Accelerating Non-Maximum Suppression: A Graph Theory Perspective

King-Siong Si[1*]   **Lu Sun**[1*]   **Weizhan Zhang**[1†]   **Tieliang Gong**[2]
Jiahao Wang[2]   **Jiang Liu**[3]   **Hao Sun**[3]

[1]School of Computer Science and Technology, MOEKLINNS Lab, Xi'an Jiaotong University
[2]School of Computer Science and Technology, BDKE Lab, Xi'an Jiaotong University
[3]Institute of Artificial Intelligence (TeleAI), China Telecom
{sjsinx, sunlu.cs}@stu.xjtu.edu.cn, {zhangwzh, gongtl}@xjtu.edu.cn
uguisu@stu.xjtu.edu.cn, {black_liu_1025, sun.010}@163.com

Abstract

Non-maximum suppression (NMS) is an indispensable post-processing step in object detection. With the continuous optimization of network models, NMS has become the "last mile" to enhance the efficiency of object detection. This paper systematically analyzes NMS from a graph theory perspective for the first time, revealing its intrinsic structure. Consequently, we propose two optimization methods, namely QSI-NMS and BOE-NMS. The former is a fast recursive divide-and-conquer algorithm with negligible mAP loss, and its extended version (eQSI-NMS) achieves optimal complexity of $\mathcal{O}(n \log n)$. The latter, concentrating on the locality of NMS, achieves an optimization at a constant level without an mAP loss penalty. Moreover, to facilitate rapid evaluation of NMS methods for researchers, we introduce NMS-Bench, the first benchmark designed to comprehensively assess various NMS methods. Taking the YOLOv8-N model on MS COCO 2017 as the benchmark setup, our method QSI-NMS provides $6.2\times$ speed of original NMS on the benchmark, with a 0.1% decrease in mAP. The optimal eQSI-NMS, with only a 0.3% mAP decrease, achieves $10.7\times$ speed. Meanwhile, BOE-NMS exhibits $5.1\times$ speed with no compromise in mAP.

1 Introduction

Object detection is a highly significant and popular topic in computer vision, widely applied in various domains, e.g., multiple object tracking [27, 41, 14], medical imaging analysis [42, 36], multimodal object detection [6, 44], and autonomous driving [30, 9, 11]. In recent years, there has been significant attention on the real-time performance of object detection, with notable successes achieved in several research endeavors [34, 31, 26]. Non-maximum suppression (NMS) [7] is a post-processing technique used to eliminate duplicate detection boxes and obtain final detections. Some research on NMS has indeed enhanced the mean average precision (mAP) of object detection, but they have also introduced additional computational overhead.

Currently, most CNN-based object detection models (such as the R-CNN family [13, 12, 34] and the YOLO series [31, 32, 33, 2]) consist of two parts in the testing phase: model inference and post-processing. In recent years, with the continuous emergence of model lightweighting techniques [17, 39, 8], the time cost of model inference has been significantly reduced. As a result, NMS gradually becomes a bottleneck in the pipeline of object detection [45]. To address this, some

---

[*]Equal contribution
[†]Corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

studies [4, 46, 40] have proposed parallelization methods to enhance NMS efficiency. However, these methods do not reduce computational overhead; they rely heavily on efficient parallel computing to reduce overall time costs. The degree of parallelism depends on the hardware environment (such as processor type, quantity, and number of cores) and architecture, leading to significant variations in efficiency when models are deployed across different platforms. Additionally, NMS research lacks a unified evaluation framework for two main reasons. First, existing NMS methods require a complete model inference for each test, consuming a significant amount of unnecessary computational resources. Second, different NMS methods are tested on different platforms using various models, making comparisons between different NMS algorithms challenging.

To reduce the computational overhead of NMS, we first map the set of bounding boxes obtained from model inference to a graph $\mathcal{G}$. We then conduct a comprehensive and systematic analysis of the intrinsic structure of NMS from a graph theory perspective. Each box is considered a node in the graph, and the suppression relationships are represented as arcs. We discovered that this forms a directed acyclic graph (DAG), allowing us to solve NMS using dynamic programming. This indicates that as long as the graph $\mathcal{G}$ can be quickly constructed, NMS can be efficiently performed. Through the analysis of $\mathcal{G}$, we find that it contains many weakly connected components (WCCs), and most of them are small. Based on these two characteristics, we propose two optimization strategies. First, due to the nature of dynamic programming, different WCCs are independent. We can use a divide-and-conquer algorithm to break down the problem into smaller subproblems corresponding to these WCCs and solve them recursively. Inspired by quicksort, we propose quicksort induced NMS (QSI-NMS), which provides $6.18\times$ speed with a negligible 0.1% decrease in mAP compared to original NMS in YOLOv8-N [21] on MS COCO 2017 [24]. Furthermore, by analyzing the structure of QSI-NMS, we propose extended QSI-NMS (eQSI-NMS) with a complexity of $\mathcal{O}(n \log n)$, achieving state-of-the-art performance. Second, leveraging the locality suppression characteristic of NMS, where most weakly connected components are small, we exclude boxes that cannot have suppression relationships through geometric analysis. This led to the development of boxes outside excluded NMS (BOE-NMS), which provides $5.12\times$ speed with no compromise in mAP compared to original NMS in YOLOv8-N on MS COCO 2017.

To facilitate the evaluation and comparison of NMS algorithms, we introduce NMS-Bench, the first end-to-end benchmark for rapid NMS validation. By decoupling model inference and post-processing, we save substantial computational resources, enabling NMS validation to be completed within minutes. Moreover, by implementing NMS algorithms fairly within this framework, different NMS algorithms can be compared on an equal footing. Thus, we integrate data, benchmarking methods, and evaluation metrics into a single framework, enabling end-to-end rapid validation and simplifying NMS research for researchers.

In summary, our contributions are as follows:

- We present the first comprehensive analysis of the NMS algorithm from a graph theory perspective, uncovering the intrinsic structure of NMS;

- We propose two efficient NMS algorithms based on the properties of the NMS-induced graph;

- We introduce NMS-Bench, the first end-to-end benchmark for rapid NMS validation.

2 Problem Definition

Original NMS, employs the intersection over union (IOU) between bounding boxes as the criterion for mutual suppression. Specifically, Given a set of candidate bounding boxes $\mathcal{B}$, original NMS selects the box b^* with the highest confidence score from $\mathcal{B}$, removes it from $\mathcal{B}$, and adds it to the final output set $\mathcal{D}$. Then, it computes IOUs between b^* and all other boxes in $\mathcal{B}$. If the IOU with a certain box b is greater than a given threshold N_t, then b is removed from $\mathcal{B}$. This process is repeated until $\mathcal{B}$ is empty.

