at two neighboring scales for each aspect ratio, followed by 3D max-pooling on the concatenated maps. After max-pooling, the anchor boxes remaining on the score maps are combined and sorted by their scores in descending order. Only the top boxes are returned as final detections.

3.2 Limitations of MaxpoolNMS

Though MaxpoolNMS [8] is more efficient than GreedyNMS by using the max-pooling operations, MaxpoolNMS suffers from a shortcoming that it is dedicated only to replacing GreedyNMS at the first stage of two-stage object detectors. To maintain high detection accuracy, GreedyNMS is still a must-have post-processing method for the second stage of two-stage detectors and one-stage detectors such as SSD [4]. This makes MaxpoolNMS less attractive in the sense that it cannot be utilized to replace GreedyNMS at all stages in all detectors.

We observe that the detection accuracy drops significantly when applying MaxpoolNMS [8] at the second stage of two-stage detectors. Specifically, we adopt MaxpoolNMS after the second stage of Faster R-CNN [3] to remove overlapped boxes. As shown in Table 3, MaxpoolNMS performs significantly worse than GreedyNMS, with an over 40% drop in detection accuracy. We find that there are two key factors that may lead to the unsatisfactory performance of MaxpoolNMS [8], *i.e.*, the score map mismatch and the difficulty of maximizing the score map sparsity with a single-scan max-pooling:

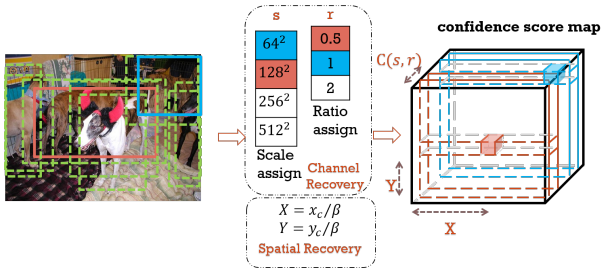


Fig. 3. **Relationship Recovery to solve the score map mismatch.**

- **Score map mismatch.** This occurs during the construction of confidence score maps. MaxpoolNMS projects anchor boxes to score maps without consideration of box regression. This leads to the score map mismatch problem if the regressed boxes corresponding to the anchor boxes have changed dramatically in location, scale, or aspect ratio. Fig. 2 shows an example of a change in ratio. The mismatch would cause wrong box projections on score maps, which in turn bring in negative effects on the following max-pooling operations.
- **Low sparsity of score maps.** Since MaxpoolNMS operates only a single-scan max-pooling on the confidence score maps, it is hard to achieve high sparsity on dense score maps, leading to a lot of highly-overlapped boxes remained after pooling, as illustrated in the left of Fig. 4 (*i.e.*, max-pooling with the single channel only). Moreover, a single-scan max-pooling on the confidence score maps would cause the edge effect. As shown in Fig. 5 left, the boxes in the adjacent cells are both kept after max-pooling, even though one of them is considered a duplication.

3.3 Our PSRR-MaxpoolNMS

3.3.1 ***Discretization:*** *Relationship Recovery*

Instead of projecting anchor boxes to the confidence score maps, our Relationship Recovery projects the regressed boxes to the maps, which solves the score map mismatch problem. With the help of box regression, the regressed boxes in general enclose the objects more accurately than their corresponding anchor boxes. As such, the score maps projected by the regressed boxes are able to better reflect the actual spatial and channel (a combination of scale and aspect ratio) relationships. Suppose the coordinates of the i^{th} regressed box b_i is represented as $[x_i^c, y_i^c, w_i, h_i]$, where x_i^c, y_i^c denotes the spatial position of the box center and w_i, h_i denotes the box width, height, respectively. Concretely, the Relationship Recovery module consists of three parts: spatial and channel recovery to infer the spatial index $[X_i, Y_i]$ and the channel $[S_i, R_i]$ which a regressed box should be mapped to, followed by the score assignment which assigns the confidence score to each cell in the maps. Here, S_i (or R_i) is the index of the discrete scale (or ratio) center assigned to. This process is depicted in Fig. 3.

Spatial Recovery. MaxpoolNMS [8] projects anchor boxes to wrong spatial locations on the score map due to the dramatic shift of locations after box regression. To address this location mismatch problem, Spatial Recovery maps the spatial location $[x_i^c, y_i^c]$ to the spatial index $[X_i, Y_i]$ on the score map as

$$X_i = \lfloor \frac{x_i^c}{\beta} \rceil, \quad Y_i = \lfloor \frac{y_i^c}{\beta} \rceil, \tag{1}$$

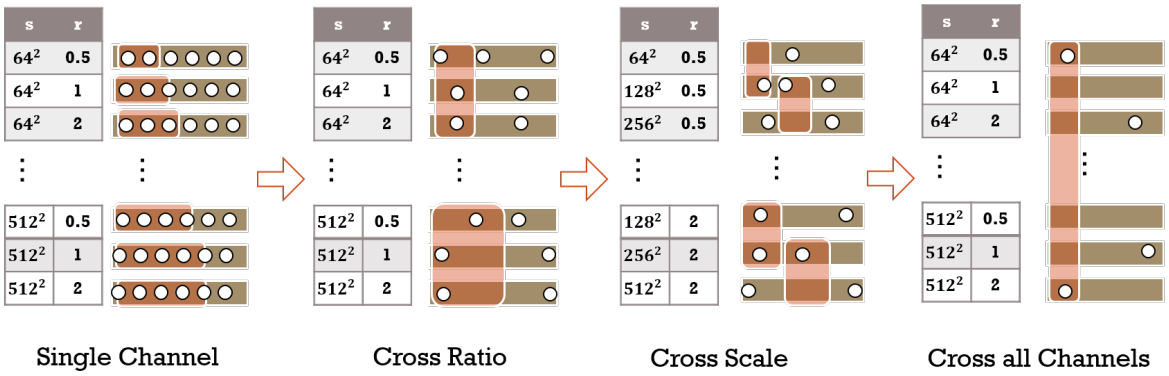


Fig. 4. **Illustration of Pyramid MaxpoolNMS.** A sequence of max-pooling is operated one after another on the confidence score maps, with different channel combinations and pooling parameters (kernel size and stride) determined by the scale (s) and ratio (r) of the map. As can be seen, the confidence score maps become more and more sparse with the Pyramid MaxpoolNMS (*i.e.*, multi-scan max-pooling).

where β is the downsampling ratio of the score maps.

Channel Recovery. MaxpoolNMS [8] determines the projected channel based on the default scale and ratio of the anchor box. However, the channel projection could be wrong if the corresponding regressed box has changed dramatically in scale and/or ratio. To solve this channel mismatch problem, Channel Recovery calculates the nearest scale S_i to $w_i \times h_i$ and the nearest ratio R_i to $\frac{h_i}{w_i}$ based on Euclidean-distance, and chooses $[S_i, R_i]$ as the projected channel.

Score Assignment. After the spatial locations and channels for all boxes are determined, each cell in the maps could have more than one box projected to it. Therefore, score assignment is introduced to only keep the box with the highest score in each cell. One may note that the score assignment is basically 1×1 max-pooling in each cell of the score maps, thus it can be treated as a pre-filtering step for removing overlapped boxes that are easy to identify.

Remarks. All operations of the relationship recovery are simple and parallelizable. Besides, the relationship recovery is anchor-free. In other words, as the first step of our PSRR-MaxpoolNMS, it opens up a possibility to extend PSRR-MaxpoolNMS from anchor-based one-stage or two-stage object detectors to anchor-free object detectors [32], [33], because the construction of confidence score maps does not rely on anchor boxes at all. Instead, it only requires the locations and sizes of regressed boxes, which are accessible in anchor-free detectors as well.

3.3.2 ***Local Score Argmax:*** *Pyramid Shifted MaxpoolNMS*

We propose *Pyramid Shifted MaxpoolNMS* to remove overlapped boxes on the confidence score maps, in which the *Pyramid MaxpoolNMS* aims to thoroughly suppress overlapped boxes across channels (scales and ratios), while the *Shifted MaxpoolNMS* aims to effectively eliminate overlapped boxes in the spatial domain by addressing the edge effect problem. After Pyramid Shifted MaxpoolNMS, the score maps become highly sparse, with only a small number of non-zero cells. The boxes in the non-zero cells are returned as final detections.

