remote sensing

MDPI

Article

DPNMS: A Density-Peak-Based Non-Maximum Suppression Algorithm for High-Quality Oriented Object Detection

Zhonghua Li [1,2,*], Bin Xiao [1,2] and Zhanghai Wu [1]

[1] School of Artificial Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; xiaobin@cqupt.edu.cn (B.X.); s250231160@stu.cqupt.edu.cn (Z.W.)
[2] Academy of Advanced Interdisciplinary Studies, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
* Correspondence: lizhonghua@cqupt.edu.cn;

Highlights

What is the main finding?

- We introduced density-peak clustering into the object detection task and designed a novel post-processing method. This method estimates density based on the IoU and confidence scores between candidate bounding boxes and gradually refines the detection results through a select–correct loop. It achieves fast and high-precision post-processing for remote sensing object detection.

What is the implication of the main finding?

- The proposed method introduces a novel post-processing architecture that considers all candidate samples within an object's neighborhood. Furthermore, the method does not depend on sorting operations, allows for parallel acceleration, can deploy on embedded devices, and balances accuracy and latency.

Abstract

Aerial imagery objects present characteristics such as arbitrary orientations, dense distribution, and a wide range of scales. Mainstream detectors add angles to traditional rectangular bounding boxes to better fit the object boundaries, but this makes the detection task even more challenging. Non-Maximum Suppression (NMS) is a filter designed to address duplicate detections in dense detectors, extracting the most representative output from multiple detection results for a single object. However, the NMS algorithm has fundamental problems, including the risk of wrong suppression due to confidence-based prioritization, poor computational speed from sequential processing, and inefficient use of its output data. To address the above challenges, and based on the data patterns of objects in remote sensing imagery, we reviewed NMS as a local density peak clustering task and proposed a novel post-processing method called DPNMS. We integrated information across each output through statistical analysis to construct its density representations, and then searched for peak values within valid neighborhoods defined by the IoU threshold. To prevent suppressing effective boxes, we designed a "select–release" cyclic workflow to correct the chosen result. The proposed method integrates the detector's output information to achieve competitive results. Our method does not require sorting the localization information in reverse order of confidence, and each operation is independent of the others, allowing for parallelization and resulting in high computational efficiency. Extensive experiments conducted on various datasets and multiple platforms have validated the effectiveness and stability of the proposed method. The unique filtering strategy offers a new framework for post-processing methods in remote sensing image object detection.

Check for updates

Academic Editor: Salah Bourennane

Received: 14 September 2026
Revised: 2 November 2026
Accepted: 14 November 2026
Published: 18 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: oriented object detection; post-processing; density peak cluster; non-maximum suppression; remote sensing

1. Introduction

In the field of remote sensing image interpretation, object detection is a fundamental visual task, which is critical to the interpretation quality. Different from natural scenes, objects in remote sensing imagery typically show characteristics such as dense distribution, random orientation, and wide scale ranges. Therefore, the horizontal bounding boxes (HBB) used in standard object detection [1–6] tasks are no longer suitable for object detection in remote sensing scenarios. Recently, many researchers have conducted in-depth and detailed studies in this subfield, including mainstream methods based on rotating bounding box representations [7–10], Gaussian-based localization representations [11–16], point-set-based localization methods [17–19], polar coordinate representations [20], and vector representations [21]. These works have significantly improved the baseline performance of remote sensing object detectors. However, CNN-based dense object detectors [7,8,10,22,23] are limited by the non-maximum suppression process (NMS), which leads to performance challenges in implementation. Although many studies in recent years have attempted to eliminate NMS to achieve true end-to-end detection [3,24–28], there remains a performance gap between them. Considering deployment and computational efficiency, dense object detectors are still the mainstream industrial detection algorithms today.

To improve training efficiency, object detectors usually generate multiple valid positive samples for a single target, enabling the model to learn this "one-to-many" pattern; consequently, during prediction, the model will output multiple similar bounding boxes for a single target. The NMS selects the "optimal" one from multiple similar localization boxes and suppresses the remaining results. Traditional NMS requires sorting by confidence, then calculating IoU for each instance and applying suppression based on the threshold. Since each inhibition step depends on the preceding ones, it is difficult to accelerate the process using parallel computing [29–31]. Current mainstream detectors have an extremely low inference latency, typically producing an output within 10 ms. However, the NMS algorithm is usually executed by the CPU, which is significantly slower, thereby reducing the detector's overall operational efficiency. Additionally, the NMS method uses a confidence-first strategy to suppress detection bounding boxes. It means that if the boundary precision and confidence of the model's output do not align, high-quality detection boxes may be suppressed. Accordingly, traditional NMS cannot achieve a globally optimal solution from a holistic perspective, often only achieving suboptimal accuracy. This is a limitation of the NMS algorithm, which is the primary focus of this paper.

Motivation: Although remote sensing image targets present challenges such as dense distribution and arbitrary orientation, the areas of overlap between similar objects are small. As a result, the detector usually produces highly concentrated output for remote sensing images. In remote sensing scenarios, NMS can be treated as clustering algorithm, with each cluster centre representing an object. The density peak algorithm (DPC) [32] utilizes the concentration of samples for clustering, and the distribution pattern of remote sensing objects aligns well with this basic requirement. The DPC [32] calculates the density of all sample points based on distance and determines the peak point among them as the cluster center. The dense object detector outputs multiple bounding boxes distributed across the target region. These boxes are scattered around the object, and the optimal one is usually located at the center of the entire distribution. Based on this observation, we developed a strategy that uses the density of the bounding boxes to select and suppress

candidates. This achieves detection results that better align with the object boundaries. First, we calculated the intersection over union (IoU) between candidate boxes. Then, we combined this with the specified threshold and confidence to determine the density of all boxes. The candidate boxes with the local peaks are the results we need, but a single filtering operation is easily prone to missing some valid boxes. Finally, we designed a "select–release" error-correction loop mechanism to obtain the final results with iteration. Because fewer iterations are required, the vast majority of the proposed method's computational load can be accelerated through parallel computing on devices such as GPUs. Only a small amount of control code relies on the CPU. Consequently, our method achieves a significant speed improvement over traditional NMS and makes the deployment of the model at the edge a more realistic possibility.

Figure 1 illustrates the basic principle of the proposed method. We regard the IoU value as the distance between candidate boxes. Other candidate boxes within a specified threshold range are counted and calculated to determine the density. Candidate boxes are distributed around the target. The optimal box has a high IoU value with other candidate boxes, whereas boxes farther from the target have lower IoU values. By analyzing local information, we obtain localization boundaries that better align with the target.

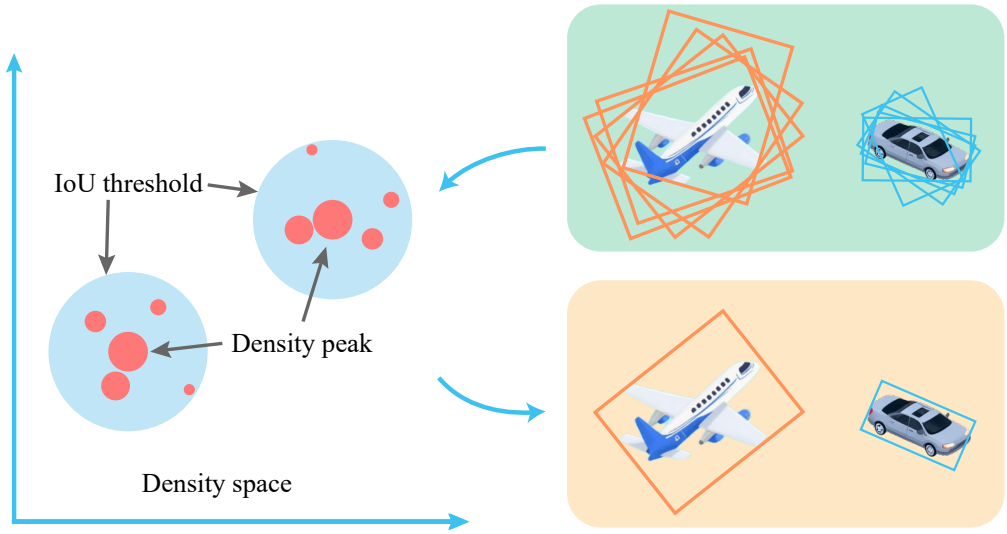


Figure 1. Basic principle of DPNMS. Dense detectors output numerous candidate boxes for each object, and the optimal boxes are usually close to others. The density of surrounding boxes is determined by their concentration, and the optimal result can be identified by selecting the boxes with the highest density.

2. Related Works

2.1. The Relationship Between Object Detection and NMS

Classic dense object detection algorithms typically assign multiple positive samples to the same geometric region of an object, thereby improving the model's convergence speed and learning performance. This pattern of allocation is known as "one-to-many" (O2M) assignment [8,10,22,23,33,34]. Conversely, there is also a pattern whereby only one positive sample is assigned to a target. This is known as "one-to-one" (O2O) assignment [3,24–27,35,36]. When the model is trained with the O2M assignment pattern, numerous redundant detections are produced during inference. To eliminate this redundancy and select the most representative results, the NMS algorithm must be applied. Traditional NMS suffers from numerous issues, such as reliance on the alignment between confidence scores and localization boundaries, low computational efficiency due to serial processing, and overly simplistic suppression criteria that compromise boundary precision.

Consequently, to bypass NMS post-processing operations, significant research has focused on O2O strategies. The O2O training strategy enables the model to perform end-to-end detection, but it usually leads to poor performance and slow convergence. For dense object detectors, O2O sample allocation causes a more severe imbalance between positive and negative samples during training, increasing the risk of failure to converge. The DETR series of models [3,27,35,36] uses the Hungarian matching mechanism in O2O training mode to achieve end-to-end detection. Many variants of fast detection have been proposed, but deploying Transformer-based detectors on edge devices remains challenging. Some dense detectors have used the O2M module to assist in training the O2O branches. YOLO [25,26], a leading industrial-grade object detection algorithm, has supported end-to-end detection since the 10th [25] generation. However, the NMS-based and NMS-free (end-to-end) branches of YOLOv26 [26] still differ in precision.