In general, NMS during post-processing is an algorithm, which takes a list of detection bounding boxes $\mathcal{B}$ with corresponding confidence scores $\mathcal{S}$ as input, and outputs a subset $\mathcal{D}$ of $\mathcal{B}$. And for convenience, we denote the cardinality of $\mathcal{B}$ by n, i.e., $n = |\mathcal{B}|$. Formally, the NMS algorithm takes

2

$(\mathcal{B}, \mathcal{S})$ as input, and outputs a sequence $K = (k_1, k_2, \ldots, k_n)$, where

$$\begin{cases} k_i = 1 & \text{if } b_i \in \mathcal{D}; \\ k_i = 0 & \text{otherwise.} \end{cases}$$

And an evaluation function is a mapping $e : \{0, 1\}^n \mapsto \mathbb{R}$, where a larger value of e indicates a better NMS. The goal of our research is to enhance algorithm efficiency under the condition that

$$e(K_{origin}) - e(K) < \varepsilon,$$

where $\varepsilon > 0$ represents the tolerance factor and K_{origin} is the output of original NMS algorithm. In the object detection tasks of this paper, we use mAP as the evaluation function e, and we set ε to 1%.

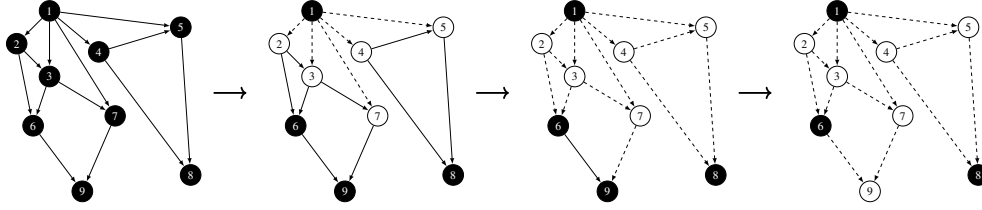



Figure 1: Dynamic programming in topological sorting. The color of the node represents the δ value, i.e., black represents 1, and white represents 0. Before suppression, each node is black. In topological sorting, traversed arcs are represented by dashed lines, showing they have been removed from the graph. After the topological sorting is completed, we can find that nodes 1, 6, and 8 are all black, that is, the last boxes retained are b_1, b_6, and b_8.

3 A Graph Theory Perspective

The bottleneck of NMS algorithms lies in the extensive computation of IOUs. In practice, many IOUs are smaller than a given threshold N_t and will not have any suppressive effect. We aim to consider only those IOUs that will affect the final result, thereby reducing the number of computations and improving efficiency. An IOU greater than N_t indicates that two boxes have a suppressive effect on each other; otherwise, they are independent. We can treat this relationship as an edge in a graph, with each box as a node. This graph reflects the intrinsic structure of NMS, representing the connections between all boxes. By this transformation, we can directly analyze the NMS algorithm through the graph. Compared to a set of boxes in a two-dimensional plane, the structure of the graph is clearer and has more properties that can be utilized.

Specifically, we can regard the input $\mathcal{B}, \mathcal{S}, N_t$ of NMS algorithms as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. That's because we can think of every box in $\mathcal{B}$ as a node in a graph and draw an arc from v to u if box v can suppress box u. Here, we give a formal definition.

Definition 1. *Given a 3-tuple $(\mathcal{B}, \mathcal{S}, N_t)$ consisting of the bounding boxes, confidence scores and an IOU threshold, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ induced by NMS described as follows, there is an injective mapping of $\mathcal{B}$ into $\mathcal{V}$ that maps each bounding box b_v in $\mathcal{B}$ to a node $v \in \mathcal{V}$, and for any ordered pair (v, u),*

$$arc\ (v, u) \in \mathcal{E} \iff s_v > s_u \wedge IOU(b_v, b_u) > N_t.$$

Proposition 1. *$\mathcal{G}$ is a directed acyclic graph (DAG).*

We prove Proposition 1 in the Appendix. Since $\mathcal{G}$ is a DAG, we can use dynamic programming to get the answer to NMS, i.e., K. Specifically, let $\delta(v)$ be the result of node v, i.e., v is retained if $\delta(v) = 1$, otherwise it's not. In original NMS, if there is a node v that can suppress the current node u, then u will not be retained. Therefore, we have the dynamic programming equation as follows,

$$\delta(u) = \begin{cases} \neg \left(\bigvee_{v,(v,u)\in\mathcal{E}} \delta(v) \right) & \text{if } d^-(u) > 0; \\ 1 & \text{otherwise,} \end{cases}$$

3

where $d^-(u)$ denotes the in-degree of u.

Theorem 1. $\forall k_i = K_{origin}[i]$, *we have*

$$k_i = \delta(i).$$

Theorem 1 shows that we can actually obtain the result through dynamic programming in topological sorting, shown in Figure 1. Because the result of DP depends only on valid topological sorts, which indicates that we do not need to sort confidence scores in descending order like original NMS to get the same answer, as long as the topological sort is valid. Additionally, we can observe that if there is no path from node v to node u, then v does not influence u. From this, we derive Corollary 1.

Corollary 1. *If v and u are in two different weakly connected components (WCCs) of $\mathcal{G}$, then $\delta(v)$ and $\delta(u)$ are independent.*

(a)

(b)

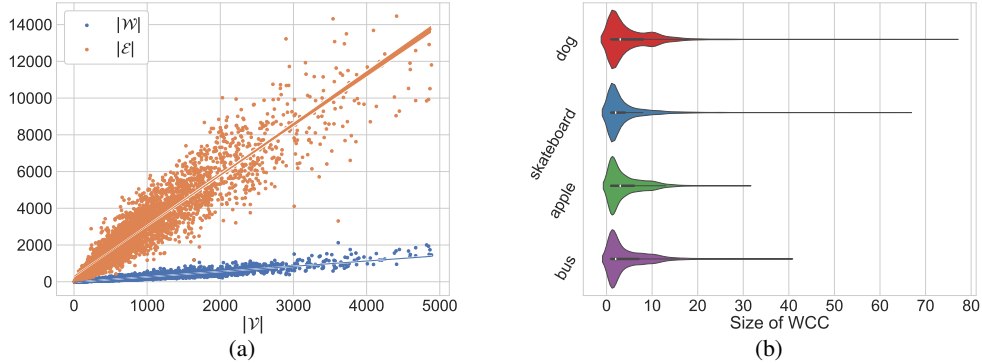



Figure 2: Statistical characteristics of graph $\mathcal{G}$ on MS COCO 2017 validation. 2(a) The scatter plot of 5000 $\mathcal{G}$s on MS COCO 2017. It indicates that the number of arcs $|\mathcal{E}|$ and the number of WCCs $|\mathcal{W}|$ exhibit an approximately linear relationship with the number of nodes $|\mathcal{V}|$, respectively. 2(b) The violin plot of the sizes of WCCs across different categories on MS COCO 2017. It reveals the distributional characteristics of the sizes of the WCCs. It shows that over 50% of the WCCs have a size less than 5, and more than 75% have a size less than 10.