Pyramid MaxpoolNMS. Pyramid MaxpoolNMS is based on the max-pooling methods proposed by MaxpoolNMS [8]. Since each score map is dedicated to a specific anchor box size, the kernel size and pooling stride for max-pooling on that score map are determined by its associated anchor box size. Given the scale-ratio pair $[S, R]$ of the anchor boxes on a specific score map, the width and height of the corresponding anchor boxes $(\hat{w}, \hat{h})$ is defined as

$$\hat{w}(S,R) = \sqrt{\frac{S}{R}}, \quad \hat{h}(S,R) = \sqrt{SR}. \tag{2}$$

The corresponding kernel size (K_x, K_y) are defined as

$$\begin{aligned} K_x(S,R) &= \max(\lceil \frac{\alpha \hat{w}(S,R)}{\beta} \rfloor, 1), \\ K_y(S,R) &= \max(\lceil \frac{\alpha \hat{h}(S,R)}{\beta} \rfloor, 1), \end{aligned} \tag{3}$$

where K_x and K_y are the kernel sizes (as well as pooling strides) in x and y directions, respectively. The parameter α represents the overlap threshold, which is used to control the trade-off between precision and recall. A larger α would suppress more overlapped boxes (leading to higher precision) but at the risk of missing object detections (leading to lower recall).

Although MaxpoolNMS [8] can suppress redundant boxes using max-pooling operations, it has some shortcomings. On one hand, MaxpoolNMS [8] operates only a single-scan max-pooling on the confidence score maps. On the other hand, MaxpoolNMS assumes that overlapped boxes exist only in the channels with neighboring scales (or ratios) and the same ratio (or scale), which is not always true as the overlapped boxes can distribute at arbitrary scales and ratios to some extent. As such, a single-scan max-pooling operation with an invalid assumption may be insufficient to suppress all overlapped boxes, resulting in low sparsity of the score maps after pooling. Even though one can increase the overlap threshold α in Eq. (3) to induce higher sparsity, this could lead to the risk of missing true positive detections.

We propose Pyramid MaxpoolNMS to progressively induce sparsity in the score maps by performing a sequence of max-pooling operations one after another on the score maps with different channel combinations, as illustrated in Fig. 4. The sequence of max-pooling operations starts with *Single-Channel* max-pooling, followed by *Cross-Ratio* and *Cross-Scale* max-pooling, and ends with *Cross-all-Channels* max-pooling. As introduced in §3.1, Single-Channel max-pooling operates on a single score map independently, while Cross-Ratio and Cross-Scale max-pooling operate on multiple score maps by concatenating channels at adjacent

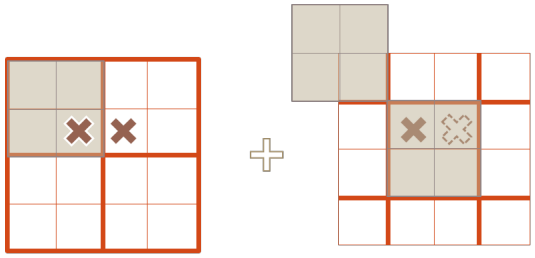


Fig. 5. **Shifted MaxpoolNMS to alleviate the edge effect.** *Left:* The boxes in the adjacent cells are both kept after max-pooling with the kernel size 2×2 and stride 2. *Right:* By adding another max-pooling with one cell shift, the box with the higher score is kept, while the other one is suppressed.

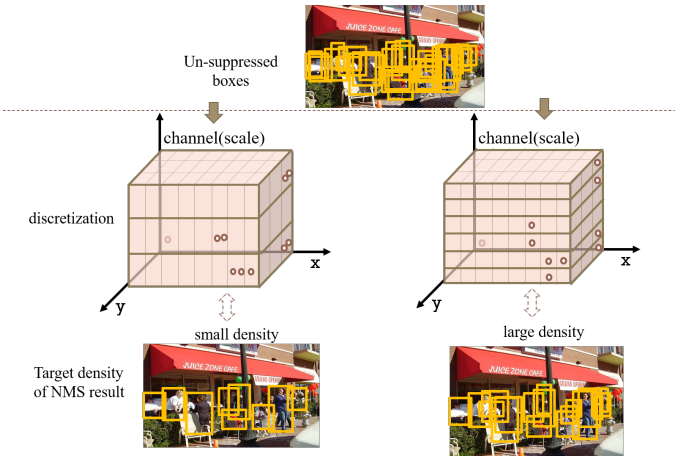


Fig. 6. **Illustration of *Density-based Discretization*.** The discretization interval (resolution) is decided by the target suppression density. The smaller the density (the overlap of the filtered box), the more likely the input un-suppressed boxes are allocated into the same discrete centers (cells in the figure). The non-maximum boxes within the same cell are more likely to be suppressed.

ratios and scales, respectively. In addition, we introduce Cross-all-Channels max-pooling which performs max-pooling across all channels. In this way, our Pyramid MaxpoolNMS gradually increases the "receptive field" of the pooling operations from local (single score map) to global (all maps), without requiring any assumption on the distribution of overlapped boxes.

When performing max-pooling on the single channel independently, the kernel size and stride for each channel are set as in Eq. (3). When performing max-pooling across multiple channels, the kernel size (or stride) is set as the minimum of kernel sizes (or strides) of the channels concatenated. On one hand, if the kernel size is larger than the minimum value, it may suppress true positives detected by the precedent Single-Channel max-pooling. On the other hand, the larger the gap between scales/ratios, the less likely to have overlapped boxes. Therefore, a small kernel size (or stride) could reduce the risk of suppressing true positives.

Shifted MaxpoolNMS. Shifted MaxpoolNMS can further increase the sparsity of confidence score maps and thus eliminate overlapped boxes in the spatial domain (X, Y) more effectively. This is achieved by introducing additional max-pooling with a spatial shift on the confidence score maps, which in turn addresses the edge effect problem, as shown in Fig. 5. Specifically, given a kernel size k, the shifted max-pooling is operated on the score maps padded with $\lfloor \frac{k}{2} \rceil$ zeros around the border. Finally, the shifted max-pooling can be appended after each pooling operation in the sequence of Pyramid MaxpoolNMS.

4 PSRR-MaxpoolNMS++

In this section, we first extend our PSRR-MaxpoolNMS to PSRR-MaxpoolNMS++ using the new Density-based Discretization for box coordinate discretization and Adjacent Scale Pooling for local score argmax calculation, as in §4.1. Then, we provide discussions about the underlying intuitions of PSRR-MaxpoolNMS++ in §4.2. Next, we introduce how to extend our discretization and local score argmax techniques to SoftNMS [9] in §4.3, suggesting the generality of our method. Finally, we present how to implement our method in §4.4.

4.1 From PSRR-MaxpoolNMS to PSRR-MaxpoolNMS++

Although our PSRR-MaxpoolNMS improves MaxpoolNMS [8] in terms of both discretization and local argmax calculation, PSRR-MaxpoolNMS still has the following shortcomings. As for the Relationship Recovery step, anchor-based discretization is irrelevant to the target density that the suppression process aims to achieve. The inappropriate setting of discrete centers may result in wrongly removing the true positives or keeping the false positives. As for the Pyramid max-pooling step, first, the Cross-all-Channels max-pooling is likely to wrongly suppress the true positives because the box pairs with dramatically different scales are not likely to be largely overlapped. Second, the scanning scheme with pyramid pooling is redundant and ineffective. Although Single-Channel, Cross-Ratio, and Cross-Scale max-pooling try to eliminate the duplicates under various settings, the combination of such three max-pooling always fails to completely remove redundant boxes in our observation.

To address the above shortcomings, we further extend PSRR-MaxpoolNMS to PSRR-MaxpoolNMS++ to improve its efficiency and accuracy. We inherit the core ideas of discretization and local argmax calculation in PSRR-MaxpoolNMS. The discretization is used to enable batch processing of box suppression, rather than calculating the overlap ratio for each individual pair of boxes. The local argmax calculation can identify all peaks in the confidence score map by a single scan of max-pooling, unlike GreedyNMS, which picks out a single candidate at a time. Moreover, local argmax can also foster parallelism due to the natural ability of the max-pooling operation. Here, we focus on two issues: 1) How to set a better interval between discrete centers? 2) How to effectively and efficiently define the local range of local argmax calculation? We tackle these issues by proposing Density-based Discretization and Adjacent Scale Pooling to improve the discretization and local score argmax calculation, respectively.

4.1.1 *Discretization: Density-based Discretization*

In the Density-based Discretization step illustrated in Fig. 6, we allocate the boxes to discrete centers. The boxes assigned to the same discrete center are of similar scales/ratios and could be suppressed in batches with the same parameter setting. For the i^{th} input un-supressed box b_i, we follow Relationship Recovery in §3.3.1 to calculate the discrete location $[X_i, Y_i, S_i, R_i]$. We define a fixed set of discrete centers in both scales and aspect ratios and assign the box b_i to the S_i-th discrete scale center and the R_i-th discrete ratio center.

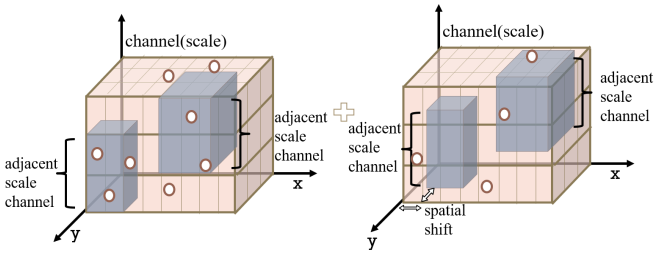


Fig. 7. **Illustration of *Adjacent Scale Pooling*.** The max-pooling operation is performed over the boxes assigned to adjacent discrete scale centers (channels). The shifted pooling is also applied to further increase the pooling sparsity.