2.2. Post-Processing Method for Object Detection

Standard object detection models can be directly categorized into two types based on NMS: NMS-based and NMS-free. NMS-based models [7,8,10,22,23,34] produce multiple dense detection results for each object and require NMS to select the optimal bounding box. In contrast, NMS-free detectors [3,5,24–27,35–37] produce a single result per object and therefore do not require NMS. Although NMS-based methods are usually more precise than NMS-free, they still face several challenges. Figure 2 lists the core issues of NMS. These can be summarized into three main points. (**a**) There is a risk that valid detections in near locations may be suppressed. (**b**) Current operations affect subsequent processes, preventing parallelization and increasing detection latency. (**c**) Suppressed localization information is discarded outright and not utilized. Therefore, improvements to the NMS primarily focus on the three parts listed above.

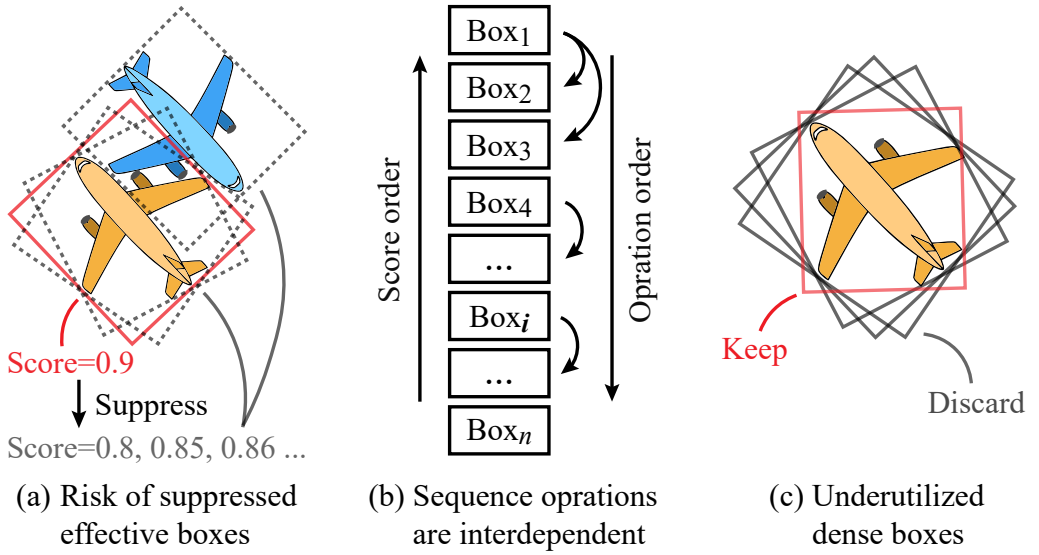


Figure 2. Key challenges for the NMS method. First, the detector's output cannot align quality and confidence, which introduces the risk that high-quality candidate boxes may be suppressed due to low confidence. Second, each step in a standard NMS affects subsequent operations, making it difficult to process them in parallel and leading to slower computation speeds. Third, standard NMS prioritizes confidence levels when determining results, without considering the distribution of candidate boxes, and thus underutilizes them.

The NMS confidence-based strategy and hard-threshold risk filter out valid detections with high overlap. To address this, soft-NMS [38] replaces hard-threshold suppression with a gradual reduction in the confidence scores of nearby candidate boxes, thereby

retaining more valid detections. Some methods focus on different IoU calculation methods, such as GIoU [39], CIoU [30], DIoU [40], SIoU [41], and ProbIoU [14]. The traditional IoU loss has zero gradients when there is no overlap between two object boxes, making it impossible to optimize. Therefore, GIoU [39] introduces a penalty term based on the smallest rectangle that encompasses both boxes, enabling non-overlapping predictions to optimize toward valid objects. DIoU [40] incorporates the normalized distance between the centers, achieving faster convergence than GIoU [39]. CIoU [30] enhances DIoU by considering parameters such as overlapping area, center-to-center distance, and the aspect ratio of the bounding boxes. SIoU [41] further incorporates the vector angle between the prediction and the ground truth as a penalty term. This reduces the degrees of freedom in the regression, significantly accelerating convergence and improving detection accuracy. Unlike the above methods, ProbIoU [14] is an IoU calculation method specifically designed for rotated objects. It converts the rotated rectangular bounding box into a corresponding Gaussian distribution, then calculates the Hellinger distance between the two Gaussian distributions as the IoU. This approach provides good protection for objects such as aircraft in remote sensing scenes with only slight overlap at the edges and corners.

Because of the sequential dependencies among suppression operations in the NMS algorithm, its execution speed cannot be improved through parallel computing, thereby limiting the development of dense object detection algorithms. Thus, many works have focused on improving computational efficiency (including work-efficient approaches). H-NMS [42] uses hash processing to filter out many irrelevant detection boxes quickly, significantly reducing the number of NMS candidate boxes and improving computational speed. However, the coarse classification introduced by hashing can lead to significant errors and reduce precision. Fast-NMS [29] calculates the IoU between all candidate boxes in a single step and constructs an upper-triangular matrix, which is then used to perform batch suppression directly based on a threshold. It significantly speeds up post-processing methods through parallel computation. However, its single-pass suppression operation risks overlooking many valid detections, thereby reducing precision. Cluster-NMS [30] is based on Fast-NMS [29] and involves multiple iterations of batch suppression and adjustments. This ultimately achieves the same result as traditional NMS. Its processing speed is slower than Fast-NMS, but it remains highly efficient compared to traditional serial NMS. Si et al. [31] viewed the problem from graph theory and employed the divide-and-conquer strategy to split numerous candidate boxes into mutually independent subgraphs, then applied NMS to each subgraph independently, thereby enabling rapid parallel processing.

Some approaches no longer focus on individual candidate boxes but instead treat them as a whole and integrate them through clustering. NMW [43] and WBF [44] cluster the prediction results, then perform a weighted sum over multiple candidate boxes, merging them into a single result. They group candidate boxes into multiple clusters based on IoU and confidence scores. They keep the IoU as high as possible within each cluster and as low as possible between clusters. Then, identify the center of each cluster as the result. They are all designed for general object detection tasks and do not consider object features in remote sensing scenarios. Therefore, their performance is poor when applied directly to remote sensing image data. To address the issue of poor synthesis results caused by OBB encoding in remote sensing scenarios, GauS [45] has developed a synthesis and decoding method based on the Gaussian distribution. This improves the stability of the synthesis results and further enhances detection precision.

3. Methods

In remote sensing scenarios, intersection areas between same-type objects are typically small, meaning that the predictions of dense detectors are more concentrated than in natural

scenarios. The detector distributes candidate boxes around the true object boundary, and we believe a candidate box closer to the center of the overall distribution lies closer to the real object boundary. Therefore, NMS can be viewed as searching for the central box within the distribution of candidate boxes. The density peak clustering algorithm constructs density by a statistical analysis of the distance attributes of neighboring samples, thereby partitioning samples into clusters organized around density peak samples, which aligns with the task requirements. Accordingly, we reformulate NMS as a density peak clustering task, where each density peak sample represents a filtered detection result. We have divided the entire task into two core components: density calculation and the select–correct loop. Figure 3 illustrates the overall workflow of our method. The pseudocode is shown in Algorithm 1.

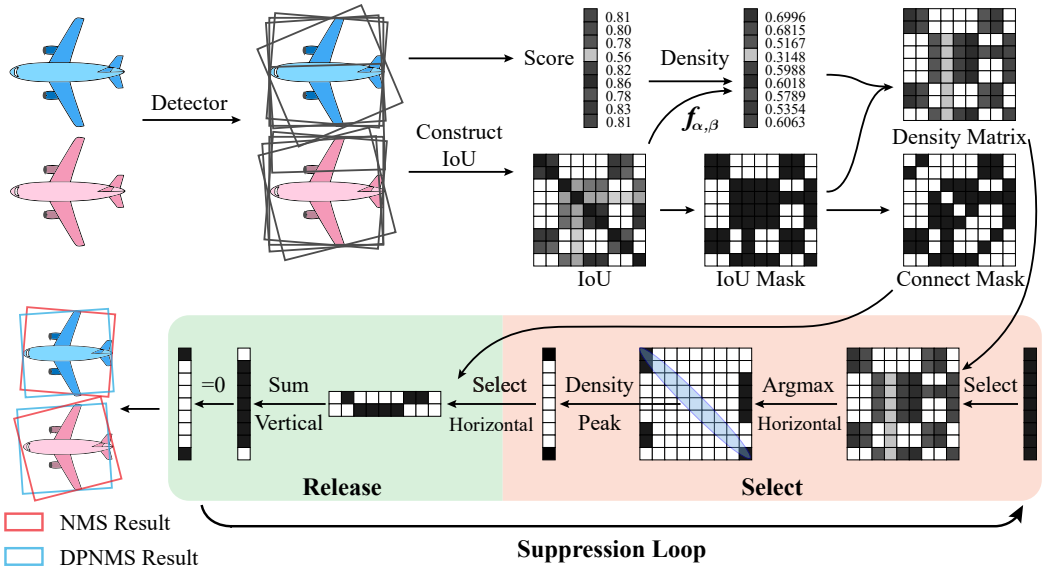


Figure 3. The core framework of the DPNMS algorithm. The algorithm comprises two parts: density construction and peak filtering loops. First, calculate the IoU matrix for the candidate boxes and construct a density matrix based on their confidence scores. Next, identify the maximum density within each IoU-associated cluster and release any candidate boxes that were incorrectly suppressed. Repeat this process until the result no longer changes, then generate the output.

For the density construction component, we believe a candidate box's density should be determined jointly by the distances and confidence scores of the surrounding boxes. Distance is an important metric in peak density algorithms and is typically used to describe the closeness or similarity between samples. For object detection task, IoU is a metric that can express the closeness between boxes. Since IoU is a normalized value, it eliminates the influence of box scale. Therefore, we define IoU as a measure of the "distance" between two bounding boxes in our method. If the candidate bounding box has small IoU with the other boxes, we can consider it as far from the object. Conversely, we regard the box as close to the object. Therefore, our method defines density as a measure of how closely samples cluster around the current bounding box. It reflects the distance between the candidate box and the true boundary. Building density based on IoU or confidence scores fails to leverage detection information, resulting in lower overall precision. Therefore, we combine IoU and confidence scores to calculate the density value for each bounding box. For detector outputs with N candidate boxes, the density is defined by Equation (1).

$$d_j = \frac{1}{k}\sum_{i}^{k} {s_i}^{\alpha}\mathrm{IoU}\left(B_j, B_i\right)^{\beta} \left| IoU_{ji} > \tau \right. \quad (1)$$

where τ means IoU threshold, k means the number of detection boxes within τ, and s means the confidence score of candidate boxes. α and β indicate the sensitivity of confidence and IoU, respectively. DPC typically uses the cumulative number of samples in a neighborhood as a measure of density. The density threshold directly affects the number of outliers. For object detection tasks, it poses a risk of missing detections and reduces recall. Therefore, the mean function is more suitable for object detection tasks.