We find that completing dynamic programming in topological sorting requires $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$ time. In real-world data, $|\mathcal{E}|$ appears to have a linear relationship to $|\mathcal{V}|$ (see Figure 2(a)), so once $\mathcal{G}$ is determined, NMS can be highly efficient via DP. However, quickly determining $\mathcal{G}$ is not a simple task. This is because, given a bounding box b, it is difficult to quickly determine which boxes in $\mathcal{B}$ have an IOU $> N_t$ with it. A related problem is improving the efficiency of the k-nearest neighbors algorithm (kNN), where [22, 10, 19, 1] have made significant progress. However, IOU is more complex than the distance defined by norms, and we can only approximate $\mathcal{G}$ through related algorithms. We tried the latest research [10], but it provided little help in acceleration due to its large constant.

Fortunately, The NMS task is quite special, as its input comes from well-trained models, meaning that bounding boxes will cluster around many possible object locations, and bounding boxes predicted as different objects are independent of each other. This implies that $\mathcal{G}$ is a sparse graph with many WCCs, as shown in Figure 2(a). Additionally, we find that most of the WCCs are quite small, as shown in Figure 2(b). These two observations respectively suggest two optimization strategies (see Figure 3). Firstly, because WCCs are independent of each other, we can use a divide-and-conquer algorithm to break down many WCCs into fewer WCCs, continuously reducing the problem size to improve computational efficiency. Thus, we design QSI-NMS. Secondly, because most WCCs are quite small in size, we can reduce the cost of constructing arcs by geometric knowledge, leading to the design of BOE-NMS.

4 Methodology

Following the graph-theoretic analysis of NMS in Section 3, we propose two optimization methods based on two distinct characteristics of graph $\mathcal{G}$. Our approach is to design algorithms through the

4

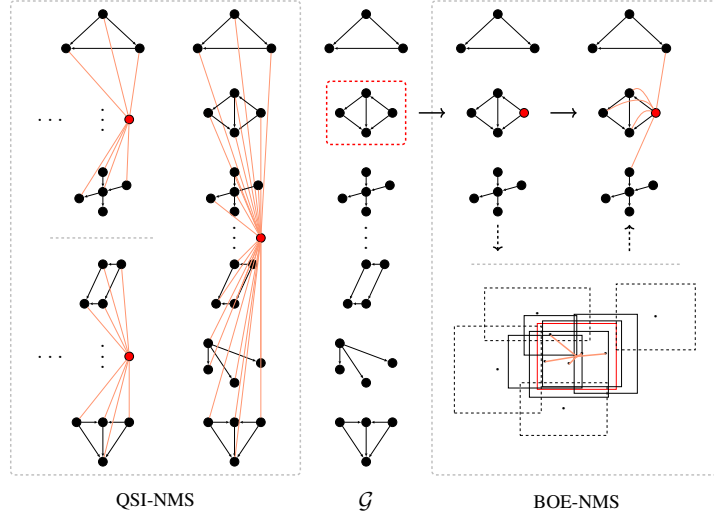
analysis of these characteristics to quickly construct $\mathcal{G}$ or an approximate graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}}) \approx \mathcal{G}$, enabling the use of dynamic programming in topological sorting to obtain NMS results.

QSI-NMS   $\mathcal{G}$   BOE-NMS

Figure 3: The key ideas behind QSI-NMS (left) and BOE-NMS (right). $\mathcal{G}$ (middle) contains many small weakly connected components (WCCs). QSI-NMS considers the global structure of the graph $\mathcal{G}$, where there are many WCCs. It selects a pivot (the red node on the left) and computes IOUs (orange edges) with all current subproblem nodes using a divide-and-conquer algorithm. BOE-NMS focuses on the local structure (the red dashed box) of $\mathcal{G}$, where most WCCs are quite small in size. It selects a node (the red node on the right) and only computes IOUs (orange edges) with its nearby nodes (solid arrows), which is derived from 2D plane geometric analysis (dashed arrows).

4.1 QSI-NMS

We observe that graph $\mathcal{G}$ contains many WCCs, and according to Corollary 1, these components do not affect each other. This implies that, unlike the original NMS, which processes bounding boxes sequentially after sorting by confidence scores and is therefore very slow, we can solve the problem more efficiently using a divide-and-conquer algorithm, breaking it down into independent subproblems that can be solved recursively. Inspired by quicksort, we design quicksort induced NMS (QSI-NMS).

In each subproblem on $\mathcal{B}$, we can similarly select a pivot and calculate IOUs between the pivot and all the other boxes in $\mathcal{B}$, thereby constructing some arcs in $\mathcal{G}$. Next, we devise a partitioning criterion to split $\mathcal{B} \setminus \{\text{pivot}\}$ into two disjoint sets, $\mathcal{B}_l$ and $\mathcal{B}_r$, which are then solved recursively. Since IOUs are not calculated between boxes in $\mathcal{B}_l$ and $\mathcal{B}_r$, some arcs in the original $\mathcal{G}$ might be missed. Therefore, we need to carefully choose the pivot and partitioning criterion to ensure that the constructed $\tilde{\mathcal{G}}$ is as similar to $\mathcal{G}$ as possible.

For the pivot selection, we need to define a priority to choose the best pivot in $\mathcal{B}$. We find that selecting nodes with an in-degree of 0 in $\mathcal{G}$ is optimal for two main reasons. First, node v_0 with an in-degree of 0 belongs to some WCCs, and since most nodes in a WCC predict the same object, v_0 with the maximum confidence score will suppress most nodes, meaning it has many outgoing arcs. Choosing other nodes in the WCC might allocate v_0's successors to different subsets, leading to significant discrepancies between $\mathcal{G}$ and $\mathcal{G}'$. Second, according to De Morgan's laws, the value of $\delta(u)$ is essentially the conjunction of the negations of the predecessors' δ values, formally described as follows:

$$\delta(u) = \bigwedge_{v, (v,u) \in \mathcal{E}} \neg\delta(v)$$

This implies that missing an arc (v_0, v) could result in $\delta(v)$ being incorrectly computed as 1, causing a chain reaction that significantly deviates K from K_{origin}. According to Definition 1, the node

5

$v^* \in \mathcal{V}$ corresponding to the box $b^* \in \mathcal{B}$ with the highest confidence score has an in-degree of 0. Hence, we select the box b^* with the highest confidence score as the pivot.