The definition of discrete centers is crucial for the box feature representation and suppression performance. We propose *Density-based Discretization* by setting the discrete centers based on the target density. We apply Density-based Discretization to set the discrete scale centers. The set of discrete scale centers is expressed as $\{2^{10} \times \delta^{2j} | j = 0, 1, ..., N\}$, where δ indicates the interval between adjacent scale centers, and N is dynamically set to satisfy $2^{10} \times \delta^{2N-2} < 512^2 \leq 2^{10} \times \delta^{2N}$. It is worth noting that, compared to the anchor-based discrete scale centers, our density-based discrete scale centers have a smaller minimum discrete center (32^2 *vs.* 64^2). The motivation for this enlarged range is to ensure that the discrete centers can more completely cover the range of detected objects, thereby allowing each bounding box to be assigned to a more appropriate discrete scale center. More precise box-center assignment helps in setting accurate pooling kernels and strides, which consequently improves suppression accuracy. We aim to set the δ parameter according to the target density. The reasonable setting is that δ should be the monocular decreasing function of the target density: Given the same suppression strategy, the larger δ is, the more likely the boxes are assigned to the same discrete centers, and more boxes are likely to be suppressed, which will finally result in decreasing suppression density. In PSRR-MaxpoolNMS++, we define $\delta = \sqrt[4]{\frac{1}{\theta}}$, in which θ is the target density parameter to determine the overlapping extent of the suppressed boxes. The explanation of this definition is provided in §4.1.2.

4.1.2 ***Local Score Argmax:*** *Adjacent Scale Pooling*

At the pooling step, we first identify a set of independent local ranges based on the discretization result, followed by the argmax calculation within each range for suppressing the non-maximum box candidates by hard removal. Thus, the definition of the local range is the key issue for the suppression performance. Similar to PSRR-MaxpoolNMS, the suppression step of PSRR-MaxpoolNMS++ is built on the max-pooling operation and the local range is the discrete cubicle range defined by the max-pooling kernels. We follow the same kernel/stride settings for each individual discrete scale/ratio center as in PSRR-MaxpoolNMS (Eq. (2) and Eq. (3)) and focus on the centers that the max-pooling operation crosses.

Instead of a sequence of max-pooling operations in PSRR-MaxpoolNMS, PSRR-MaxpoolNMS++ performs a single *Adjacent Scale Pooling*. As shown in Fig. 7, Adjacent Scale Pooling is operated as follows: we perform max-pooling over the boxes belonging to adjacent scale pairs. For each pair of adjacent discrete scale centers, we group the boxes assigned to this pair regardless of the aspect ratios and then perform max-pooling over the grouped boxes. We scan all the adjacent discrete scale center pairs to thoroughly enumerate the overlapped box pairs to maximize the box sparsity. The reason for this design can be found in §4.2. As mentioned above, the spatial kernel (or stride) is calculated as the minimum value among the discrete scale-ratio pairs it operates on. Same as PSRR-MaxpoolNMS, after each step of the max-pooling scan, we perform an additional shifted max-pooling scan to address the edge effect as mentioned in §3.3.2. Since α in Eq. (3) determines the density of the suppressed boxes, it should be adaptively set according to the target density. α should be within the range of $[0, 1]$ as there may exist overlap between suppressed anchor boxes. Besides, α should be a monocular decreasing function of the density parameter: Given the same suppression strategy, the larger the α is, the more likely the boxes are covered by the same pooling kernel, and more boxes are likely to be suppressed, which will finally result in decreasing suppression density. In this paper, we define $\alpha = 1 - \theta$, where θ is the target density parameter.

With the above definition, we take the Adjacent Scale Pooling for adjacent scale centers $2^{10} \times \delta^{2j}$ and $2^{10} \times \delta^{2j+2}$ as an example to explain the definition of δ in §4.1.1. Since each box is assigned to its nearest scale center, the smallest scale of the boxes assigned to scale center $2^{10} \times \delta^{2j}$ is $2^{10} \times (\delta^{2j-2} + \delta^{2j})/2 = 2^9 \times (\delta^{2j-2} + \delta^{2j})$, and the largest scale of the boxes assigned to scale center $2^{10} \times \delta^{2j+2}$ is $2^{10} \times (\delta^{2j+2} + \delta^{2j+4})/2 = 2^9 \times (\delta^{2j+2} + \delta^{2j+4})$. Hence, the scale range of this scan of Adjacent Scale Pooling lies between $2^9 \times (\delta^{2j-2} + \delta^{2j})$ and $2^9 \times (\delta^{2j+2} + \delta^{2j+4})$. For simplicity, when considering boxes across adjacent scale centers, we disregard other factors such as spatial shifts and aspect ratios. Thus, the Intersection over Union (IoU) can be calculated as $\frac{\delta^{2j-2} + \delta^{2j}}{\delta^{2j+2} + \delta^{2j+4}} = \frac{1}{\delta^4}$, which is the ratio between the smallest and largest box scales. If the density parameter θ approximates the target IoU threshold, we have $\frac{1}{\delta^4} = \theta$, equivalent to $\delta = \sqrt[4]{\frac{1}{\theta}}$.

When setting the α parameter, for simplicity, assume that two identical overlapping anchor boxes only have a spatial shift in one dimension (*e.g.*, the horizontal dimension), while disregarding factors such as differences in scale, aspect ratio, or spatial shift in the other dimension. If the box height is h and the box width is w, the shift in the horizontal dimension is $\alpha \times w$, and the intersected area between the two boxes is $(1 - \alpha) \times w \times h$. For convenience, if the density parameter θ is interpreted as the ratio of the intersection area to the box area (*i.e.*, precision), we can derive $\theta = \frac{(1-\alpha) \times w \times h}{w \times h} = 1 - \alpha$. Notably, there is a subtle difference in the definition of θ between the settings of δ and α. However, when setting δ in the ideal case, the smaller box can be entirely enclosed by the larger box, where IoU is equivalent to precision.

Notably, in our PSRR-MaxpoolNMS++ approach, both the box scale and the distance between boxes are correlated to the target density. In the density-based discretization (§4.1.1), the effect of the box scale is considered because the interval δ between adjacent scale centers is set as the monocular decreasing function of the target density. In the local score argmax calculation (§4.1.2), the impact of the distance between boxes is considered when adjusting the kernel sizes and strides of max-pooling using Eq. (3), and the influence of the box scale is considered when determining the scale centers that the max-pooing operation crosses.

	64×64	128×128	256×256	512×512
64×64	4.49%	0.90%	0.00%	0.00%
128×128	0.90%	10.57%	3.52%	0.00%
256×256	0.00%	3.52%	32.37%	12.20%
512×512	0.00%	0.00%	12.20%	19.35%

Fig. 8. Distribution of overlapped box pairs with $\mathrm{IoU} > 0.3$ over discrete scale centers generated from the PASCAL VOC dataset [34] with the Faster R-CNN (ResNet-50 [35]) detector [3].

	0.5	1	2
0.5	21.17%	10.66%	0.47%
1	10.66%	35.41%	6.11%
2	0.47%	6.11%	8.95%

Fig. 9. Distribution of overlapped box pairs with $\mathrm{IoU} > 0.3$ over discrete ratio centers generated from the PASCAL VOC dataset [34] with the Faster R-CNN (ResNet-50 [35]) detector [3].

TABLE 2
Statistics of scale/ratio similarity between the overlapped box pairs with $\mathrm{IoU} > 0.3$ generated from the PASCAL VOC dataset [34] with the Faster R-CNN detector [3]. "mean": the average similarity; "std": standard deviation.

	ResNet-50 [35]	ResNet-101 [35]	ResNet-152 [35]
scale-mean	85.3%	89.1%	90.4%
scale-std	15.6%	12.7%	13.4%
ratio-mean	89.0%	90.7%	92.6%
ratio-std	11.6%	10.2%	10.1%

4.2 Discussion: Scale *vs.* Ratio

In both our Density-based Discretization and Adjacent Scale Pooling, we emphasize the scale similarity instead of the ratio similarity to estimate the overlap extent. In this part, we discuss the empirical intuition of emphasizing scales.