Algorithm 1: DPNMS: Density Peak Non-Maximum Suppression

Input: Candidate boxes $\mathcal{B} = \{b_1, \ldots, b_N\}$; scores $\mathcal{S} = \{s_1, \ldots, s_N\}$; labels $\mathcal{L} = \{l_1, \ldots, l_N\}$; IoU threshold τ; exponents α, β; max iterations T; image max edge: Len

Output: Set of kept indices $\mathcal{K}$

```
// Step 1:  construct the density matrix
```

for $i = 1$ **to** N **do**

  $b_i.cx \leftarrow b_i.cx + Len\, l_i$;

  $b_i.cy \leftarrow b_i.cy + Len\, l_i$;

end

$I \leftarrow \mathrm{RotatedIoU}(\mathcal{B}) \in \mathbb{R}^{N \times N}$;

$M \leftarrow I > \tau$;

$\mathcal{D} \leftarrow M \odot I^{\beta} \odot (\mathcal{S}^{\alpha})^{\top}$;

for $i = 1$ **to** N **do**

  $\rho_i \leftarrow \dfrac{\sum_j \mathcal{D}_{ij}}{\sum_j M_{ij}}$;

end

$G \leftarrow \boldsymbol{\rho}^{\top} \odot M$;

$\hat{M} \leftarrow M - \mathrm{diag}(M)$;

```
// Step 2:  peak select-correct loop
```

$\mathcal{R} \leftarrow \{1, \ldots, N\}$, $\mathcal{K} \leftarrow \emptyset$, $G^{\mathrm{rem}} \leftarrow G$;

for $t = 1$ **to** T **do**

  $\boldsymbol{\pi} \leftarrow \arg\max_j \; G^{\mathrm{rem}}_{:,j}$;

  $\mathcal{K}_t \leftarrow \{\, i \in \mathcal{R} : \pi_i = i \,\}$; // self-pointing boxes are peaks

  if $\mathcal{K}_t = \emptyset$ **or** $\mathcal{K}_t = \mathcal{K}$ **then**

    Break

  end

  $\mathcal{K} \leftarrow \mathcal{K}_t$;

  for $j \in \mathcal{R}$ **do**

    if $\sum_{i \in \mathcal{K}} \hat{M}_{ij} > 0$ **then**

      $\mathcal{R} \leftarrow \mathcal{R} \setminus \{j\}$; // release incorrectly suppressed boxes

    end

  end

  $G^{\mathrm{rem}} \leftarrow G^{\mathrm{rem}}[\mathcal{R}, \mathcal{R}]$;

end

return $\mathcal{K}$;

For each candidate box, we need to calculate the IoU with all surrounding boxes, allowing us to process them once and construct the IoU matrix. The IoU computations between different boxes are independent. We rapidly constructed the IoU matrix Mat_{iou} through parallel computation. It is an N-th order symmetric matrix, and all elements on its

diagonal are 1. The Mat_{iou} is clipped at τ to yield mask $Mask_{iou}$. Thus, the density vector can be calculated using the following formula.

$$D = \frac{\text{SUM}_h\left(Mask_{iou} \cdot \text{expand}_v(score^{\alpha}) \cdot Mat_{iou}^{\beta}\right)}{\text{SUM}_h(Mask_{iou})} \tag{2}$$

where SUM_h indicates horizontal summation and returns an N-dimensional vector. Symbol "$\cdot$" means element-wise matrix product. Function expand_v means expanding a vector into a square matrix in the vertical direction.

Expand vector D into an $N \times N$ square matrix using the broadcast mechanism, along the vertical direction. Then apply $Mask_{iou}$ to filter out invalid values, yielding the density matrix $\mathcal{D}$.

$$\mathcal{D} = \text{expand}_v(D|_{1\times \text{N}}) \cdot Mask_{iou} \tag{3}$$

By filtering through the mask, we extracted the density of samples within the neighborhood and located the maximum value among them. Therefore, most samples in a group share the same peak. Since each sample's associated localization box differs, we only recognize the peak samples on the main diagonal as valid. Single-pass peak filtering risks missing some results. For example, if box A suppresses box B, and box C is suppressed by B but not by A, peak filtering will directly suppress B and C, where C is the box that is incorrectly suppressed. Therefore, we cannot rely on a single peak filtering operation to obtain complete results. Inspired by Cluster-NMS [30], we refined the results iteratively within the loop through multiple iterations. The core idea is to identify candidate boxes that are not associated with the density-peak box, meaning their IoU is below the threshold τ. We call this the "select–release" or "select–correct"' loop. Algorithm 1 illustrates the pseudocode for the select–release loop. We keep the filtered peaks and the boxes that are not associated with any peaks, then perform peak filtering again. After several iterations, when remaining boxes no longer change, we obtain the results.

4. Experiments

4.1. Datasets

DOTA [46] is a large-scale dataset for aerial object detection. The data are collected from different sensors and platforms. DOTA-v1.0 contains 2806 large aerial images with size ranges from 800 × 800 to 4000 × 4000 and 188,282 instances among 15 common categories: Plane (**PL**), Baseball diamond (**BD**), Bridge (**BR**), Ground track field (**GTF**), Small vehicle (**SV**), Large vehicle (**LV**), Ship (**SH**), Tennis court (**TC**), Basketball court (**BC**), Storage tank (**ST**), Soccer-ball field (**SBF**), Roundabout (**RA**), Harbor (**HA**), Swimming pool (**SP**) and Helicopter (**HC**).

DIOR-R[47] is a large-scale benchmark dataset for object detection in optical remote sensing images, which consists of 23,463 images and 192,518 object instances annotated with oriented bounding boxes. It covers a wide range of scenes and 20 common object classes. The categories of objects in DIOR-R include Airplane (**PL**), Airport (**APO**), Baseball Field (**BF**), Basketball Court (**BC**), Bridge (**BR**), Chimney (**CH**), Expressway Service Area (**ESA**), Expressway Toll Station (**ETS**), Dam (**DAM**), Golf Field (**GF**), Ground Track Field (**GTF**), Harbor (**HA**), Overpass (**OP**), Ship (**SH**), Stadium (**STA**), Storage Tank (**ST**), Tennis Court (**TC**), Train Station (**TS**), Vehicle (**VE**) and Windmill (**WM**).

FAIR1M[48] is a large-scale dataset for fine-grained object detection and recognition in Remote sensing images. The FAIR1M dataset is collected from different sensors and platforms, with a spatial resolution ranging from 0.3 m to 0.8 m. There are more than 1 million instances and more than 40,000 images in this dataset. All objects in the FAIR1M dataset are annotated with respect to 5 categories and 37 sub-categories by oriented bound-

ing boxes. Each image size ranges from 1000 × 1000 to 10,000 × 10,000 pixels and contains objects exhibiting a wide variety of scales, orientations, and shapes.

Our proposed method only involves post-processing for dense detectors, meaning it does not include a training phase. For DOTA-v1.0, DIOR-R, we used the test set for testing. We tested the validation set on FAIR1M-v2.0. All images were cropped into 1024 × 1024 patches with a gap of 200.

4.2. Implementation Details

We use MMRotate [49] to implement it and use the official MMRotate model for testing. We use a single NVIDIA RTX-3090 GPUs@24G for testing. The NMS threshold was set to 0.1 when merging image patches, and the confidence threshold was set to 0.05 during testing. For the YOLO26 [26] and RTMDet [50] models, we used IoU and confidence thresholds of 0.2 and 0.1, respectively.

4.3. Ablation Study

We used the S^2ANet [8] and RetinaNet [2] models as baselines, conducting ablation experiments on the DOTA-v1.0 dataset to explore the impact of hyperparameters α and β on DPNMS performance. First, we explored the impact of IoU and confidence on the performance of DPNMS. The results are shown in Table 1. The "-" symbol indicates the results of the original NMS. If $\alpha = 0$, it means that the confidence score is not used; if $\beta = 0$, it means that IoU is used only to construct the correlation matrix ($Mask_{iou}$). Confidence and IoU produced similar effects on both the RetinaNet [2] and S^2ANet [8] models. Using only the IoU between the models' outputs yielded better performance than using confidence. However, employing only one type of information also resulted in a significant decrease in the model's precision.

Table 1. The impact of IoU and confidence on detection precision.

α	β	RetinaNet			S^2ANet		
		AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP
-	-	65.36	39.23	38.06	73.05	35.56	39.42
0	1	61.37	36.59	35.78	70.29	36.65	38.77
1	0	26.00	14.26	14.77	28.45	13.82	15.40

To evaluate the influence of parameters α and β on detection performance, we conducted extensive experiments on parameter combinations within the ranges $\alpha, \beta \in [1, 6]$. Table 2 lists all the results of the ablation experiments. Parameters α and β both affect performance. By fixing α and β separately, we observed how changes in the other parameter affected performance. We found that precision peaks as the parameter increases and then declines. Additionally, the optimal parameter combinations show differences across models. For the RetinaNet [2], the parameter combinations (6, 2), (5, 3), and (5, 4) are optimal, whereas for the S^2ANet [8], the optimal combinations are (2, 3), (5, 3), and (5, 4). Overall, we recommend selecting (5, 3) and (5, 4) as the general parameter combinations. Because confidence and IoU are multiplied directly, larger values of α and β indicate stronger discriminative capability. However, as both confidence and IoU lie within the range [0, 1], the individual density components become tiny when α and β increase, making them easily affected by floating-point calculation errors. Therefore, we selected the smaller combination (5, 3) as the parameters for our following experiments. As shown in Table 2, the (5, 3) parameter has a limited impact on AP_{50}, but it significantly improves AP_{75} and mAP. For RetinaNet [2], compared to the original NMS, AP_{75} and mAP improved by 1.6%

and 1.02%, respectively, with this parameter. The S^{2}ANet [8] achieved improvements of 2.81% and 1.4% in AP_{75} and mAP, respectively.

Table 2. The impact of α and β for DPNMS.