For the partitioning criterion, we need to consider the spatial characteristics of different WCCs. Different WCCs are relatively dispersed in 2D space, so we can define the partitioning criterion based on the positions of the boxes. We represent the position of a box by its centroid, as it is intuitive and representative. Since the centroid is an ordered pair (x, y), we can not compare it directly like real numbers. We need to define a preorder in $\mathbb{R}^2$. We find that different preorders have negligible effects on mAP. See Appendix C.4 for details. We finally adopt the Manhattan distance to the origin $O(0, 0)$, i.e., the L^1 norm, as the comparison standard. Formally, we define a homogeneous relation $\preceq_M$ on $\mathbb{R}^2$:

$$(x_1, y_1) \preceq_M (x_2, y_2) \Leftrightarrow |x_1| + |y_1| \leq |x_2| + |y_2|.$$

Finally, we partition the set $\mathcal{B} \setminus \{b^*\}$ as follows:

$$\begin{cases} \mathcal{B}_l = \{b_c | b_c \preceq_M b_c^* \wedge b \in \mathcal{B} \setminus \{b^*\}\}; \\ \mathcal{B}_r = \{b_c | b_c \npreceq_M b_c^* \wedge b \in \mathcal{B} \setminus \{b^*\}\}, \end{cases}$$

where b_c and b_c^* denote the centroid of b and b^*, respectively. Since we always choose the box with the highest confidence score, this creates a valid topological sort of $\tilde{\mathcal{G}}$. Thus, we can avoid explicitly constructing $\tilde{\mathcal{G}}$, further reducing computational overhead. The pseudo-code for QSI-NMS can be found in the Appendix.

eQSI-NMS Taking $\mathcal{O}(n \log n)$ Time   Though QSI-NMS performs very well in the real world, it is not an $\mathcal{O}(n \log n)$ algorithm for the simple reason that the pivot is not chosen randomly. By analyzing the structure of QSI-NMS, we further optimize it and propose extended QSI-NMS (eQSI-NMS), which only takes $\mathcal{O}(n \log n)$ time. Since in carrying out QSI-NMS we always split the problem into two subproblems, we can thus construct a binary tree.

Definition 2. *Given a 3-tuple $(\mathcal{B}, \mathcal{S}, N_t)$, a QSI-tree for $\mathcal{B}$ denoted by $QT(\mathcal{B})$ is a binary tree defined recursively as follow:*

- *Its root is a node corresponding to the box $b_v \in \mathcal{B}$ with maximum confidence s_v.*

- *Its left subtree is $QT(\mathcal{B}_l)$, where $\mathcal{B}_l$ is the left subset of $\mathcal{B}$ in QSI-NMS.*

- *Its right subtree is $QT(\mathcal{B}_r)$, where $\mathcal{B}_r$ is the right subset of $\mathcal{B}$ in QSI-NMS.*

The basic case is that if $\mathcal{B}$ is empty, then QSI-tree is also empty, i.e., $QT(\emptyset) = \emptyset$.

An example of QSI-tree is shown in Figure 4. QSI-tree reveals the inherent structure of QSI-NMS, allowing us to consider QSI-NMS from a high-level perspective. More generally, in QSI-NMS, we tag each box with an ordered pair (p, c), where $p \in \mathcal{P}$ represents the priority and $c \in \mathcal{C}$ is the key used for partitioning. We define preorder relations $\preceq_{\mathcal{P}}$ on $\mathcal{P}$ and $\preceq_{\mathcal{C}}$ on $\mathcal{C}$. This indicates that the QSI-tree is essentially a binary search tree that satisfies the max-heap property: the priority of the parent node is not less than that of the child nodes, the keys in the left subtree are all less than or equal to the parent node, and the keys in the right subtree are all greater than the parent node. Furthermore, we have the following Theorem 2.

Theorem 2. *We sort all the elements of $\mathcal{B}$ in ascending order of boxes' centroids according to the preorder $\preceq_{\mathcal{C}}$ into a sequence:*

Figure 4: A Cartesian tree for B. The x-axis represents the centroid, where the node on the left $\preceq_{\mathcal{C}}$ the one on the right. The y-axis represents the confidence score, where the node below $\preceq_{\mathcal{P}}$ the one above. The values of the sequence below the x-axis are the confidence scores of B.

$$B = (b_{i_1}, b_{i_2}, \ldots, b_{i_n}),$$

then QSI-tree is a Cartesian tree for B in which each key is the confidence score of the corresponding box.

6

According to the dynamic programming, if a node v can affect the result $\delta(u)$ of node u, there must exist a path between v and u. In QSI-NMS, this manifests as node v only being able to influence nodes within its subtrees in QSI-tree. Theorem 2 states that QSI-tree is a Cartesian tree, indicating that the subtree of v corresponds to a contiguous interval in B, as shown in Figure 4.

Specifically, the subtree of v corresponds to the contiguous interval $B[l_v + 1 : r_v - 1]$ in B, where l_v is the last position before v that is greater than s_v, and r_v is the first position after v that is greater than s_v. Finding (l_v, r_v) for all v is known as the all nearest greater values problem, which can be solved in $\mathcal{O}(n)$ time by maintaining a stack. Similarly to QSI-NMS, we can complete the suppression during the algorithm. Therefore, we obtain the following time complexity:

$$\mathcal{O}(n \log n + n) = \mathcal{O}(n \log n).$$

According to our best knowledge, this algorithm is the most optimal in terms of complexity. The pseudo-code can also be found in the Appendix.

4.2 BOE-NMS

We find that the vast majority of WCCs in $\mathcal{G}$ are very small, as shown in Figure 2(b). This is because there are not many bounding boxes predicting the same object, and NMS is a form of local suppression. We hope to consider the locality of box distributions, so that the currently selected box only computes IOUs with boxes corresponding to nodes in the same WCC, rather than computing IOUs with all boxes as in original NMS.

We focus on the spatial locality of boxes. We found that a box is likely to have large IOUs only with neighbors that are relatively close to it in 2D space, which also indicates that $\mathcal{G}$ is a sparse graph. Formally, we have the following theorem:

Theorem 3. *Given a bounding box $b^* \in \mathcal{B}$, $\forall b \in \mathcal{B}$, we have $IOU(b^*, b) \leq \frac{1}{2}$ if the centroid of b does not lie within b^*. Formally,*

$$\left(x_c^{(b)} > x_{rb}^{(b^*)} \vee x_c^{(b)} < x_{lt}^{(b^*)} \right) \vee \left(y_c^{(b)} > y_{rb}^{(b^*)} \vee y_c^{(b)} < y_{lt}^{(b^*)} \right),$$

where $(x_c^{(b)}, y_c^{(b)})$, $(x_{lt}^{(b^*)}, y_{lt}^{(b^*)})$ and $(x_{rb}^{(b^*)}, y_{rb}^{(b^*)})$ denote the coordinates of the centroid of b, the left-top and the right-bottom corners of b^*, respectively.

Since N_t is usually greater than 0.5, e.g., 0.7 for YOLOv8 and Faster R-CNN. By Theorem 3 we can check IOUs only for those boxes whose centorids lie within the current box. Based on this, we propose boxes outside excluded NMS (BOE-NMS), a method devoid of mAP loss.