We first calculate statistics of the overlapped boxes over the discrete scale/ratio centers to show that the boxes with large overlaps are usually distributed at adjacent scales and arbitrary aspect ratios. We take the anchor-based discrete scale/ratio centers in PSRR-MaxpoolNMS as an example. Here, the discrete scale centers are $\{64^2, 128^2, 256^2, 512^2\}$ and the discrete ratio centers are set as $\{0.5, 1, 2\}$. For each image, we calculate the IoU score between each pair of un-suppressed candidate boxes and record the discrete scale/ratio centers of largely overlapped pairs with $\mathrm{IoU} > 0.3$. Fig. 8 and Fig. 9 report the distributions of overlapped pairs over discrete scale and ratio centers, respectively. As can be observed, overlapped pairs are distributed across adjacent scale centers and arbitrary ratio centers. Even a box pair assigned to distant ratio centers (*e.g.*, 0.5 *vs.* 2) still has the chance of largely overlapping. However, a box pair assigned to distant scale centers (*e.g.*, 64 *vs.* 256) has no chance of largely overlapping. Thus, we are motivated to perform the local argmax calculation across adjacent scales regardless of the ratios.

We proceed by calculating statistics about the scale and ratio similarity between the largely overlapped box pairs to show that scale similarity is more difficult to achieve and thus receives more attention to precisely define the adjacent scales. For each pair of largely overlapped (*e.g.*, $\mathrm{IoU} > 0.3$) boxes, we evaluate their scale/ratio similarity by the division between the smaller and bigger values. The statistics of the scale/ratio similarity are reported in Table 2. We can see that the largely overlapped box pairs are similar in both scale and ratio (the mean similarity values are both close to 100%). However, the ratios are more similar (larger mean similarity) and clustered (smaller standard deviation) over the scales. This indicates that ratio similarity among largely overlapped pairs is more easily achieved by detectors and could thus be discarded in identifying large overlapping extent. Thus, it is vital to differentiate the scale difference to accurately identify the largely overlapped boxes, which motivates us to investigate the interval between adjacent scale centers by the target density.

4.3 Extension to SoftNMS

In this subsection, we extend our schemes of box coordinate discretization and local score argmax calculation to SoftNMS [9]. The original SoftNMS follows the following pipeline. For each iteration, the box with the largest confidence score among the left un-suppressed boxes is picked and added to the picked set. Then, the weights for other left boxes are calculated based on the overlap rate between themselves and the picked one: the larger the overlap rate, the smaller the weight. Next, the weights (from 0 to 1) are multiplied by the scores of the left boxes. In this way, the boxes that have a large overlap with the picked box are suppressed through score decreasing instead of directly being deleted. After that, the left boxes are filtered by thresholding the weighted scores. Such an iteration terminates after the left boxes are exhausted. In each iteration, the weight for the unpicked box b_i is computed as $e^{-\frac{[\mathrm{IoU}(b_i, b_t)]^2}{\sigma}}$, where b_t is the picked box in the t^{th} iteration and σ is a tunable parameter.

Similar to §4.1, we assign the boxes into different discrete scale/ratio centers and define the local range in the same way as Adjacent Scale Pooling to perform suppression. The main difference lies in how the suppression works within each local range. In each scan, we pick the box with the largest confidence score and calculate the weights for other boxes based on their overlap rates with the picked one. Then, the box confidence scores are diminished by the weight multiplication within the corresponding local range.

Besides discretizing local ranges for suppression, we also propose the discretized weight calculation. We have three observations: 1) the discretized local range will naturally discrete the box scales, ratios, and spatial shifts; 2) the discrete scale/ratio centers and spatial shifts are also countable; 3) the weight calculation is only confined within limited local ranges. It is also easy to know that the IoU of two bounding boxes can be formulated by a function of box scales, aspect ratios, and spatial shifts. Hence, the discretized weights (or IoU score) could be enumerated. Taking advantage of this, we build a look-up table of the weights, whose indices are the functions of discrete centers and center shifts to reduce the time of weight calculation.

4.4 Implementation

In this part, we introduce how to implement our PSRR-MaxpoolNMS++. As in §3, the kernel size and stride of each pooling scan are equal. For the pooling scan with the spatial kernel size k_x, k_y over the i^{th} box $b_i = [X_i, Y_i, S_i, R_i]$, we calculate the following key as the index for encoding the box:

$$\mathrm{Key}(b_i) = [\lfloor \frac{X_i}{k_x} + \gamma \rfloor, \lfloor \frac{Y_i}{k_y} + \gamma \rfloor, \lfloor \frac{S_i + \omega}{2} \rfloor], \tag{4}$$

where $\gamma \in \{0, 0.5\}$ denotes the spatial shift parameter: $\gamma = 0$ denotes the original max-pooling operation while $\gamma = 0.5$ denotes

Algorithm 1 PSRR-MaxpoolNMS++ Implementation

Input: The set $\mathbf{B}$ of un-suppressed box candidates, and the set $\mathbf{P}$ of corresponding confidence scores for one object class.
Output: The suppressed boxes
for each box b_i in $\mathbf{B}$ **do**
  Perform coordinate discretization for b_i
end for
for ω in $\{0, 1\}$ **do**
  for γ in $\{0, 0.5\}$ **do**
    keep_index = Dict()
    keep_score = Dict()
    for b_i in $\mathbf{B}$ **do**
      Encode b_i using $\mathrm{Key}(b_i)$ in Eq. (4)
      if $\mathrm{Key}(b_i)$ is in keep_index **then**
        if $p_i >$ keep_score[$\mathrm{Key}(b_i)$] **then**
          keep_index[$\mathrm{Key}(b_i)$] = i
          keep_score[$\mathrm{Key}(b_i)$] = p_i
        end if
      else
        keep_index[$\mathrm{Key}(b_i)$] = i
        keep_score[$\mathrm{Key}(b_i)$] = p_i
      end if
    end for
    $\mathbf{B}$ = $\mathbf{B}$[keep_index.values()]
  end for
end for

the additional shifted max-pooling operation with spatial shift of $[\lfloor \frac{k_x}{2} \rfloor, \lfloor \frac{k_y}{2} \rfloor]$. $\omega \in \{0, 1\}$ denotes the shifts parameter in the discrete scale centers. For example, $\omega = 0$ denotes the max-pooling over the scale index pairs of $\{(0, 1), (2, 3), ...\}$ and $\omega = 1$ denotes the max-pooling over the scale index pairs of $\{(1, 2), (3, 4), ...\}$. As mentioned in §3.3.2, k_x (or k_y) is the minimum value of the kernels of the discrete scales that the max-pooling operation covers. We can formulate it as $k_x = \min_{\{j | \lfloor \frac{S_j+\omega}{2} \rfloor = \lfloor \frac{S_i+\omega}{2} \rfloor\}} K_x(S_j, R_j)$, in which $K_x(S_j, R_j)$ is the horizontal kernel size of the S_j-th discrete scale center and the R_j-th discrete ratio center. In each pooling scan with a fixed spatial shift γ and a discrete scale shift ω, each box is assigned a key code as Eq. (4). If multiple boxes share the identical key code, they are considered within the same local range, in which the non-maximum ones should be deleted for suppression. We enumerate all the pairs of (γ, ω) combinations to sequentially perform multiple times of max-pooling scans. The pseudo-code is formulated in Algorithm 1.

After the key for each box has been encoded as in Eq. (4), we utilize PyTorch [36] and PyTorch-Scatter libraries to implement Adjacent Scale Pooling, which is parallelizable across candidate boxes. In our previous conference version, we implement PSRR-MaxpoolNMS in §3 using the PyTorch [36] max-pooling API. The pooling efficiency is related to the resolution of the confidence score map. Thus, it is inefficient to suppress the low-density, sparse boxes since it needs to scan over all the grids of the score map even if the grid is void (all zeros). In contrast, our implementation for PSRR-MaxpoolNMS++ here is more friendly for the sparsely distributed boxes because the efficiency is only related to the number of input boxes.

TABLE 3
Comparison between our methods and MaxpoolNMS [8] on the PASCAL VOC dataset [34], at the second stage of Faster R-CNN [3] with ResNet-50 [35]. "Box Overlap" indicates the overlap rate between the outputs by each accelerated NMS method and GreedyNMS. As a reference, the mAP of GreedyNMS is 78.1%.

Methods	Box Overlap (%)	mAP (%)
MaxpoolNMS (Single-Channel)	15.0	33.0
MaxpoolNMS (Cross-Ratio)	18.5	36.6
MaxpoolNMS (Cross-Scale)	11.6	26.5
PSRR-MaxpoolNMS	45.3	77.6
PSRR-MaxpoolNMS++	**48.4**	**78.3**

5 Experiments

5.1 Experimental Setup

Base detectors. We evaluate various NMS approaches at the inference stage of various detection pipelines, including two-stage/one-stage detectors and anchor-based/anchor-free detectors. The used detectors are introduced as follows:

1) Faster R-CNN [3] is a two-stage anchor-based object detector. We use ResNet-50 [35], ResNet-101 [35], P2T-Small [40], and P2T-Large [40] as the backbone network architectures. For Faster R-CNN training, we follow the default training parameters of the public PyTorch [36] implementation[1].
2) SSD [4] is a one-stage anchor-based object detector. We use VGG-16 [41] and MobileNet-v2 [42] as the backbones. We evaluate NMS using the pre-trained models provided by the public PyTorch [36] implementation[2].
3) CenterNet [37] is an anchor-free detector that uses keypoint estimation to find center points and regresses to all other object properties. DLA-34 [38], ResNet-DCN-18 [39], ResNet-DCN-101 [39], and Hourglass-104 [33] are used as the backbones. We use NMS to merge the multi-scale testing results of CenterNet. The pre-trained models provided by the public PyTorch [36] implementation[3] are used for evaluation.
4) TOOD [37] is a one-stage detector that explicitly aligns the object classification and localization in a learning-based manner. We use ResNet-DCN-101 [39] and ResNeXt-101 [43] as the backbones. We evaluate NMS using the pre-trained models provided by the public PyTorch [36] implementation[4].