α	β	RetinaNet			α	β	RetinaNet			α	β	S^{2}ANet			α	β	S^{2}ANet		
		AP_{50}	AP_{75}	mAP			AP_{50}	AP_{75}	mAP			AP_{50}	AP_{75}	mAP			AP_{50}	AP_{75}	mAP
-	-	65.36	39.23	38.06	4	1	65.51	40.76	38.78	-	-	73.05	35.56	39.42	4	1	72.75	37.98	40.42
1	1	64.19	40.08	37.92	4	2	65.69	40.88	38.99	1	1	71.99	37.61	39.91	4	2	72.99	38.16	40.59
1	2	64.38	39.83	37.98	4	3	65.72	40.79	39.07	1	2	72.64	38.47	40.52	4	3	72.95	38.35	40.70
1	3	64.49	39.92	38.05	4	4	65.74	40.71	38.99	1	3	72.75	38.20	40.53	4	4	72.91	38.42	40.80
1	4	64.72	39.93	38.25	4	5	65.60	40.71	38.95	1	4	72.72	38.05	40.61	4	5	72.89	38.27	40.69
1	5	64.77	39.89	38.16	4	6	65.51	40.66	38.99	1	5	72.75	37.88	40.65	4	6	72.87	38.64	40.77
1	6	64.93	39.96	38.28	5	1	65.37	40.69	38.79	1	6	72.86	37.76	40.48	5	1	72.64	37.92	40.19
2	1	65.09	40.51	38.73	5	2	65.70	40.89	39.08	2	1	72.55	38.00	40.34	5	2	73.05	38.27	40.69
2	2	65.11	40.46	38.59	5	3	65.78	40.82	39.08	2	2	72.77	38.29	40.52	5	3	72.95	38.37	40.82
2	3	65.19	40.25	38.60	5	4	65.68	40.96	39.15	2	3	72.89	38.74	40.94	5	4	72.89	38.51	40.84
2	4	65.26	40.20	38.61	5	5	65.59	40.74	39.16	2	4	73.02	38.45	40.83	5	5	72.93	38.62	40.77
2	5	65.30	40.21	38.71	5	6	65.55	40.87	39.12	2	5	72.99	38.38	40.68	5	6	72.92	38.70	40.68
2	6	65.36	40.34	38.69	6	1	65.21	40.76	38.81	2	6	73.06	38.59	40.78	6	1	72.58	38.13	40.40
3	1	65.25	40.61	38.73	6	2	65.76	40.96	39.14	3	1	72.60	38.04	40.39	6	2	72.96	38.46	40.77
3	2	65.58	40.63	38.85	6	3	65.71	40.91	39.16	3	2	72.90	38.12	40.63	6	3	73.02	38.34	40.71
3	3	65.77	40.69	38.95	6	4	65.65	40.90	39.20	3	3	73.00	38.55	40.93	6	4	72.95	38.56	40.82
3	4	65.61	40.64	38.84	6	5	65.52	40.93	39.20	3	4	72.95	38.28	40.64	6	5	72.94	38.41	40.76
3	5	65.55	40.67	38.85	6	6	65.55	40.94	39.06	3	5	73.00	38.17	40.91	6	6	72.96	38.14	40.55
3	6	65.56	40.68	38.92	-	-	-	-	-	3	6	73.05	38.35	40.57	-	-	-	-	-

4.4. Study on IoU and Density

We conducted statistical analyses on the DOTA-v1.0 and DIOR-R datasets to explore the relationship between the density and IoU of candidate boxes relative to the ground truth. We adopted RetinaNet [2] and S^{2}ANet [8] as our baseline models. We trained our model using the training and validation sets of the DOTA-v1.0 dataset and evaluated it using the validation set. For the DIOR-R dataset, we trained our model using the training and validation sets, then evaluated it using the test set. We calculated the IoU between each candidate bounding box and the ground truth (GT) bounding boxes, determined the corresponding GT based on the highest IoU, and then constructed clusters for each GT. For candidate boxes with a maximum IoU below the threshold, we consider them not belonging to any GT cluster. Here, we use the IoU threshold from the model evaluation (default threshold is 0.1). We sorted the candidate boxes in each GT cluster by density and IoU, and calculated their ranking percentage within the cluster ($P = \frac{Order}{Number_{cluster}}$). A lower value indicates a higher rank. The value at the end of the cluster being 1.0. We divided the IoU rankings into 10 statistical intervals with 0.1-interval steps ranging from 0 to 1, and calculated the mean of the density rankings for each interval across the entire dataset, as shown in Figures 4 and 5.

For Figure 4, we conducted experiments on the DIOR-R dataset using RetinaNet [2] as the baseline, covering hyperparameters $\alpha, \beta \in [0, 5]$. When $\alpha = 1, \beta = 0$, the density ranking percentage fluctuates around 0.49, and the IoU ranking percentage has virtually no effect on it. Therefore, we believe no correlation exists between the density ranking and the IoU ranking for this case. When $\beta = 1$, we observed a consistent trend across multiple results. Overall, as the IoU ranking percentage value increases, the density ranking percentage value also increases. This indicates that the lower a candidate box's IoU, the lower its density. This significant positive correlation validates the effectiveness of density as a metric for filtering candidate boxes.

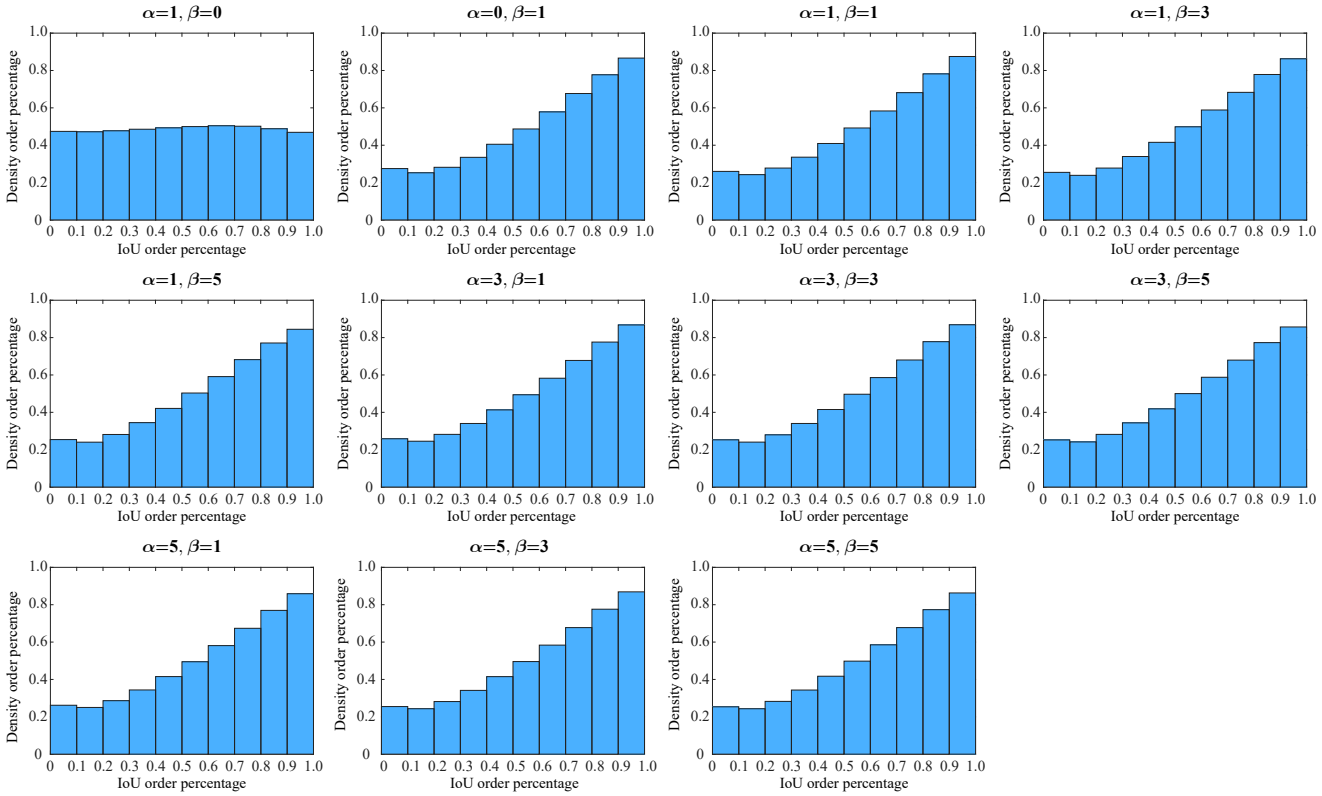


Figure 4. Statistics on density ranking and IoU ranking for RetinaNet with different parameters on the DIOR-R dataset. When $\beta = 0$, density ranking has little effect on IoU ranking. Density ranking and IoU ranking can be considered unrelated in this case. For multiple results where $\beta > 0$, a positive correlation between IoU ranking and density ranking emerges, indicating that the hypothesis regarding the validity of using density as a metric is reasonable.

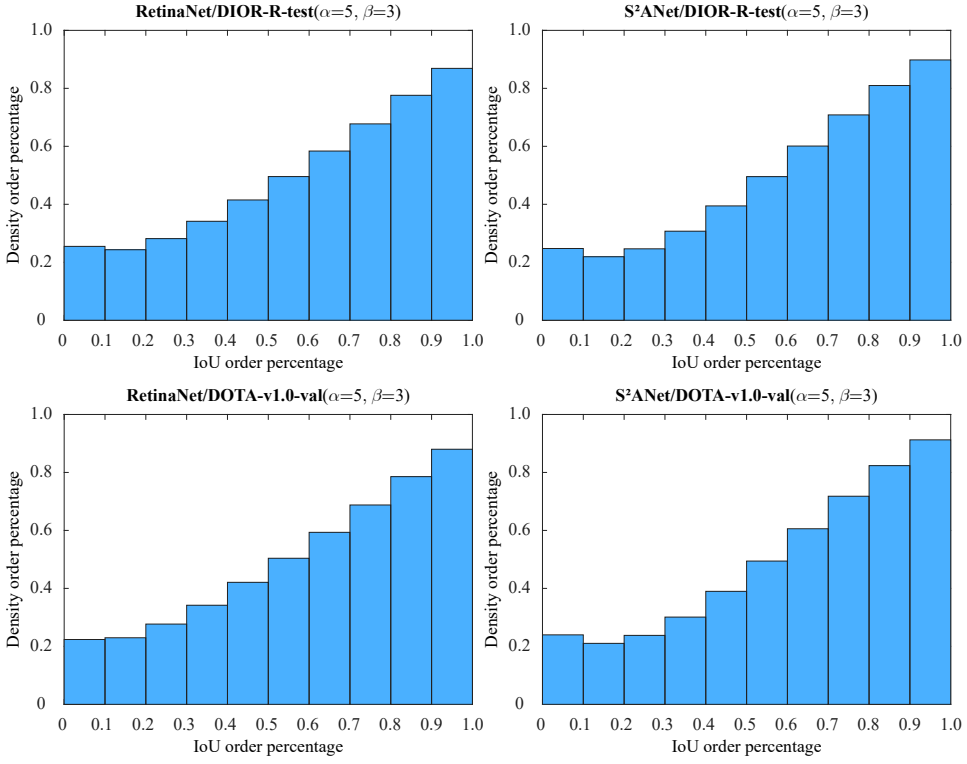


Figure 5. Statistical results for IoU-density ranking of RetinaNet and S^2ANet on the DOTA and DIOR-R Datasets. Overall, with parameters $\alpha = 5, \beta = 3$, the IoU rankings and density rankings of Retinanet and S^2ANet generally show a positive correlation.