In BOE-NMS, We first sort the boxes by their centroids according to lexicographic order $\preceq_L$ on $\mathbb{R}^2$ which is defined as follows:

$$(x_1, y_1) \preceq_L (x_2, y_2) \Leftrightarrow (x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 \leq y_2).$$

Then for the current box b^*, we can find all the boxes whose centroids may lie in b^* in $\mathcal{O}(\log n)$ time, and we just need to check one by one whether the IOUs between b and these boxes are greater than N_t. The pseudo-code for BOE-NMS is described in Algorithm 3 which can be found in the Appendix. Let's set aside Theorem 3 for now. A more intuitive but weaker conclusion is that if two boxes do not intersect, their IOU must be 0. However, this is not conducive to efficient implementation because of the high cost of maintaining the corresponding data structure. We discuss this issue in the Appendix.

N_t is typically set to 0.7, and the method based on Theorem 3 does not introduce errors. We also provide a tighter bound to further optimize BOE-NMS. Based on Theorem 4 which is a generalization of Theorem 3, we can handle cases where N_t is any real number $\in (0, 1)$.

Theorem 4. *We use s to denote a scaling factor, and then we can use $\alpha(b, s)$ to represent the new box b' obtained by scaling b. Formally,*

$$\begin{cases} x_{lt}^{(b')} = x_c^{(b)} - s \times |x_{lt}^{(b)} - x_c^{(b)}|, \\ x_{rb}^{(b')} = x_c^{(b)} + s \times |x_{rb}^{(b)} - x_c^{(b)}|, \\ y_{lt}^{(b')} = y_c^{(b)} - s \times |y_{lt}^{(b)} - y_c^{(b)}|, \\ y_{rb}^{(b')} = y_c^{(b)} + s \times |y_{rb}^{(b)} - y_c^{(b)}|. \end{cases}$$

Given any $N_t \in (0, 1)$, if the centroid of b does not lie within $\alpha(b^*, 1/N_t - 1)$, then $IOU(b, b^*) \leq N_t$.

7

Since BOE-NMS only excludes boxes with IOU $\leq N_t$, the graph constructed by the BOE-NMS is the same as $\mathcal{G}$. In other words, the results of BOE-NMS are identical to original NMS. However, unlike original NMS, BOE-NMS does not need to compute IOUs with all remaining boxes but rather determines the boxes that could potentially be suppressed in $\mathcal{O}(\log n)$ time. Next, inspect each of these t ($t \approx$ size of the corresponding WCC) boxes one by one in $\mathcal{O}(t)$ time. As shown in Figure 2(b), the sizes of weakly connected components are almost all less than a constant, say 10. This means that the actual performance of BOE-NMS approaches linear time complexity, but strictly speaking, the complexity is still $\mathcal{O}(n^2)$.

5 Experiments

In this section, we first introduce NMS-Bench, the first end-to-end benchmark for rapid validation of NMS algorithms. Next, we validate our algorithms on NMS-Bench and compare them with classical algorithms: original NMS [7], Fast NMS [4], and Cluster-NMS [46]. We conduct tests on MS COCO 2017 [24] and Open Images V7 [23] using YOLOv5 [20], YOLOv8 [21], and Faster R-CNN [12] as validation models. More experimental details can be found in the Appendix.

5.1 NMS-Bench

NMS-Bench is a robust framework that allows researchers to evaluate various NMS methods over different models and datasets in a few minutes. NMS-Bench primarily consists of three components: original bounding box data without NMS applied, implementations of various NMS algorithms as benchmarking methods, and evaluation metrics. The code for NMS-Bench is available on GitHub[†].

For the original boxes, we extracted non-NMS boxes using different models (YOLO series [31, 21] and Faster R-CNN [12]) on various datasets [24, 23] to create the NMS-Bench dataset, thereby decoupling the model inference and post-processing stages. This approach saves significant computational resources during inference. We provide a large amount of data for testing, including original boxes from a total of 273,100 images. More detailed information can be found in the Appendix.

For benchmarking methods, NMS-Bench implements classical algorithms such as original NMS [7], Fast NMS [4], Cluster-NMS [46], and PSRR-MaxpoolNMS [43]. QSI-NMS (including eQSI-NMS) and BOE-NMS are also included in NMS-Bench. These methods enable researchers to reproduce and study NMS algorithms. All algorithms are implemented fairly. Researchers can also quickly implement and validate their own NMS algorithms, as NMS-Bench is a plug-and-play, end-to-end benchmark.

For evaluation metrics, we use COCO-style mAP as the accuracy metric and average processing latency per image as the efficiency metric. The latency calculation begins from the input of bounding boxes and ends when the retained bounding boxes are output. For a dataset containing N images, latency is measured by using the bounding boxes generated per image as input, and the total latency for the N images is averaged. To mitigate random errors, this measurement is repeated 5 times, and the average of these measurements is used as the final average latency.

5.2 Results

In Table 1, we compare our methods with some mainstream work on MS COCO 2017. We observe that our methods, particularly eQSI-NMS, demonstrate substantial performance enhancements in processing speed across different models on MS COCO 2017. eQSI-NMS stands out by offering up to $16.9\times$ speed of original NMS, $4.3\times$ speed of Fast NMS, and $8.9\times$ speed of Cluster-NMS with a mAP decrease of about 0.5%. Similarly, QSI-NMS provides $8.8\times$ speed of original NMS, $2.2\times$ speed of Fast NMS, and $4.6\times$ speed of Cluster-NMS with a marginal mAP decrease of about 0.2%. BOE-NMS also shows significant enhancements, being $9.1\times$ as fast as original NMS, $2.3\times$ as fast as Fast NMS, and $4.8\times$ as fast as Cluster-NMS with no mAP decrease.

Table 2 shows that on Open Images V7, eQSI-NMS provides approximately $10.2\times$ speed of original NMS, $3.7\times$ speed of Fast NMS, and $7.0\times$ speed of Cluster-NMS. QSI-NMS is about $5.6\times$ as fast as original NMS, $2.0\times$ as fast as Fast NMS, and $3.9\times$ as fast as Cluster-NMS. Similarly, BOE-NMS

[†]https://github.com/Yuri3-xr/NMS-Bench

8

achieves $5.4\times$ speed of original NMS, $2.0\times$ speed of Fast NMS, and $3.8\times$ speed of Cluster-NMS. On Open Images V7, QSI-NMS and eQSI-NMS perferm well in mAP with about 0.2% mAP decreasing.