We simply replace the GreedyNMS in these detectors with the proposed PSRR-MaxpoolNMS or PSRR-MaxpoolNMS++ to show that our NMS methods can be applied to all stages of various types of detectors. For the IoU threshold in GreedyNMS, we use the default parameter settings provided by the official sources.

Datasets. We carry out experiments on three datasets: PASCAL VOC [34], MS-COCO [44], and KITTI [45]. The PASCAL VOC and MS-COCO datasets comprise natural images, while KITTI captures urban scenes. To measure the detection performance, we use the mean average precision (mAP) metric. We train Faster R-CNN [3] on the PASCAL VOC, MS-COCO, and KITTI datasets. For the PASCAL VOC dataset, we train Faster R-CNN using PASCAL VOC 2007 and 2012 `trainval` sets and evaluate on the PASCAL VOC 2007 test set. For the MS-COCO dataset, we train Faster R-CNN using the MS-COCO `train2014` and

1. https://github.com/jwyang/faster-rcnn.pytorch
2. https://github.com/qfgaohao/pytorch-ssd
3. https://github.com/xingyizhou/CenterNet
4. https://github.com/open-mmlab/mmdetection/tree/main/configs/tood

TABLE 4
Detection accuracy (mAP, %) of Faster R-CNN [3] and CenterNet [37] with various backbones on the PASCAL VOC dataset [34]. The number after the symbol "/" for DLA [38] and ResNet-DCN [39] indicates the input resolution for CenterNet [37].

Methods	Faster R-CNN [3]			
	ResNet-50 [35]	ResNet-101 [35]	P2T-Small [40]	P2T-Large [40]
GreedyNMS	78.1	78.4	**81.3**	83.7
PSRR-MaxpoolNMS	77.6	78.0	80.6	83.3
PSRR-MaxpoolNMS++	**78.3**	**79.0**	81.2	**83.8**
Methods	CenterNet [37]			
	DLA-34/512 [38]	ResNet-DCN-18/384 [39]	ResNet-DCN-18/512 [39]	ResNet-DCN-101/512 [39]
GreedyNMS	**79.0**	**74.3**	76.7	**80.0**
PSRR-MaxpoolNMS	77.8	73.4	76.2	78.8
PSRR-MaxpoolNMS++	78.9	74.1	**77.2**	79.7

TABLE 5
Detection accuracy (mAP, %) of SSD [4] with various backbones on the PASCAL VOC dataset [34].

Methods	VGG-16	MobileNet-v2
GreedyNMS	**77.3**	**68.7**
PSRR-MaxpoolNMS	76.1	67.8
PSRR-MaxpoolNMS++	76.7	68.3

`val2014` without the subset of `minival2014` dataset, and the MS-COCO `minival2014` is adopted for evaluation. For the KITTI dataset, we randomly split the dataset into 5611 training images and 1870 testing images and report the mAP at different levels of difficulty. As for other detectors, we utilize pre-trained models and experimental settings from public sources and evaluate our NMS approach during the inference stage.

5.2 Comparison with MaxpoolNMS

We first compare our PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ with MaxpoolNMS [8] at the second stage of Faster R-CNN [3]. The evaluation results on the PASCAL VOC dataset [34] are shown in Table 3. To estimate the quality of mimicking GreedyNMS, given the same set of un-suppressed boxes, we compare the selected box indices from accelerated NMS methods and GreedyNMS, and then the ratio between the number of commonly selected indices and the number of all selected indices (similar to IoU) is taken as the final score, *i.e.*, "Box Overlap" in Table 3. As can be observed from Table 3, all three variants of MaxpoolNMS perform very poorly, *i.e.*, with an over 40% drop in mAP. Although the detection mAP increases with the box overlap, the box overlap between MaxpoolNMS and GreedyNMS is pretty low, *i.e.*, a maximum overlap of 18.5%. This indicates that MaxpoolNMS can be only applied to the first stage of two-stage detectors as it claims [8] and is unsuitable for the final detections. In contrast, our PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ better mimic GreedyNMS, which is evidenced by the large overlap ratio and comparable detection accuracy with GreedyNMS (only 0.5% drop in mAP). It is worth noting that similar to MaxpoolNMS [8], the only parameter to be set for PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ is the overlap threshold θ. Hence, our PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ outperform MaxpoolNMS by a significant margin without adding extra parameter tuning workload. Due to the unsatisfactory performance of MaxpoolNMS [8] as a generic replacement for GreedyNMS, we mainly compare our method with GreedyNMS in our experiments, as shown in §5.3.

5.3 Comparison with GreedyNMS

We continue by comparing our NMS approaches with the default NMS method in current object detectors, *i.e.*, GreedyNMS, on various datasets with various detectors. The experimental results on the PASCAL VOC dataset [34] are summarized in Table 4 and Table 5, where we benchmark two-stage (*i.e.*, Faster R-CNN [3]), one-stage (*i.e.*, SSD [4]), and anchor-free (*i.e.*, CenterNet [37]) object detectors with various backbone networks. As can be seen from Table 4 and Table 5, our PSRR-MaxpoolNMS++ consistently outperforms our preliminary PSRR-MaxpoolNMS in all cases, demonstrating the effectiveness of our new techniques. For anchor-based detectors, our PSRR-MaxpoolNMS++ performs better than GreedyNMS when equipped in Faster R-CNN [3] but performs slightly worse than GreedyNMS for SSD [4]. For the anchor-free CenterNet [37], our PSRR-MaxpoolNMS++ attains comparable performance with GreedyNMS, that is, PSRR-MaxpoolNMS++ achieves the same or better accuracy for some backbones and slightly worse accuracy for other backbones. Note that our PSRR-MaxpoolNMS++ and PSRR-MaxpoolNMS are parallelizable, while GreedyNMS is not, as shown in Table 1.

Table 6 displays the results on the MS-COCO dataset [44], where we benchmark Faster R-CNN [3], CenterNet [37], and TOOD [46] with various backbones. The evaluation is based on the standard metrics from the MS-COCO dataset [44]. For Faster R-CNN [3] and CenterNet [37], our PSRR-MaxpoolNMS++ consistently outperforms GreedyNMS as well as our preliminary PSRR-MaxpoolNMS in all cases. For TOOD [46], our PSRR-MaxpoolNMS++ performs slightly worse than GreedyNMS. Note that our goal in this paper is to develop a fast and parallelizable NMS approach to replace GreedyNMS, not to improve accuracy.

To evaluate the generality of our NMS approaches, we also carry out experiments on the KITTI dataset [45] with urban scenes, and the evaluation results are shown in Table 7. On the KITTI dataset [45], we use Faster R-CNN [3] with various ResNet [35] backbones as the detectors. Our PSRR-MaxpoolNMS++ reaches comparable detection accuracy with GreedyNMS, regardless of the backbone networks used. PSRR-MaxpoolNMS++ outperforms PSRR-MaxpoolNMS in most cases, indicating that it is more suitable to use PSRR-MaxpoolNMS++ in practical applications.

Our experiments demonstrate that our PSRR-MaxpoolNMS++ achieves comparable or even better accuracy than GreedyNMS on various kinds of datasets using various kinds of object detectors. This implies that PSRR-MaxpoolNMS++ is a good replacement for GreedyNMS for object detection. Since PSRR-MaxpoolNMS++ is parallelizable, it would accelerate object detectors by replacing the non-parallelizable GreedyNMS, benefit-

TABLE 6
Detection accuracy of Faster R-CNN [3], CenterNet [37], and TOOD [46] with various backbones on the MS-COCO dataset [44].