We used RetinaNet [2] and S^2ANet [8] as baselines and conducted a statistical analysis using the general parameters $\alpha = 5, \beta = 3$, covering the DOTA and DIOR-R datasets. The results are shown in Figure 5. Figure 5 shows a phenomenon similar to that observed in Figure 4, indicating that the output patterns of these detectors all meet the data requirements of the density peak algorithm. We also observe from the figure that for the top 10% ranked in IoU, their average density is actually slightly lower than that of the 10% to 20% range, potentially resulting in a density peak that is not optimal. We believe this is due to the density definition function and the peak selection strategy. Therefore, future research could explore designing more reasonable density functions and selection strategies to improve performance.

We statistically analyzed the mean percentage of IoU rankings for the detection results of the original NMS, DPNMS, and GauS within the GT clusters, as shown in Table 3. The IoU ranking percentage is a normalized representation of the IoU ranking of detection results within the corresponding GT clusters. Its average is calculated as the mean of the IoU rankings across the entire test dataset. This metric ranges from [0, 1]. The smaller the value, the closer the detection results are to the ground truth across the entire test set. Overall, the NMS detection boxes ranked behind DPNMS and GauS on the DOTA-v1.0 and DIOR-R datasets. This indicates that their detection results have a higher IoU with GT, and therefore perform better on high-precision metrics.

Table 3. The IoU ranking percentage of the detection results in the corresponding cluster.

Detectors	Methods	Order Percentage (%)	
		DOTA-v1.0	DIOR-R
RetinaNet	NMS	23.27	24.40
	DPNMS	21.44	21.55
	GauS	21.47	22.36
S^2ANet	NMS	30.81	29.43
	DPNMS	24.23	24.05
	GauS	22.96	23.27

4.5. Study on Select–Release Loop

A one-time peak filtering carries the risk of missing valid detections, so repeated correction is a standard procedure. To study the impact of the maximum iterations on model performance, we conducted experiments on the DOTA-v1.0 dataset using RetinaNet and S^2ANet as baselines; the result as shown in Table 4. Referring to the data in Table 4, we found that applying only one peak filtering yielded the lowest performance. As the maximum iterations increased, the precision gradually improved until it stabilized. The maximum iterations refer to the maximum cycles in the select–release loop. It means that if the detection results stabilize before the maximum iterations, the loop will exit early; otherwise, the filter results from the last iteration will be output. Therefore, as it exceeds a certain value, the model's performance will not continue to change. Due to the early termination mechanism, the actual number of iterations also stabilizes accordingly, resulting in minimal impact on computational latency. Therefore, we set the default value for the maximum iterations to 10 and used it in all other experiments.

Table 4. The impact of different maximum iterations on detector performance.

Max Iterations	RetinaNet			S^2ANet		
	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP
1	64.41	40.80	38.54	72.01	38.05	40.36
2	65.70	40.71	38.98	72.83	38.35	40.80
4	65.78	40.82	39.07	72.90	38.36	40.76
6	65.78	40.82	39.08	72.95	38.32	40.78
8	65.78	40.82	39.08	72.95	38.37	40.82
10	65.78	40.82	39.08	72.95	38.37	40.82

4.6. Comparisons with Other Methods

We conducted extensive experiments on multiple datasets and detectors; the result are shown in Table 5. The detectors included RetinaNet [2], R^3det [7], S^2ANet [8], and RTMDet [50], while the datasets included DOTA-v1.0 [46] (Train and validation set for training, test set for testing), DIOR-R [47] (Train and validation set for training, test set for testing), and FAIR1M-v2.0 [48] (Train set for training, validation set for testing). The post-processing methods compared include the original NMS, DIoU-NMS [40], Soft-NMS [38], NMW [43], WBF [44], GauS [44], Fast-NMS [29], and Cluster-NMS [30]. Methods such as NMW [43], WBF [44], and GauS [45] synthesize multiple bounding boxes into a single one, while the others represent conventional methods. The results are shown in Table 5, where **red bold** indicates the current best value, while **blue bold** indicates the second-best result. We took S^2ANet [8] as a baseline, with a confidence threshold of 0.3, and visualized the results for the above methods, as shown in Figure 6.

In natural scenarios, DIoU-NMS [40] incorporates a distance component into IoU, enhancing its ability to differentiate objects. In remote sensing scenarios, the OBB more accurately reflects the true boundaries of objects compared to the HBB. Hence, the OBB's IoU more accurately represents the distances between objects, lessening the extra effect of center-to-center distance. As illustrated in Table 5 and Figure 6, the results of DIoU-NMS [40] are very close to the original NMS, further supporting our analysis.

In natural scenes, objects often overlap significantly. The hard threshold used in the original NMS can easily filter out overlapping objects incorrectly. Soft-NMS [38] adjusts the confidence based on the distance between the objects' center points to keep occluded boxes. However, the overlap between remote sensing objects is quite low, meaning Soft-NMS [38] will retain more detections, which leads to fluctuations in performance.

Both the NMW [43] and the WBF [44] methods are used to synthesize internal boundaries. As they perform a weighted sum of the boundary parameters directly, they cannot handle specific angular ranges, and the synthesis results are unstable. Therefore, the performance of NMW [43] and WBF [44] in Table 5 is highly unstable. Figure 6 illustrates that NMW synthesis results can help refine NMS outputs. However, the effect is limited.

GauS [45] resolves the challenge of angular synthesis offset using the Gaussian distribution, which reliably improves detection performance. However, it relies on the preliminary results from the original NMS. Based on these results, it combines other localization information within the same cluster into a single box. GauS [45] is essentially a method for adjusting NMS boundaries. It can generate more precise bounding boxes. However, it cannot handle missed objects. Therefore, for the overlapping vehicles and aircraft in the first and second columns of the sub-images shown in Figure 6, GauS failed to identify the missing objects, whereas DPNMS could detect them.

Fast-NMS [29] computes the IoU for all candidate boxes in one pass, then filters and selects location boxes using sorting and triangulation. Due to Fast-NMS conducting the filtering step just once, some valid detections might be overlooked, resulting in reduced

accuracy. Cluster-NMS [30] iterates multiple times based on Fast-NMS, gradually correcting detections that were erroneously suppressed, thereby achieving the same accuracy as the original NMS.

DPNMS achieved competitive performance in most experiments, with results that were mostly best or second-best, validating the effectiveness of the proposed method. It achieved significant performance improvements on detectors such as RetinaNet [2], R^3det [7], and S^2Anet [8]. However, applying GauS and DPNMS significantly weakened the performance gains of the RTMDet [50] model. We believe it arises from Quality Focus Loss [51], which is used in the RTMDet model to align confidence and IoU. As a result, it is easier to preserve high-quality bounding boxes, limiting the performance of methods such as GauS and DPNMS.

Table 5. The comparison of DPNMS with other post-processing methods.

Detectors	Backbone	Method	DOTA v1.0 Trainval/Test			DIOR-R Trainval/Test			FAIR1M-v2.0 Train/Val		
			AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP
RetinaNet	ResNet50 [52]	NMS	65.36	39.23	38.06	54.44	34.54	34.29	27.67	16.57	16.38
		DIoU-NMS	65.29	38.88	37.98	54.53	34.48	34.31	27.61	16.57	16.36
		Soft-NMS	65.28	39.18	38.07	50.64	32.38	31.96	27.61	16.57	16.37
		NMW	62.25	26.67	32.18	49.18	21.33	25.94	**28.31**	14.09	15.15
		WBF	54.30	16.04	24.55	43.89	13.84	20.30	25.71	10.52	12.94
		GauS	**65.53**	**40.73**	**39.16**	**54.91**	**36.61**	**35.37**	27.95	**17.98**	**17.21**
		Fast-NMS	64.64	38.78	37.81	53.48	34.36	34.01	27.11	16.38	16.21
		Cluster-NMS	65.36	39.23	38.06	54.44	34.54	34.29	27.67	16.57	16.38
		Ours	**65.78**	**40.82**	**39.08**	**54.76**	**37.09**	**35.66**	**28.35**	**18.06**	**17.50**
R^3Det	ResNet50 [52]	NMS	**69.85**	37.71	38.60	54.54	32.77	32.33	**31.39**	18.20	18.24
		DIoU-NMS	69.48	37.20	38.33	54.52	32.71	32.31	31.31	18.20	18.21
		Soft-NMS	69.77	37.59	38.61	53.50	32.36	31.82	31.34	18.19	18.22
		NMW	69.14	38.98	38.85	**55.03**	32.31	32.51	31.12	18.80	18.64
		WBF	65.39	34.38	35.95	51.88	26.71	28.80	28.49	16.84	16.99
		GauS	69.68	**39.93**	**39.70**	**55.20**	**33.72**	**33.39**	**31.36**	**19.03**	**18.70**
		Fast-NMS	68.61	37.14	38.16	54.17	32.52	32.18	30.98	18.18	18.11
		Cluster-NMS	**69.85**	37.71	38.60	54.54	32.77	32.33	**31.39**	18.20	18.24
		Ours	**69.94**	**40.21**	**39.73**	54.82	**33.60**	**33.17**	31.21	**19.01**	**18.75**
S^2ANet	ResNet50 [52]	NMS	**73.05**	35.56	39.42	61.34	36.75	36.36	**35.31**	22.44	21.44
		DIoU-NMS	72.98	35.57	39.38	61.65	36.63	36.41	35.18	22.44	21.41
		Soft-NMS	73.00	35.71	39.51	**62.06**	36.79	36.57	35.21	22.44	21.42
		NMW	71.38	36.88	39.65	51.89	21.07	26.16	34.41	17.62	19.20
		WBF	67.09	32.05	35.78	45.85	16.52	21.96	32.15	16.25	17.71
		GauS	**73.06**	**39.00**	**40.93**	**61.98**	**38.35**	**37.82**	**35.24**	**22.69**	**21.79**
		Fast-NMS	72.63	36.00	39.48	60.82	36.45	36.14	34.98	22.40	21.33
		Cluster-NMS	**73.05**	35.56	39.42	61.34	36.75	36.36	**35.31**	22.44	21.44
		Ours	72.95	**38.37**	**40.82**	61.46	**37.96**	**37.33**	35.11	**22.61**	**21.72**
RTMDet	CSPNeXt-m [50]	NMS	76.17	53.27	48.77	67.31	47.75	45.72	38.53	25.40	24.66
		DIoU-NMS	76.20	**53.32**	48.76	67.38	47.74	45.73	38.51	25.40	24.66
		Soft-NMS	**76.21**	53.25	48.74	**67.39**	47.74	**45.78**	38.44	25.40	24.65
		NMW	75.89	52.41	48.41	67.38	**47.89**	45.73	38.44	25.28	24.59
		WBF	75.14	51.57	47.63	66.93	46.68	45.22	38.44	25.40	24.65
		GauS	**76.31**	53.27	**48.83**	**67.40**	**47.96**	**45.82**	**38.64**	**25.49**	**24.87**
		Fast-NMS	76.18	53.30	48.71	66.78	47.76	45.62	38.38	25.41	24.64
		Cluster-NMS	76.17	53.27	48.77	67.31	47.75	45.72	38.53	25.40	24.66
		Ours	76.14	**53.42**	**48.87**	67.17	47.67	45.59	**38.56**	**25.56**	**24.82**