Table 1: NMS Methods Performance on MS COCO 2017

Model	Size	Target	original NMS	Fast NMS	Cluster-NMS	BOE-NMS	QSI-NMS	eQSI-NMS
YOLOv8	N	Average Latency (μs)	906.9	321.4	600.8	176.8	146.8	**85.0**
		AP 0.5:0.95 (%)	37.2	37.0	37.2	37.2	37.1	36.9
	S	Average Latency (μs)	531.2	232.5	421.5	126.1	109.4	**63.4**
		AP 0.5:0.95 (%)	44.8	44.6	44.8	44.8	44.6	44.5
	M	Average Latency (μs)	353.3	202.6	348.5	100.8	93.1	**53.7**
		AP 0.5:0.95 (%)	50.2	50.0	50.2	50.2	50.0	49.9
	L	Average Latency (μs)	196.5	51.3	90.7	82.1	67.1	**39.0**
		AP 0.5:0.95 (%)	52.8	52.6	52.8	52.8	52.7	52.5
	X	Average Latency (μs)	183.0	148.6	262.2	69.2	66.8	**39.6**
		AP 0.5:0.95 (%)	53.9	53.7	53.9	53.9	53.8	53.6
YOLOv5	N	Average Latency (μs)	10034.2	1724.2	3949.1	719.6	688.9	**339.0**
		AP 0.5:0.95 (%)	27.8	27.6	27.8	27.8	27.5	27.4
	S	Average Latency (μs)	4441.4	996.4	2152.5	438.1	449.2	**226.5**
		AP 0.5:0.95 (%)	37.2	36.9	37.2	37.2	36.9	36.6
	M	Average Latency (μs)	3351.6	874.1	1851.2	350.5	408.3	**204.9**
		AP 0.5:0.95 (%)	45.1	44.8	45.1	45.1	44.9	44.5
	L	Average Latency (μs)	2531.2	732.8	1484.2	286.3	353.3	**178.4**
		AP 0.5:0.95 (%)	48.8	48.4	48.8	48.8	48.6	48.2
	X	Average Latency (μs)	1959.1	630.8	1273.9	248.5	314.7	**160.3**
		AP 0.5:0.95 (%)	50.5	50.1	50.5	50.5	50.3	49.9
Faster R-CNN R50-FPN	-	Average Latency (μs)	57.2	112.0	184.4	41.1	36.6	**25.7**
		AP 0.5:0.95 (%)	39.8	39.9	39.8	39.8	39.5	39.3
Faster R-CNN R101-FPN	-	Average Latency (μs)	49.5	100.2	175.8	37.1	33.9	**23.9**
		AP 0.5:0.95 (%)	41.8	41.7	41.8	41.8	41.5	41.4
Faster R-CNN X101-FPN	-	Average Latency (μs)	39.7	95.8	169.7	31.9	30.1	**21.4**
		AP 0.5:0.95 (%)	43.0	42.8	43.0	43.0	42.7	42.5

Table 2: NMS Methods Performance on Open Images V7

Model	Size	Target	original NMS	Fast NMS	Cluster-NMS	BOE-NMS	QSI-NMS	eQSI-NMS
YOLOv8	N	Average Latency (μs)	1627.9	498.1	952.4	260.2	231.8	**132.3**
		AP 0.5:0.95 (%)	18.1	18.2	18.1	18.1	18.1	18.0
	S	Average Latency (μs)	1212.1	412.8	781.6	214.7	199.5	**111.2**
		AP 0.5:0.95 (%)	27.3	27.2	27.3	27.3	27.1	27.1
	M	Average Latency (μs)	1003.3	371.6	744.0	191.9	189.0	**100.6**
		AP 0.5:0.95 (%)	33.1	33.1	33.1	33.1	33.0	32.9
	L	Average Latency (μs)	853.1	350.5	665.6	175.0	180.3	**100.8**
		AP 0.5:0.95 (%)	34.4	34.4	34.4	34.4	34.3	34.2
	X	Average Latency (μs)	803.4	342.0	648.8	168.5	173.2	**95.1**
		AP 0.5:0.95 (%)	35.9	35.8	35.9	35.9	35.8	35.7

Table 3: Comparisons of Our Methods and PSRR-MaxpoolNMS

Model	Size	Target	original NMS	PSRR-MaxpoolNMS	BOE-NMS	QSI-NMS	eQSI-NMS
YOLOv5	N	Average Latency (μs)	8568.5	599.6	906.2	628.0	**325.6**
		AP 0.5:0.95 (%)	27.8	26.5	27.8	27.5	27.4
	S	Average Latency (μs)	3858.2	409.4	547.7	408.2	**217.5**
		AP 0.5:0.95 (%)	37.2	35.6	37.2	36.9	36.6
	M	Average Latency (μs)	2918.1	380.5	424.7	371.3	**197.4**
		AP 0.5:0.95 (%)	45.1	43.1	45.1	44.9	44.5
Faster R-CNN R50-FPN	-	Average Latency (μs)	53.0	89.0	43.1	34.3	**24.6**
		AP 0.5:0.95 (%)	39.8	37.5	39.8	39.5	39.3
Faster R-CNN R101-FPN	-	Average Latency (μs)	45.5	86.4	38.6	31.9	**23.0**
		AP 0.5:0.95 (%)	41.8	39.5	41.8	41.5	41.4
Faster R-CNN X101-FPN	-	Average Latency (μs)	37.0	86.9	33.4	28.6	**20.6**
		AP 0.5:0.95 (%)	43.0	40.5	43.0	42.7	42.5

MaxpoolNMS [5] and ASAP-NMS [38] are only applicable to the first stage of two-stage detectors, while the problem we are discussing is more general, so we do not include them in our comparison. We compare our methods with PSRR-MaxpoolNMS [43], which is applicable to anchor-based models. We conduct experiments on anchor-based models (Faster R-CNN and YOLOv5) using MS COCO 2017, and the results are shown in Table 3. As we can see, eQSI-NMS achieves the lowest latency while maintaining a favorable trade-off with mAP. However, PSRR-MaxpoolNMS experiences a $1 \sim 2\%$ mAP loss in the Faster R-CNN and YOLOv5 models.

In the case of Faster R-CNN, the latency performance of PSRR-MaxpoolNMS is not competitive. This is because PSRR-MaxpoolNMS requires 8 max-pooling operations, which, although not affecting the

9

algorithm's complexity, introduces a large constant factor that hampers efficiency when the number of bounding boxes is small (e.g., the average number of bounding boxes in the three Faster R-CNN models is less than 300). However, it performs well when the number of bounding boxes is large (e.g., YOLOv5-S has an average of 2898 bounding boxes). This demonstrates that the speedup of PSRR-MaxpoolNMS is highly dependent on the degree of parallelism, whereas our methods directly reduce computational overhead (see Figure 5 in Appendix D.3), making it hardware-agnostic and suitable for resource-constrained edge devices.