Methods	AP @ 0.5:0.95	AP @ 0.5	AP @ 0.75	AP Small	AP Medium	AP Large
MS-COCO `minival2014` with Faster R-CNN [3]						
GreedyNMS (ResNet-50 [35])	33.3	52.8	35.9	14.0	37.8	50.2
PSRR-MaxpoolNMS (ResNet-50 [35])	33.0	51.8	35.7	13.6	37.7	49.9
PSRR-MaxpoolNMS++ (ResNet-50 [35])	**33.6**	**52.9**	**36.3**	**14.1**	**38.2**	**50.6**
GreedyNMS (ResNet-101 [35])	35.1	54.0	37.6	14.7	39.2	53.2
PSRR-MaxpoolNMS (ResNet-101 [35])	34.7	53.1	37.3	14.4	39.2	52.6
PSRR-MaxpoolNMS++ (ResNet-101 [35])	**35.3**	**54.2**	**37.8**	**14.9**	**39.6**	**53.5**
MS-COCO `test2017` with CenterNet [37]						
Methods	AP @ 0.5:0.95	AP @ 0.5	AP @ 0.75	AP Small	AP Medium	AP Large
GreedyNMS (ResNet-DCN-101 [39])	38.3	57.7	40.7	20.1	41.8	54.9
PSRR-MaxpoolNMS (ResNet-DCN-101 [39])	37.8	56.0	40.5	18.5	41.7	55.3
PSRR-MaxpoolNMS++ (ResNet-DCN-101 [39])	**38.8**	**58.2**	**41.3**	**20.3**	**42.5**	**55.7**
GreedyNMS (Hourglass-104 [32], [47])	43.9	62.5	47.9	27.2	46.4	59.1
PSRR-MaxpoolNMS (Hourglass-104 [32], [47])	42.9	60.0	47.0	24.7	46.4	59.4
PSRR-MaxpoolNMS++ (Hourglass-104 [32], [47])	**44.4**	**62.7**	**48.4**	**27.3**	**47.3**	**59.6**
MS-COCO `test2017` with TOOD [46]						
GreedyNMS (ResNet-DCN-101 [39])	**49.3**	**66.8**	53.6	**32.2**	**53.4**	64.0
PSRR-MaxpoolNMS (ResNet-DCN-101 [39])	46.1	62.2	50.2	27.4	50.9	62.4
PSRR-MaxpoolNMS++ (ResNet-DCN-101 [39])	49.2	66.3	**53.7**	31.9	53.3	**64.1**
GreedyNMS (ResNeXt-101-64×4d [43])	**47.6**	**65.8**	**51.6**	**30.6**	**51.4**	**59.7**
PSRR-MaxpoolNMS (ResNeXt-101-64×4d [43])	44.5	61.2	48.3	25.8	49.1	58.3
PSRR-MaxpoolNMS++ (ResNeXt-101-64×4d [43])	47.4	65.4	**51.6**	30.4	**51.4**	**59.7**

TABLE 7
Detection accuracy (mAP, %) of Faster R-CNN [3] with ResNets [35] as the backbone on the KITTI dataset [45].

Methods	mAP (easy to hard)			Car			Pedestrian			Cyclist		
				Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
GreedyNMS (ResNet-50)	**92.3**	**87.3**	80.6	**96.7**	94.5	84.9	**87.7**	**79.4**	**72.4**	92.3	88.1	84.4
PSRR-MaxpoolNMS (ResNet-50)	91.3	86.1	79.4	95.1	94.2	83.5	87.6	77.7	72.1	91.1	86.4	82.7
PSRR-MaxpoolNMS++ (ResNet-50)	91.7	**87.3**	**81.0**	95.1	**95.3**	**86.6**	87.6	78.3	71.9	**92.4**	**88.4**	**84.6**
GreedyNMS (ResNet-101)	91.6	86.2	79.1	**96.2**	93.4	83.9	87.2	**78.9**	**71.6**	91.2	86.2	81.6
PSRR-MaxpoolNMS (ResNet-101)	91.1	85.6	78.7	93.5	93.1	83.7	87.0	77.1	69.7	**92.7**	86.7	**82.7**
PSRR-MaxpoolNMS++ (ResNet-101)	**91.7**	**87.1**	**79.3**	94.8	**96.6**	**85.9**	**88.7**	77.9	69.8	91.5	**86.9**	82.2

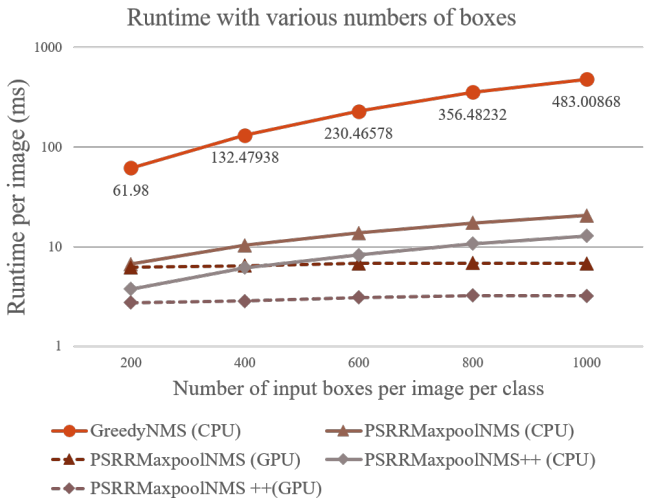


Fig. 10. Execution time of PSRR-MaxpoolNMS, PSRR-MaxpoolNMS++, and GreedyNMS as a function of the number of bounding boxes.

ing the real-world applications of object detection. As an improved version of PSRR-MaxpoolNMS, PSRR-MaxpoolNMS++ achieves its goal of improving the detection accuracy of PSRR-MaxpoolNMS while still being parallelizable.

Discussion. It is somewhat surprising to note that PSRR-MaxpoolNMS++ sometimes attains higher accuracy than GreedyNMS. To explain this, we note that Zheng *et al.* [48] found that both the overlapping extent (*e.g.*, IoU) and distance between two boxes are helpful for bounding box suppression. While GreedyNMS focuses solely on the former by eliminating largely overlapped boxes, PSRR-MaxpoolNMS++ considers both factors. Specifically, the Density-based Discretization in PSRR-MaxpoolNMS++ adopts the target density parameter θ when determining discrete scale centers, and Adjacent Scale Pooling employs the parameter θ when determining the kernel sizes and strides of max-pooling using Eq. (3). Besides, PSRR-MaxpoolNMS++ also considers the distance between two boxes using max-pooling with different kernel sizes and strides for different adjacent scale pairs, as illustrated in Eq. (3).

Limitations. Meanwhile, it is worth noting that our methods might be less effective for less powerful detectors, such as SSD [4], as shown in Table 5. A reasonable explanation is that less powerful detectors may not accurately regress candidate boxes to their corresponding ground-truth locations. Consequently, our

NMS approaches may face difficulty in identifying neighboring boxes that enclose the same instance. In contrast, GreedyNMS can handle this scenario more effectively by calculating the overlap extent to identify neighboring boxes. The impact of this limitation could be mitigated through the ongoing development of more powerful object detectors.

5.4 Efficiency Analyses

We carry out both theoretical and experimental analyses on the computing efficiency of our methods.

5.4.1 Theoretical Analyses of Time Complexity

We first provide the theoretical analysis of the time complexities for GreedyNMS and our methods. Given the number of input boxes N, the time complexities of PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ are both $\mathcal{O}(N)$, which is much smaller than $\mathcal{O}(N \log N) + \mathcal{O}(N^2)$ of GreedyNMS. Moreover, our approaches can be easily parallelized, which in turn would further reduce the execution time.

5.4.2 Runtime Comparison

We proceed with measuring the execution time of GreedyNMS and our methods on an AMD EPYC 7313P CPU and RTX3090 GPU, with different numbers of bounding boxes being processed. We conduct experiments on the MS-COCO `minival2014` dataset [44]. We follow the pipeline to process all boxes in all categories, similar to batched NMS, to implement our NMS methods. Since GreedyNMS is not parallelizable, it is non-straightforward to implement it on GPU. Thus, we only provide GreedyNMS's execution time on CPU. For a fair comparison, we implement PSRR-MaxpoolNMS using the same pooling scheme as PSRR-MaxpoolNMS++ with corresponding key encoding. The results are summarized in Fig. 10. With the increasing number of input bounding boxes, both PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ are more and more efficient than GreedyNMS. PSRR-MaxpoolNMS++ achieves better efficiency than PSRR-MaxpoolNMS. It is interesting to observe that the plot of GreedyNMS appears more efficient than its complexity, *i.e.*, $O(N \log N) + O(N^2)$. According to the computational complexity theory, the complexity is the upper bound in the worst-case scenario. Thus, it is common for theoretical complexity not to align perfectly with actual runtime. Besides, the plots of our methods' GPU implementations look like constant functions. This is because our methods are highly efficient, such that processing up to 1000 boxes cannot make full use of a modern GPU. Thus, the runtime increase is minor when the number of boxes is increased within this range.