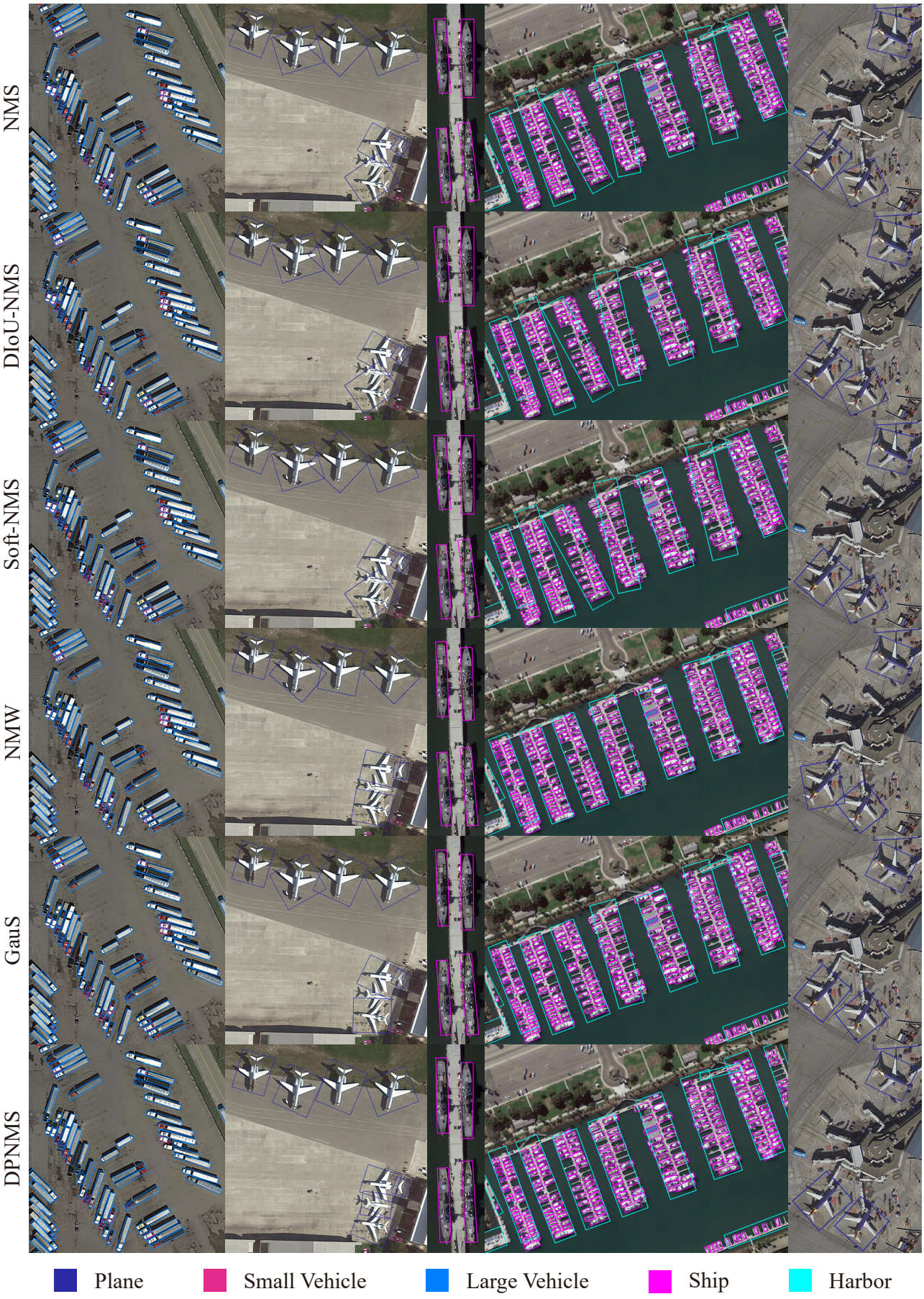


Figure 6. Comparison with other post-processing methods on DOTA-v1.0. The confidence threshold is set to 0.3 when visualizing these results.

Overall, the results of GauS [45] and DPNMS are stable, achieving the best and second-best performance in most experiments. GauS depends on preliminary NMS filters, which lead to higher computational latency than DPNMS. The detailed results are presented in Section 4.7. DPNMS considers the distribution of surrounding bounding boxes to determine which ones to keep. This method employs the density peak clustering framework, moving away from the original NMS's confidence-first paradigm and the mode of simply discarding boxes within a threshold. It constructs a density map by integrating candidate boxes to find the center of their distribution, thereby obtaining an OBB that is closer to the object's boundary.

4.7. Latency Cost

We conducted comprehensive comparisons of the inference latency for different post-processing methods. These are shown in Tables 6 and 7. The experiments reported in Table 6 were conducted on the DOTA-v1.0 dataset (Test set), with image sizes of 1024 × 1024, using mmrotate [49] as the runtime environment, and 5000 test images for each method. We tested the latency of models without post-processing algorithms to help estimate the latency of specific post-processing methods. We statistically analyzed the cumulative percentage of candidate boxes generated by the above model on the DOTA-v1.0 test set, as shown in Figure 7. For RetinaNet, 95% of the patches in the entire DOTA-v1.0 test set have fewer than 3592 candidate boxes, and 99% of the patches have fewer than 4961 candidate boxes. The Cumulative percentage of R^3Det is (3030, 0.95) and (4097, 0.99); the result of S^2ANet is (2672, 0.95) and (3604, 0.99).

The original NMS is the official CUDA implementation by mmcv [53], which offers extremely high computational efficiency. We use this as a baseline for analyzing other methods.

DIoU [40] and Soft-NMS [38] are implemented in mmrotate [49], resulting in lower computational efficiency and higher latency. Both methods introduce additional computational distance penalty terms, increasing the computational load compared to traditional NMS. However, they do not adjust the serial processing logic, meaning no latency benefits are achieved.

NMW [43] introduced single-step weighted fusion based on NMS, making its computation latency similar to the original NMS. WBF [44] requires multiple iterations for fusion. As it lacks a standalone NMS module, it runs in a mmrotate [49] environment, which results in significantly increased latency.

GauS [45] could be considered an improved version of NMW for OBB. It introduced Gaussian encoding and decoding operations, resulting in higher latency than NMW. Unlike NMS, which uses serial processing, DPNMS breaks away from this pattern by constructing large-scale density matrices and performing parallel optimization, achieving rapid post-processing.

Fast-NMS, Cluster-NMS, and DPNMS all include steps for calculating the IoU matrix. Fast-NMS performs a single filtering operation, which sacrifices some accuracy but offers the fastest speed. Both Cluster-NMS and DPNMS use multi-iterative optimization approaches, but Cluster-NMS requires multiplying the entire IoU matrix during each iteration, so an increase in the number of candidate boxes significantly increases latency. During the iteration process, DPNMS gradually removes suppressed candidate boxes, reducing the scale of the matrix and decreasing latency. As illustrated in Figure 7, RetinaNet, R^3Det, and S^2ANet produce significantly more candidate bounding boxes per object than RTMDet. Since the data scale dominates the processing latency, Cluster-NMS is slower than DPNMS. RTMDet produces fewer candidate boxes, while DPNMS involves more complex operations

per iteration and requires longer processing times in a Python environment (Version 3.8.14). Consequently, in the RTMDet baseline, it takes longer to run than Cluster.

Table 6. The comparison of inference latency on a single RTX 3090 GPU.

Detectors	Backbone	Method	FPS (img/s)	Latency (ms)
RetinaNet	ResNet50	No-NMS	32.99	30.31
		NMS(CUDA)	26.47	37.77 (+7.46)
		DIoU-NMS	13.48	74.17 (+43.86)
		Soft-NMS	14.13	70.78 (+40.47)
		NMW	26.25	38.09 (+7.78)
		WBF	2.17	460.35 (+430.04)
		GauS	25.62	39.04 (+8.73)
		Fast-NMS	26.74	37.39 (+7.08)
		Cluster-NMS	26.27	38.06 (+7.75)
		Ours	26.60	37.59 (+7.28)
R^3Det	ResNet50	No-NMS	20.87	47.91
		NMS(CUDA)	19.12	52.31 (+4.39)
		DIoU-NMS	11.94	83.74 (+35.83)
		Soft-NMS	12.30	81.28 (+33.36)
		NMW	19.08	52.42 (+4.50)
		WBF	3.56	280.54 (+232.63)
		GauS	18.84	53.08 (+5.16)
		Fast-NMS	19.33	51.73 (+3.82)
		Cluster-NMS	19.15	52.22 (+4.31)
		Ours	19.17	52.16 (+4.24)
S^2ANet	ResNet50	No-NMS	24.72	40.46
		NMS(CUDA)	22.47	44.51 (+4.05)
		DIoU-NMS	13.50	74.05 (+33.59)
		Soft-NMS	14.22	70.32 (+29.86)
		NMW	21.94	45.58 (+5.13)
		WBF	4.99	200.40 (+159.95)
		GauS	21.64	46.22 (+5.76)
		Fast-NMS	22.94	43.59 (+3.13)
		Cluster-NMS	22.62	44.21 (+3.75)
		Ours	22.83	43.79 (+3.34)
RTMDet	CSPNeXt-m	No-NMS	38.44	26.02
		NMS(CUDA)	33.03	30.27 (+4.26)
		DIoU-NMS	24.14	41.42 (+15.40)
		Soft-NMS	25.52	39.18 (+13.17)
		NMW	33.05	30.25 (+4.24)
		WBF	11.44	87.38 (+61.36)
		GauS	31.99	31.26 (+5.24)
		Fast-NMS	34.84	28.70 (+2.68)
		Cluster-NMS	34.27	29.18 (+3.16)
		Ours	33.39	29.95 (+3.94)

Table 7. The comparison of inference latency on the Jetson Orin NX super platform.