6 Related Work

NMS algorithm is widely used in object detection tasks [35, 31, 34]. Original NMS [7] operates on a greedy principle, suppressing bounding boxes with an Intersection over Union (IOU) higher than a given threshold, starting from the ones with the highest confidence scores. On one hand, numerous improvements have been made to NMS to achieve higher mAP in certain scenarios [3, 29, 16, 25, 47, 18, 37]. On the other hand, some research focuses on enhancing speed. Fast NMS [4] improves NMS efficiency by avoiding the sequential processing of bounding boxes that need to be suppressed, making it more conducive to parallel computing and thus speeding up the process, though it may slightly reduce accuracy compared to original NMS. Cluster-NMS [46], employs matrix operations and iterative processing, running the Fast NMS algorithm in each iteration to accelerate the original NMS without compromising accuracy. MaxpoolNMS [5] and ASAP-NMS [38] take into account the setting of "anchors" in the region proposal network (RPN) of two-stage detectors. MaxpoolNMS maps anchors of different sizes onto several score maps and performs spatial max-pooling on these score maps to avoid calculating IOUs, thereby improving the speed of NMS. ASAP-NMS eliminates some boxes with relatively small IOUs by precomputing the IOUs between the current box and neighboring anchors. PSRR-MaxpoolNMS [43] improves upon MaxpoolNMS by introducing Relationship Recovery, which addresses the issue of score map mismatch that may arise in MaxpoolNMS, enabling PSRR-MaxpoolNMS to be used in the second stage of two-stage detectors. CUDA NMS by torchvision [28] is a CUDA implementation of the original NMS, leveraging a GPU to accelerate computation-intensive tasks, though it cannot be used in scenarios without a GPU.

7 Conclusion

In this paper, we systematically analyze the NMS algorithm from a graph theory perspective and discover strong connections between NMS, directed graph topological sorting, dynamic programming, and weak connected components. Through these analyses, we first propose QSI-NMS, a fast divide-and-conquer algorithm with negligible loss, and its extended version, eQSI-NMS, achieves the state-of-the-art complexity $\mathcal{O}(n \log n)$. Additionally, starting from the sparsity of graphs, we design BOE-NMS, which considers the locality suppression feature of NMS, optimizes the NMS algorithm at a constant level, and maintains precision. Furthermore, we introduce NMS-Bench, the first end-to-end benchmark integrating bounding box datasets, NMS benchmarking methods, and evaluations, facilitating NMS research for researchers. Finally, we conducted experiments on NMS-Bench, and the experimental results validated our theory, demonstrating the superiority of our algorithms.

Acknowledgements   This work was supported in part by the National Natural Science Foundation of China under Grant 62192781, Grant 62172326 and Grant 62137002, and in part by the Project of China Knowledge Centre for Engineering Science and Technology.

References

[1] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms – improving object detection with one line of code. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5561–5569, 2017.

10

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166, 2019.

[5] Lile Cai, Bin Zhao, Zhe Wang, Jie Lin, Chuan Sheng Foo, Mohamed Sabry Aly, and Vijay Chandrasekhar. Maxpoolnms: Getting rid of nms bottlenecks in two-stage object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9348–9356, 2019.

[6] Yue Cao, Junchi Bin, Jozsef Hamari, Erik Blasch, and Zheng Liu. Multimodal object detection by channel switching and spatial attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 403–411, 2023.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[8] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021.

[9] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2009.

[10] Yury Elkin and Vitaliy Kurlin. A new near-linear time algorithm for k-nearest neighbor search using a compressed cover tree. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 9267–9311, 2023.

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[14] Mei Han, Amit Sethi, Wei Hua, and Yihong Gong. A detection-based multiple object tracking method. In *Proceedings of the International Conference on Image Processing*, volume 5, pages 3065–3068, 2004.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[16] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2883–2892, 2019.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[18] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4507–4515, 2017.

[19] Mike Izbicki and Christian Shelton. Faster cover trees. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1162–1170, 2015.

[20] Glenn Jocher. YOLOv5 by Ultralytics, May 2020.

[21] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, January 2023.

11

[22] Thomas Kollar. Fast nearest neighbors. Technical report, Computer Science and Artificial Intelligence Lab, 2006.

[23] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 2020.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755, 2014.

[25] Songtao Liu, Di Huang, and Yunhong Wang. Adaptive nms: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6459–6468, 2019.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37, 2016.

[27] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[28] TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library, November 2016.

[29] Chengcheng Ning, Huajun Zhou, Yan Song, and Jinhui Tang. Inception single shot multibox detector for object detection. In *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops*, pages 549–554, 2017.

[30] Rui Qian, Xin Lai, and Xirong Li. 3d object detection for autonomous driving: A survey. *Pattern Recognition*, 130:108796, 2022.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[32] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[35] A. Rosenfeld and M. Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, C-20(5):562–569, 1971.

[36] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19:221–248, 2017.

[37] Charalampos Symeonidis, Ioannis Mademlis, Ioannis Pitas, and Nikos Nikolaidis. Efficient feature extraction for non-maximum suppression in visual person detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5, 2023.

[38] Rohun Tripathi, Vasu Singla, Mahyar Najibi, Bharat Singh, Abhishek Sharma, and Larry Davis. Asap-nms: Accelerating non-maximum suppression using spatially aware priors. *arXiv preprint arXiv:2007.09785*, 2020.

12

[39] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 390–391, 2020.

[40] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *Advances in Neural Information Processing Systems*, volume 33, pages 17721–17732, 2020.

[41] Yingkun Xu, Xiaolong Zhou, Shengyong Chen, and Fenfen Li. Deep learning for multiple object tracking: a survey. *IET Computer Vision*, 13(4):355–368, 2019.

[42] Ruixin Yang and Yingyan Yu. Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. *Frontiers in Oncology*, 11:638182, 2021.

[43] Tianyi Zhang, Jie Lin, Peng Hu, Bin Zhao, and Mohamed M. Sabry Aly. Psrr-maxpoolnms: Pyramid shifted maxpoolnms with relationship recovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15840–15848, 2021.

[44] Zhihao Zhang, Shengcao Cao, and Yu-Xiong Wang. Tamm: Triadapter multi-modal learning for 3d shape understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21413–21423, 2024.

[45] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2024.

[46] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wang-meng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586, 2021.

[47] Penghao Zhou, Chong Zhou, Pai Peng, Junlong Du, Xing Sun, Xiaowei Guo, and Feiyue Huang. Noh-nms: Improving pedestrian detection by nearby objects hallucination. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1967–1975, 2020.