5.4.3 Parallelism Analysis

The parallelism inherent in our PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ can leverage the parallelism-friendly GPU platform to further enhance efficiency. As depicted in Fig. 10, GPU implementations generally outperform CPU counterparts in terms of efficiency. To better showcase the advantages of parallelism, we conducted experiments using the batched NMS pipeline on PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ to process large quantities of input bounding boxes, thereby highlighting the algorithms' parallel nature. Fig. 11 illustrates the comparison of execution time on CPU and GPU platforms. It is evident that, owing to the algorithm's parallelism and the powerful parallel capabilities

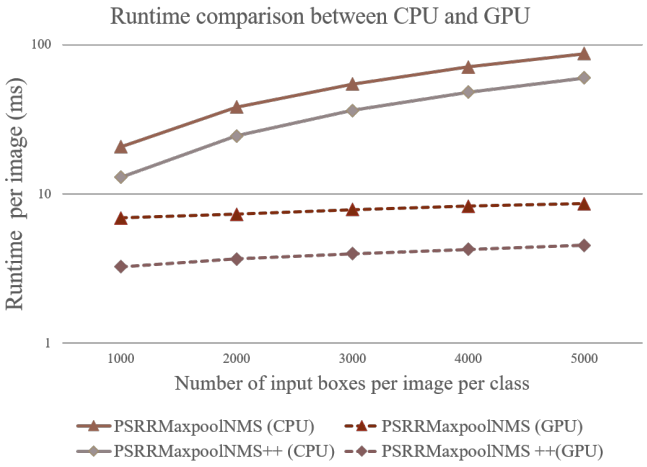


Fig. 11. Execution time of CPU and GPU implementations of PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++ when processing large numbers of bounding boxes.

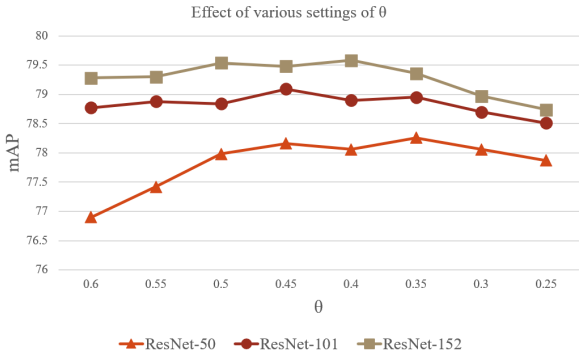


Fig. 12. Evaluation results (mAP, %) on the PASCAL VOC dataset [34] with various settings of the target density parameter θ using the Faster R-CNN detector [3].

of GPU, execution time is significantly reduced compared to the CPU implementation. The speed advantage of our methods on GPU compared to CPU becomes more pronounced as the number of input boxes increases, thus demonstrating the benefits of our methods' parallelism.

5.5 Ablation Study

In this section, we mainly discuss our new designs in the proposed PSRR-MaxpoolNMS++. For more discussions about PSRR-MaxpoolNMS, please refer to our preliminary paper [1].

5.5.1 Density-based Discretization

In this section, we validate the superiority of our new Density-based Discretization in PSRR-MaxpoolNMS++ over the anchor-based discretization in PSRR-MaxpoolNMS. We perform experiments on the PASCAL VOC [34]) and MS-COCO [44] datasets. The experimental results are reported in Table 8. We observe that, compared with anchor-based discretization, our new Density-based Discretization can be adapted to different target suppression densities more flexibly and achieves better detection accuracy. We also conduct additional experiments to investigate the effect of the range of discrete scale centers. In Table 8, the "Anchor-based*" method refers to the anchor-based discretization with an enlarged range of discrete scale centers $\{32^2, 64^2, 128^2, 256^2, 512^2\}$,

TABLE 8
Comparison of different discretization methods in terms of mAP on PASCAL VOC [34] and MS-COCO [44] datasets. "Anchor-based*" refers to the Anchor-based discretization with the same minimum discrete scale center as our density-based discretization.

Datasets	Methods	Anchor-based	Anchor-based*	Density-based
PASCAL VOC [34]	Faster R-CNN [3] (ResNet-50 [35])	78.0	77.9	**78.3**
	Faster R-CNN [3] (ResNet-101 [35])	78.8	78.8	**79.0**
MS-COCO [44]	TOOD [46] (ResNeXt-101 [43])	46.7	47.2	**47.4**
	TOOD [46] (ResNet-DCN-101 [39])	48.5	49.0	**49.2**

TABLE 9
Comparison of different pooling schemes in terms of detection accuracy and execution time on the PASCAL VOC dataset [34]. The detector is Faster R-CNN [3] with the ResNet-50 [35] backbone. SC: Single-Channel MaxpoolNMS; CR: Cross-Ratio MaxpoolNMS; CS: Cross-Scale MaxpoolNMS; All: Cross-all-Channel MaxpoolNMS; ASP: Adjacent Scale Pooling; ARP: Adjacent Ratio Pooling.

Pooling Methods	mAP (ms/image)
SC + CR + CS + All	77.3 (32.9)
ASP	**78.3** (**15.7**)
ARP	58.3 (18.1)
ASP + ARP	77.5 (29.9)

which utilizes the same minimum scale center value and a similar maximum scale center value compared to our Density-based Discretization. Table 8 shows that the enlarged range of anchor-based discrete scale centers may not improve detection performance, whereas the enlarged range of density-based discrete scale centers usually achieves better results.

5.5.2 Adjacent Scale Pooling

In this part, we validate the effect of our Adjacent Scale Pooling with regard to both detection accuracy and efficiency. We first experiment with different pooling approaches proposed in PSRR-MaxpoolNMS and PSRR-MaxpoolNMS++. For a fair comparison, we implement all pooling approaches as in §4.4 with the same parameter settings. The evaluation results are reported in Table 9. Compared with the pyramid pooling (*i.e.*, Single-Channel + Cross-Scale + Cross-Ratio + Cross-all-Channels MaxpoolNMS) in PSRR-MaxpoolNMS, the Adjacent Scale Pooling in PSRR-MaxpoolNMS++ archives better detection accuracy with less runtime because PSRR-MaxpoolNMS++ has fewer rounds of scanning than the pyramid pooling in PSRR-MaxpoolNMS.

Besides, we can observe in Fig. 9 that although the largely-overlapped box pairs exist across the discrete ratios, the chance for largely-overlapped box pairs with dramatic ratio difference (*e.g.*, 0.5 *vs.* 2) is small. Thus, it is intuitive to further investigate the pooling across the adjacent ratios. Similar to Adjacent Scale Pooling, adjacent ratio pooling refers to pooling across adjacent discrete ratio centers regardless of the scale centers. As shown in Table 9, the adjacent ratio pooling underperforms our Adjacent Scale Pooling significantly. The underlying reason may be that the large overlapping could not happen between a pair of boxes with a dramatic scale difference (see Fig. 8). We also carry out experiments by simultaneously performing the adjacent scale and adjacent ratio pooling schemes. That is to say, the pooling is operated across the boxes assigned to both the neighboring discrete scale and ratio centers. As in Table 9, under the same settings, the simultaneous adjacent scale/ratio pooling does not bring an improvement in detection accuracy but brings an increase in runtime. Therefore, we choose to perform the Adjacent Scale Pooling and disregard the factor of ratios.

TABLE 10
Results (mAP, %) of extending our methods to SoftNMS on the PASCAL VOC dataset [34] with the Faster R-CNN detector [3].

	ResNet-50 [35]	ResNet-101 [35]
SoftNMS [9]	**78.7**	79.2
Ours-Soft-Anchor	78.6	79.3
Ours-Soft-Density	**78.7**	**79.4**

5.5.3 Target Density Parameter

The target density parameter θ is the parameter to decide the overlapping extent of the suppressed boxes. In PSRR-MaxpoolNMS++, it decides the discrete center resolution and spatial pooling kernel size, which directly influences the suppression precision and recall. We perform experiments with various θ values using the Faster R-CNN detector [3] with various backbones. The results on the PASCAL VOC dataset [34] in terms of mAP are depicted in Fig. 12. As can be seen, the optimal θ value is in the range of [0.35, 0.45]. We also observe that more powerful detectors (*e.g.*, with ResNet-101 or ResNet-152 backbones [35]) will result in less sensitivity on the target density parameter θ.

5.6 Extension to SoftNMS

We continue by validating that our approach can be extended to SoftNMS [9]. We conduct experiments using the Faster R-CNN [3] detector with various backbones. The local range for box suppression is defined the same as the Adjacent Scale Pooling with anchor-based (*i.e.*, "Ours-Soft-Anchor") or density-based (*i.e.*, "Ours-Soft-Density") discretization approach. The detection accuracy is reported in Table 10. We observe that our approach achieves comparable detection accuracy (mAP) with the SoftNMS baseline using various backbones, suggesting the good generality of our approach. Besides, the Density-based Discretization approach also helps to improve detection accuracy since it defines a more reasonable local range of suppression.