Detector	Image Size	Methods	AP_{50}	AP_{75}	mAP	FPS	Latency
YOLO26n	1024 × 1024	No-NMS	-	-	-	124.27	8.05 ms
		NMS	74.83	53.34	49.40	82.87	12.07 ms
		End-to-End	73.23	52.44	48.47	124.28	8.05 ms
		Ours	75.13	53.49	49.66	110.63	9.04 ms

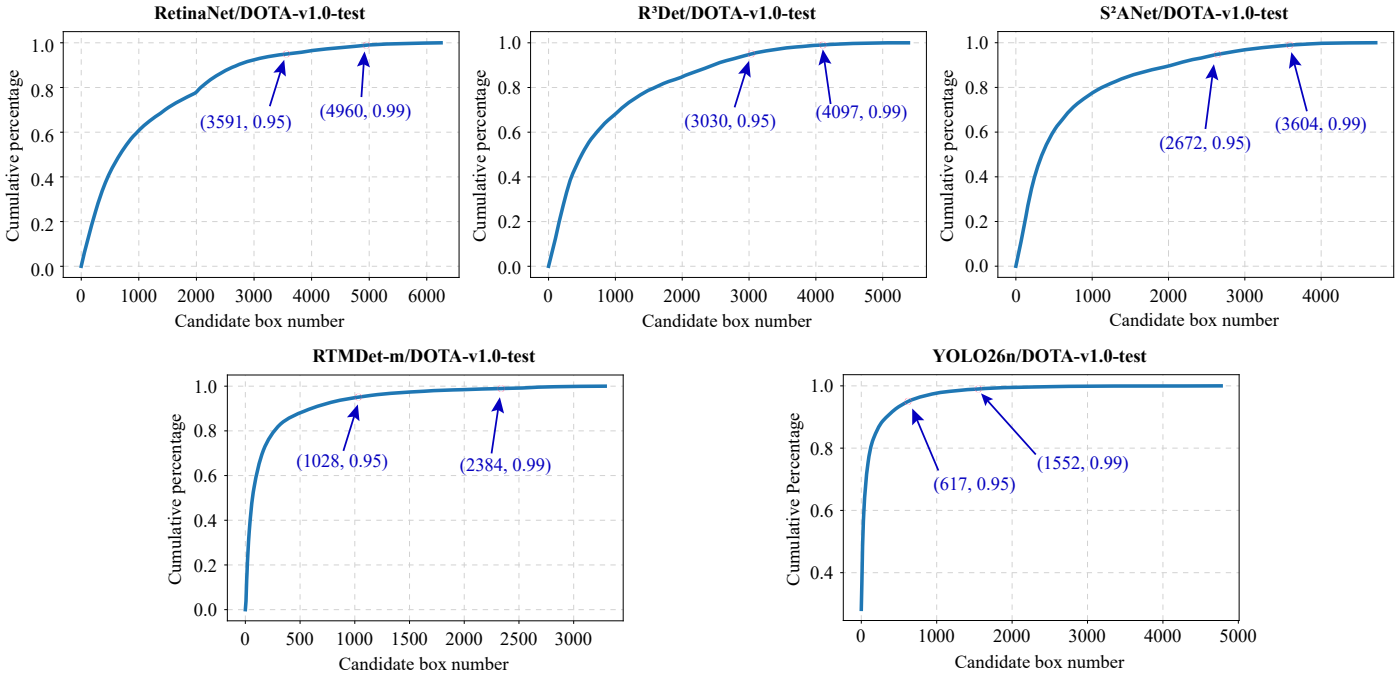


Figure 7. Cumulative count of remaining candidate boxes after confidence filtering on the DOTA-v1.0 dataset. Where the x-axis represents the number of remaining boxes, and the y-axis represents their cumulative percentage. The RetinaNet, R^3Det, and S^2ANet models produce more candidate instances than RTMDet-m and YOLO26n, resulting in a gap in post-processing latency.

To further explore the potential of DPNMS, we developed the TensorRT plugin for it and conducted experiments based on the YOLO26n [26], as shown in Table 7. We used "nano", the lightest official one in the YOLO26 series as our baseline, specifically its FP16 TensorRT model. The FPS value represents the average speed calculated after running inferences on the TEST set of the entire DOTA-v1.0 dataset. By optimizing intermediate variables and cache space, our DPNMS achieved a computation latency of nearly 1 ms on the Jetson Orin NX platform, reaching an overall FPS of 110.63 without sacrificing precision. The processing speed of DPNMS is approximately four times that of TensorRT's official NMS (https://github.com/NVIDIA/TensorRT, accessed on 17 November 2026.) Although the end-to-end branch of YOLO26 is faster at 124.28 FPS, the difference in mAP and DPNMS is approximately 1.19%. An increase in latency of just 1 ms is perfectly acceptable. The latency of DPNMS on the Jetson NX stems primarily from three factors. First, operator parallelization; second, fast processing by TensorRT (C++); and third, lightweight candidate boxes. During each iteration, DPNMS handles the retained candidate boxes in batches, and speeds up the process through parallel computation. However, the filtering operations in DPNMS involve complex instructions, and the code runs significantly faster in a C++ environment, resulting in lower latency compared to code running in a PyTorch environment (Version 1.11.0). Finally, for 95% of the patches in the DOTA-v1.0 test set, YOLO26n generated fewer than 618 candidate boxes (As shown in Figure 7), representing a relatively low computational load.

4.8. Analysis and Comprehension for DPNMS

Traditional NMS methods require high alignment between IoU and confidence, where high-quality candidate boxes are paired with high confidence scores to increase retention probability. DPNMS combines IoU and confidence scores of candidate boxes to determine which one to retain. Even if a candidate box does not have the highest confidence score within its cluster, it may still be kept. Consequently, DPNMS reduces the coupling re-

quirements between the classification branch and the localization branch of the model. Furthermore, in the DPNMS workflow, computations involving different candidate boxes are independent of each other, making the process easy to parallelize. Even with numerous iterations, the latency becomes negligible. Crucially, DPNMS offers a novel design approach for post-processing algorithms, thereby advancing the development of NMS-based detectors.

5. Conclusions

The original NMS method has multiple problems, including the risk of incorrect detections due to misalignment between classification and localization branches, and high latency from sequential processing. These limit the development of NMS-based detectors. To solve the above problem, we proposed a novel NMS method based on density peak clustering, named DPNMS. By taking a density peak cluster perspective, it achieves better results than the original NMS and reduces the coupling requirements between the classification and localization branches. DPNMS processes each candidate design independently, reducing latency significantly through large-scale parallel computing. We conducted systematic experiments across multiple datasets and detectors, achieving the best or second-best accuracy in most of them, which demonstrated the effectiveness and stability of the proposed method. Using the YOLO26n detector, we achieved 1-millisecond processing latency on the Jetson Orin NX Super platform and the DOTA-v1.0 test set, significantly narrowing the speed gap between end-to-end detectors and NMS-based detectors. Non-maximum suppression and DPNMS essentially represent specific approaches that have evolved from different computational strategies within the peak filtering mode. The DPNMS has provided new insights into the development of NMS.

Author Contributions: Conceptualization, Z.L. and B.X.; methodology, Z.L. and Z.W.; software, Z.L. and Z.W.; validation, Z.W. and Z.L.; formal analysis, Z.L. and Z.W.; investigation, Z.L.; resources, Z.L. and B.X.; data curation, Z.L.; writing—original draft preparation, Z.L. and Z.W.; writing—review and editing, B.X.; visualization, Z.W.; supervision, B.X.; project administration, Z.L.; funding acquisition, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 6250013632, and the Postdoctoral Fellowship Program (Grade C) of the China Postdoctoral Science Foundation under Grant Number GZC20252295, and the China Postdoctoral Science Foundation under Grant Number 2025MD784187.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets DOTA-v1.0, DIOR-R, and FAIR1M-v2.0 are publicly available. The DOTA dataset can be found at https://captain-whu.github.io/DOTA (accessed on 1 December 2019); the DIOR-R dataset can be found at https://gcheng-nwpu.github.io/#Datasets (accessed on 31 October 2026); The FAIR1M dataset can be found at https://www.gaofen-challenge.com/benchmark (accessed on 15 August 2026). The code used in this manuscript is available at: https://github.com/lzh420202/dgl (accessed on 17 November 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99.
2. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2017; pp. 2999–3007. https://doi.org/10.1109/ICCV.2017.324.

3. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *Proceedings of the 2020 European Conference on Computer Vision (ECCV)*; Springer: Cham, Switzerland, 2020; pp. 213–229. https://doi.org/10.1007/978-3-030-58452-8_13.
4. Zhou, Q.; Yu, C. Object Detection Made Simpler by Eliminating Heuristic NMS. *IEEE Trans. Multimed.* **2023**, *25*, 9254–9262. https://doi.org/10.1109/TMM.2023.3248966.
5. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850. https://doi.org/10.48550/arXiv.1904.07850.
6. Feng, C.; Zhong, Y.; Gao, Y.; Scott, M.R.; Huang, W. TOOD: Task-aligned One-stage Object Detection. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2021; pp. 3490–3499. https://doi.org/10.1109/ICCV48922.2021.00349.
7. Yang, X.; Yan, J.; Feng, Z.; He, T. R3Det: Refined Single-Stage Detector with Feature Refinement for Rotating Object. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 3163–3171. https://doi.org/10.1609/aaai.v35i4.16426.
8. Han, J.; Ding, J.; Li, J.; Xia, G.S. Align Deep Features for Oriented Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–11. https://doi.org/10.1109/TGRS.2021.3062048.
9. Ding, J.; Xue, N.; Long, Y.; Xia, G.S.; Lu, Q. Learning RoI Transformer for Oriented Object Detection in Aerial Images. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2019; pp. 2844–2853. https://doi.org/10.1109/CVPR.2019.00296.
10. Li, Z.; Hou, B.; Wu, Z.; Ren, B.; Yang, C. FCOSR: A Simple Anchor-Free Rotated Detector for Aerial Object Detection. *Remote Sens.* **2023**, *15*, 5499. https://doi.org/10.3390/rs15235499.
11. Yang, X.; Yan, J.; Ming, Q.; Wang, W.; Zhang, X.; Tian, Q. Rethinking Rotated Object Detection with Gaussian Wasserstein Distance Loss. In Proceedings of the 38th International Conference on Machine Learning, Virtually, 18–24 July 2021; Volume 139, pp. 11830–11841.
12. Yang, X.; Zhang, G.; Yang, X.; Zhou, Y.; Wang, W.; Tang, J.; He, T.; Yan, J. Detecting Rotated Objects as Gaussian Distributions and its 3-D Generalization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 4335–4354. https://doi.org/10.1109/TPAMI.2022.3197152.
13. Yang, X.; Zhou, Y.; Zhang, G.; Yang, J.; Wang, W.; Yan, J.; Zhang, X.; Tian, Q. The KFIoU Loss for Rotated Object Detection. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
14. Murrugarra-Llerena, J.; Kirsten, L.N.; Zeni, L.F.; Jung, C.R. Probabilistic Intersection-Over-Union for Training and Evaluation of Oriented Object Detectors. *IEEE Trans. Image Process.* **2024**, *33*, 671–681. https://doi.org/10.1109/TIP.2023.3348697.
15. Li, Z.; Hou, B.; Wu, Z.; Guo, Z.; Ren, B.; Guo, X.; Jiao, L. Complete Rotated Localization Loss Based on Super-Gaussian Distribution for Remote Sensing Images. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–14. https://doi.org/10.1109/TGRS.2023.3305578.
16. Hou, L.; Lu, K.; Yang, X.; Li, Y.; Xue, J. G-Rep: Gaussian Representation for Arbitrary-Oriented Object Detection. *Remote Sens.* **2023**, *15*, 757. https://doi.org/10.3390/rs15030757.
17. Xu, Y.; Fu, M.; Wang, Q.; Wang, Y.; Chen, K.; Xia, G.S.; Bai, X. Gliding Vertex on the Horizontal Bounding Box for Multi-Oriented Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1452–1459. https://doi.org/10.1109/tpami.2020.2974745.
18. Qian, W.; Yang, X.; Peng, S.; Yan, J.; Guo, Y. Learning modulated loss for rotated object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2021, Volume 35, pp. 2458–2466. https://doi.org/10.1609/aaai.v35i3.16347.
19. Lu, D.; Li, D.; Li, Y.; Wang, S. OSKDet: Orientation-sensitive Keypoint Localization for Rotated Object Detection. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2022; pp. 1172–1182. https://doi.org/10.1109/CVPR52688.2022.00125.
20. Zhao, P.; Qu, Z.; Bu, Y.; Tan, W.; Guan, Q. PolarDet: A fast, more precise detector for rotated target in aerial images. *Int. J. Remote Sens.* **2021**, *42*, 5831–5861. https://doi.org/10.1080/01431161.2021.1931535.
21. Yi, J.; Wu, P.; Liu, B.; Huang, Q.; Qu, H.; Metaxas, D. Oriented Object Detection in Aerial Images with Box Boundary-Aware Vectors. In *Proceedings of the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*; IEEE: Piscataway, NJ, USA, 2021; pp. 2149–2158. https://doi.org/10.1109/WACV48630.2021.00220.
22. Yuan, X.; Zheng, Z.; Li, Y.; Liu, X.; Liu, L.; Li, X.; Hou, Q.; Cheng, M.M. Strip R-CNN: Large strip convolution for remote sensing object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2026; Volume 40, pp. 12259–12267.
23. Wu, Z.; Liu, Y.; Dong, H.; Tang, X.; Yang, J.; Jin, B.; Chen, M.; Wei, X. R2det: Exploring relaxed rotation equivariance in 2d object detection. In Proceedings of the International Conference on Learning Representations, Singapore, 24–28 April 2025; Volume 2025, pp. 82455–82475.
24. Zhang, C.; Chen, Z.; Xiong, B.; Ji, K.; Kuang, G. EOOD: End-to-end oriented object detection. *Neurocomputing* **2025**, *621*, 129251.
25. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 107984–108011.
26. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748. https://doi.org/10.48550/arXiv.2606.03748.

27. Zeng, Y.; Chen, Y.; Yang, X.; Li, Q.; Yan, J. ARS-DETR: Aspect Ratio-Sensitive Detection Transformer for Aerial Oriented Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–15. https://doi.org/10.1109/TGRS.2024.3364713.
28. Robinson, I.; Robicheaux, P.; Popov, M.; Ramanan, D.; Peri, N. RF-DETR: Neural architecture search for real-time detection transformers. In Proceedings of the International Conference on Learning Representations, Rio de Janeiro, Brazil, 23–27 April 2026; Volume 2026, pp. 56087–56100.
29. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2019; pp. 9156–9165. https://doi.org/10.1109/ICCV.2019.00925.
30. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q.; Zuo, W. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Trans. Cybern.* **2021**, *52*, 8574–8586. https://doi.org/10.1109/TCYB.2021.3095305.
31. Si, K.S.; Sun, L.; Zhang, W.; Gong, T.; Wang, J.; Liu, J.; Sun, H. Accelerating Non-Maximum Suppression: A Graph Theory Perspective. *arXiv* 2024, arXiv:2409.20520. https://doi.org/10.48550/arXiv.2409.20520.
32. Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science* **2014**, *344*, 1492–1496.
33. Xie, X.; Cheng, G.; Wang, J.; Yao, X.; Han, J. Oriented R-CNN for Object Detection. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2021; pp. 3500–3509. https://doi.org/10.1109/ICCV48922.2021.00350.
34. Li, Y.; Hou, Q.; Zheng, Z.; Cheng, M.M.; Yang, J.; Li, X. Large Selective Kernel Network for Remote Sensing Object Detection. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2023; pp. 16748–16759. https://doi.org/10.1109/ICCV51070.2023.01540.
35. Hu, Z.; Gao, K.; Zhang, X.; Wang, J.; Wang, H.; Yang, Z.; Li, C.; Li, W. EMO2-DETR: Efficient-Matching Oriented Object Detection With Transformers. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–14. https://doi.org/10.1109/TGRS.2023.3300154.
36. Ma, X.; Lv, P.; Zhong, Y. QETR: A Query-Enhanced Transformer for Remote Sensing Image Object Detection. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 1–5. https://doi.org/10.1109/LGRS.2024.3378531.
37. Wang, J.; Song, L.; Li, Z.; Sun, H.; Sun, J.; Zheng, N. End-to-End Object Detection with Fully Convolutional Network. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2021; pp. 15844–15853. https://doi.org/10.1109/CVPR46437.2021.01559.
38. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Soft-NMS–Improving Object Detection with One Line of Code. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2017; pp. 5562–5570. https://doi.org/10.1109/ICCV.2017.593.
39. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. *arXiv* **2019**, arXiv:1902.09630.
40. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2020; Volume 34, pp. 12993–13000. https://doi.org/10.1609/aaai.v34i07.6999.
41. Gevorgyan, Z. SIoU loss: More powerful learning for bounding box regression. *arXiv* **2022**, arXiv:2205.12740. https://doi.org/10.48550/arXiv.2205.12740.
42. Wang, J.; Yin, X.; Wang, L.; Zhang, L. Hashing-based Non-Maximum Suppression for Crowded Object Detection. *arXiv* **2020**, arXiv:2005.11426. https://doi.org/10.48550/arXiv.2005.11426.
43. Ning, C.; Zhou, H.; Song, Y.; Tang, J. Inception Single Shot MultiBox Detector for object detection. In *Proceedings of the 2017 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*; IEEE: Piscataway, NJ, USA, 2017; pp. 549–554. https://doi.org/10.1109/ICMEW.2017.8026312.
44. Solovyev, R.; Wang, W.; Gabruseva, T. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image Vis. Comput.* **2021**, *107*, 104117. https://doi.org/10.1016/j.imavis.2021.104117.
45. Li, Z.; Hou, B.; Wu, Z.; Ren, B.; Ren, Z.; Jiao, L. Gaussian Synthesis for High-Precision Location in Oriented Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–12. https://doi.org/10.1109/TGRS.2023.3310619.
46. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: Piscataway, NJ, USA, 2018; pp. 3974–3983. https://doi.org/10.1109/CVPR.2018.00418.
47. Cheng, G.; Wang, J.; Li, K.; Xie, X.; Lang, C.; Yao, Y.; Han, J. Anchor-Free Oriented Proposal Generator for Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–11. https://doi.org/10.1109/TGRS.2022.3183022.
48. Sun, X.; Wang, P.; Yan, Z.; Xu, F.; Wang, R.; Diao, W.; Chen, J.; Li, J.; Feng, Y.; Xu, T.; et al. FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery. *ISPRS J. Photogramm. Remote Sens.* **2022**, *184*, 116–130. https://doi.org/10.1016/j.isprsjprs.2021.12.004.
49. Zhou, Y.; Yang, X.; Zhang, G.; Wang, J.; Liu, Y.; Hou, L.; Jiang, X.; Liu, X.; Yan, J.; Lyu, C.; et al. Mmrotate: A rotated object detection benchmark using pytorch. In *Proceedings of the 30th ACM International Conference on Multimedia*; ACM: New York, NY, USA, 2022; pp. 7331–7334. https://doi.org/10.1145/3503161.3548541.

50. Lyu, C.; Zhang, W.; Huang, H.; Zhou, Y.; Wang, Y.; Liu, Y.; Zhang, S.; Chen, K. RTMDet: An Empirical Study of Designing Real-Time Object Detectors. *arXiv* **2022**, arXiv:2212.07784. https://doi.org/10.48550/arXiv.2212.07784.
51. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21002–21012.
52. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2016; pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
53. MMCV Contributors. MMCV: OpenMMLab Computer Vision Foundation. 2018. Available online: https://github.com/open-mmlab/mmcv (accessed on 17 November 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.