6 Conclusion

This paper targets accelerating non-maximum suppression (NMS) for deep-learning-based object detection. Existing NMS approaches are either non-parallelizable (*e.g.*, GreedyNMS) or difficult to be applied to all stages of object detectors (*i.e.*, MaxpoolNMS [8]). To address this challenge, we noted that the parallelizable NMS approaches can be divided into two stages: *box coordinate discretization* and *local score argmax calculation* (§1). We first introduced our preliminary NMS approach, **PSRR-MaxpoolNMS**, which applies anchor-based discretization, *i.e.*, *Relationship Recovery*, and *Pyramid Shifted MaxpoolNMS* to improve the above two stages, respectively. PSRR-MaxpoolNMS extends the parallelizable NMS approach to all stages of all detectors (§3). Then, we further proposed **PSRR-MaxpoolNMS++** to improve the accuracy and computational efficiency, which consists of *Density-based Discretization* and *Adjacent Scale Pooling* for handling the above two stages, respectively (§4). We conducted comprehensive experiments on various datasets using various object detectors (§5). The results show that our approaches outperform parallelizable MaxpoolNMS [8] significantly. Notably, our

PSRR-MaxpoolNMS++ achieves comparable detection accuracy with much better efficiency than non-parallelizable GreedyNMS. In the future, we plan to explore the implementation of our NMS approaches on more platforms (*e.g.*, ASIC) for facilitating the real-world deployment of object detectors.

REFERENCES

[1] T. Zhang, J. Lin, P. Hu, B. Zhao, and M. M. S. Aly, "PSRR-MaxpoolNMS: Pyramid shifted MaxpoolNMS with relationship recovery," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 840–15 848. 1, 2, 12

[2] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448. 1, 2

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99. 1, 2, 3, 4, 8, 9, 10, 11, 12, 13

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision*, 2016, pp. 21–37. 1, 2, 4, 9, 10, 11

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788. 1, 2

[6] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387. 1

[7] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-Datacenter performance analysis of a tensor processing unit," in *Annual International Symposium on Computer Architecture*, 2017, pp. 1–12. 1

[8] L. Cai, B. Zhao, Z. Wang, J. Lin, C. S. Foo, M. S. Aly, and V. Chandrasekhar, "MaxpoolNMS: Getting rid of NMS bottlenecks in two-stage object detectors," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9356–9364. 1, 2, 3, 4, 5, 6, 9, 10, 13

[9] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS – Improving object detection with one line of code," in *IEEE International Conference on Computer Vision*, 2017, pp. 5561–5569. 2, 3, 6, 8, 13

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969. 2

[11] Z.-M. Chen, X. Jin, B. Zhao, X.-S. Wei, and Y. Guo, "Hierarchical context embedding for region-based object detection," in *European Conference on Computer Vision*, 2020, pp. 633–648. 2

[12] C.-D. Xu, X.-R. Zhao, X. Jin, and X.-S. Wei, "Exploring categorical regularization for domain adaptive object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 724–11 733. 2

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

[14] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. 2

[15] Y. Liu, S. Li, and M.-M. Cheng, "RefinedBox: Refining for fewer and high-quality object proposals," *Neurocomputing*, vol. 406, pp. 106–116, 2020. 2

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893. 2

[17] S. Liu, D. Huang, and Y. Wang, "Adaptive NMS: Refining pedestrian detection in a crowd," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6459–6468. 3

[18] J. Hosang, R. Benenson, and B. Schiele, "A convnet for non-maximum suppression," in *German Conference on Pattern Recognition*, 2016, pp. 192–204. 3

[19] N. Gählert, N. Hanselmann, U. Franke, and J. Denzler, "Visibility Guided NMS: Efficient boosting of amodal object detection in crowded traffic scenes," *arXiv preprint arXiv:2006.08547*, 2020. 3

[20] N. O. Salscheider, "FeatureNMS: Non-maximum suppression by learning feature embeddings," in *International Conference on Pattern Recognition*, 2021, pp. 7848–7854. 3

[21] J. Wang, X. Yin, L. Wang, and L. Zhang, "Hashing-based non-maximum suppression for crowded object detection," *arXiv preprint arXiv:2005.11426*, 2020. 3

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020, pp. 213–229. 3

[23] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations*, 2021. 3

[24] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "DN-DETR: Accelerate DETR training by introducing query denoising," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 619–13 627. 3

[25] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. Ni, and H. Shum, "DINO: DETR with improved denoising anchor boxes for end-to-end object detection," in *International Conference on Learning Representations*, 2023. 3

[26] Q. Chen, X. Chen, J. Wang, S. Zhang, K. Yao, H. Feng, J. Han, E. Ding, G. Zeng, and J. Wang, "Group DETR: Fast DETR training with group-wise one-to-many assignment," in *IEEE International Conference on Computer Vision*, 2023, pp. 6633–6642. 3

[27] Z. Zong, G. Song, and Y. Liu, "DETRs with collaborative hybrid assignments training," in *IEEE International Conference on Computer Vision*, 2023, pp. 6748–6758. 3

[28] Q. Hong, F. Liu, D. Li, J. Liu, L. Tian, and Y. Shan, "Dynamic sparse R-CNN," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4723–4732. 3

[29] J. Wang, L. Song, Z. Li, H. Sun, J. Sun, and N. Zheng, "End-to-end object detection with fully convolutional network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 849–15 858. 3

[30] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, "RTMDet: An empirical study of designing real-time object detectors," *arXiv preprint arXiv:2212.07784*, 2022. 3

[31] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475. 3

[32] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *European Conference on Computer Vision*, 2018, pp. 734–750. 5, 11

[33] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *IEEE International Conference on Computer Vision*, 2019, pp. 6569–6578. 5, 9

[34] M. Everingham, L. Van Gool, C. K. 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, 9, 10, 12, 13

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 8, 9, 10, 11, 13

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Annual Conference on Neural Information Processing Systems*, 2019, pp. 8026–8037. 9

[37] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019. 9, 10, 11

[38] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2403–2412. 9, 10

[39] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 764–773. 9, 10, 11, 13

[40] Y.-H. Wu, Y. Liu, X. Zhan, and M.-M. Cheng, "P2T: Pyramid pooling transformer for scene understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12 760–12 771, 2022. 9, 10

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. 9

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520. 9

[43] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500. 9, 11, 13

[44] 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 *European Conference on Computer Vision*, 2014, pp. 740–755. 9, 10, 11, 12, 13

[45] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361. 9, 10, 11

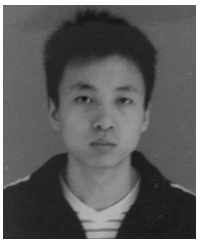
[46] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "TOOD: Task-aligned one-stage object detection," in *IEEE International Conference on Computer Vision*, 2021, pp. 3490–3499. 10, 11, 13

[47] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*, 2016, pp. 483–499. 11

[48] 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 *AAAI Conference on Artificial Intelligence*, 2020, pp. 12 993–13 000. 11

Tianyi Zhang received the B.E. degree from Beihang University, China, in 2012, the Master's degree from University of Pennsylvania, USA, in 2014, and the Ph.D. degree degree from Nanyang Technological University, Singapore, in 2020. She is currently an Assistant Professor at School of Cyber Science and Technology, Beihang University, China. Her research interests include computer vision and deep learning, with emphasis on image/video recognition and weakly supervised learning.

Chunyun Chen received the B.E. degree from University of Electronic Science and Technology of China, Chengdu, Sichuan, China, in 2020, and the Ph.D. degree in the College of Computing and Data Science, Nanyang Technological University, Singapore, in 2024. His research interests include intelligent learning machine systems, deep learning processor architecture, transformer accelerator architecture, network-on-chip, and distributed systems.



Yun Liu received his B.E. and Ph.D. degrees from Nankai University in 2016 and 2020, respectively. Then, he worked with Prof. Luc Van Gool as a postdoctoral scholar at Computer Vision Lab, ETH Zurich, Switzerland. After that, he worked as a senior scientist at the Institute for Infocomm Research (I2R), A*STAR, Singapore. His research interests include computer vision and machine learning.

Xue Geng received her B.E. degree in computer Science from Northeastern University, China, in 2012, and Ph.D. degree in computer science from NUS, Singapore, in 2017. Currently, she is a senior scientist at the Institute for Infocomm Research (I2R), A*STAR, Singapore. Her research interests include model compression and efficient machine learning.

Mohamed M. Sabry Aly is an Associate Professor at Nanyang Technological University, Singapore, and the founder of EMASS. He received his Ph.D. degree in electrical and computer engineering from École Polytechnique Fédérale de Lausanne (EPFL), in 2013. He was a postdoctoral research fellow at Stanford University from 2014 till 2017. His current research interests include system-level design and optimization of computing systems enabled by emerging technologies, with particular emphasis on computing systems for artificial intelligence. He is an active close collaborator with top industrial and academic partners such as, Stanford University and TSMC. He is a Senior IEEE member, and he was the recipient of the Swiss National Science Foundation Early Post-Doctoral Mobility Fellowship in 2013.

Jie Lin was a senior scientist at the Institute for Infocomm Research (I2R), A*STAR, Singapore. His research interests include deep learning, AI accelerators, privacy-preserving machine learning, data compression, and computer vision. In particular, he worked on resource-efficient AI to build lightweight, fast, and energy-efficient deep learning algorithms for next-generation AI hardware and privacy-preserving applications. He published over 80 peer-reviewed papers on top conferences and journals.