13

Appendix

Table of Contents

A   Preliminaries   **15**

   A.1   Graph Theory   15

   A.2   Cartesian Tree   15

B   Proofs   **16**

   B.1   Proof of Proposition 1   16

   B.2   Proof of Theorem 1   16

   B.3   Proof of Theorem 2   16

   B.4   Proof of Theorem 3   16

   B.5   Proof of Theorem 4   17

   B.6   Proof of Proposition 3   19

C   Discussion   **20**

   C.1   Discussion of Fast NMS and Cluster-NMS   20

   C.2   Discussion of MaxpoolNMS, ASAP-NMS, and PSRR-MaxpoolNMS   21

   C.3   Discussion on mAP loss of QSI-NMS   22

   C.4   Further Discussion of QSI-NMS and eQSI-NMS   24

   C.5   Further Discussion of BOE-NMS   24

D   Experimental Details   **26**

   D.1   More Information about NMS-Bench   26

   D.2   Experimental Environment and Settings   26

   D.3   More Results   27

E   Pseudo-Codes   **29**

   E.1   Pseudo-Code for QSI-NMS   29

   E.2   Pseudo-Code for eQSI-NMS   30

   E.3   Pseudo-Code for BOE-NMS   31

14

A Preliminaries

A.1 Graph Theory

Notations and Terminology   A directed graph $\mathcal{G}$, also called a digraph, is an ordered pair $(\mathcal{V}, \mathcal{E})$ consisting of a nonempty set $\mathcal{V}$ of nodes, a set $\mathcal{E}$ of arcs. An arc, also called an arrow, is an ordered pair (v, u) of nodes, where $v, u \in \mathcal{V}$. v and u are the head and tail of the arc (v, u), respectively. We say two nodes v, u are adjacent if there exists an arc $e \in \mathcal{E}$ such that $e = (v, u) \vee e = (u, v)$. A direct predecessor of node v is the head of an arc whose tail is v, and a direct successor is the tail of an arc whose head is v. The in-degree $d^-(v)$ of a node $v \in \mathcal{V}$ is the number of arcs with tail v while the out-degree $d^+(v)$ is the number of arcs with head v. A walk $(v_0, v_1, \ldots, v_k)$ in a directed graph is a sequence of nodes satisfying $(v_i, v_{i+1}) \in \mathcal{E}$ for all $i = 0, 1, \ldots, k-1$. A path in a directed graph is a walk in which all vertices are distinct. A cycle is a path $(v_0, v_1, \ldots, v_k)$ together with the arc (v_k, v_0). We say a graph $\mathcal{G}$ is simple if there are no multi-edges or self-loops in $\mathcal{G}$. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains a graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ if $\mathcal{V}' \subset \mathcal{V} \wedge \mathcal{E}' \subset \mathcal{E}$, we then say $\mathcal{G}'$ is a subgraph of $\mathcal{G}$ and denote $\mathcal{G}' \subset \mathcal{G}$.

We next introduce several key concepts.

Definition 3 (Directed acyclic graph). *A directed acyclic graph (DAG) is a directed graph without cycles.*

Definition 4 (Strongly connected). *A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called strongly connected if it contains a path between v, u and a path between u, v, for every pair $v, u \in \mathcal{V}$.*

Definition 5 (Weakly connected). *A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is weakly connected if the symmetric graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ with $\mathcal{E}' = \mathcal{E} \cup \{(u, v) | (v, u) \in \mathcal{E}\}$ is strongly connected.*

Definition 6 (Strongly connected component). *For a directed graph, an inclusion-maximal strongly connected subgraph is called strongly connected component (SCC).*

Definition 7 (Weakly connected component). *For a directed graph, an inclusion-maximal weakly connected subgraph is called weakly connected component (WCC).*

A.2 Cartesian Tree

Definition 8. *A Cartesian tree for a sequence is a binary tree constructed as follows,*

- *The root of the tree is the maximum element of the sequence.*

- *Its left and right subtrees are formed by recursively constructing Cartesian tree for the subsequences to the left and right.*

The basic case is that if the sequence is empty, then the Cartesian tree is also empty.

During the construction process of a Cartesian tree, the maximum value is selected from the current sequence as the root each time. Thus, the Cartesian tree obeys the max-heap property whose inorder traversal returns the original sequence. The complexity of constructing a Cartesian tree according to Definition 8 is $\mathcal{O}(n^2)$ in the worst-case which is too slow. Fortunately, we have the following proposition:

Proposition 2. *Given a sequence S of length n, a Cartesian tree for S can be constructed in $\mathcal{O}(n)$ time.*

Here, we provide an informal proof. We can establish Proposition 2 by designing an algorithm with $\mathcal{O}(n)$ complexity. We consider adding elements from S one by one to construct the Cartesian tree. Assume that we have already constructed the Cartesian tree for $S[1 : k-1]$. Now, when adding $S[k]$, we observe that $S[k]$ will be the rightmost node of the tree. Furthermore, since Cartesian trees are max-heaps, this implies that the parent of $S[k]$ must be greater than or equal to $S[k]$. Therefore, we need to find the rightmost node v in the Cartesian tree that is greater than or equal to $S[k]$, and assign its right child u as the left child of $S[k]$, updating the right child of v to be $S[k]$. We can implement this algorithm efficiently by maintaining a stack: pop the stack until the stack top is not less than $S[k]$, and then update the tree structure accordingly. After this, push $S[k]$ onto the stack. When k is iterated from 1 to n, the construction of the Cartesian tree is also completed. Since each element is pushed onto and popped from the stack at most once, the overall complexity is $\mathcal{O}(n)$.

15

B Proofs

B.1 Proof of Proposition 1

Proof. Assume that there is cycle in the directed graph $\mathcal{G}$ with length k, i.e., there exists a sequence

$$(v_0, v_1, v_2, \ldots, v_k),$$

where $v_0 = v_k$ and $\forall i = 0, 1, \ldots, k-1, (v_i, v_{i+1}) \in \mathcal{E}$.

By definition, we know that if $(v, u) \in \mathcal{E}$, then $b_v > b_u$. Therefore,

$$b_{v_0} > b_{v_1} > b_{v_2} > \ldots > b_{v_k} = b_{v_0},$$

which demonstrates that

$$b_{v_0} > b_{v_0}.$$

This is contradictory to the fact that b_{v_0} is a real number. □

B.2 Proof of Theorem 1

Proof. We first sort the scores in $\mathcal{S}$ in descending order into a sequence $S : (s_{i_1}, s_{i_2}, \ldots, s_{i_n})$. To prove Theorem 1, we prove that statement $\delta(i_t) = k_{i_t}$ holds for all $t = 1, 2, \ldots, n$ by mathematical induction.

For $t = 1$, $k_{i_1} = 1$, that's because no box can suppress b_{i_1} with the maximum score s_{i_1}. $\delta(i_1) = 1 = k_{i_1}$ since the in-degree of i_1 is 0. Therefore, the statement is true for $t = 1$.

Assume it is true for integers $1, 2, \ldots, t-1, t \geq 2$, then we turn our attention to t. If $k_{i_t} = 1$, it implies that there does not exist a $p < t$ such that retained b_{i_p} can suppress b_{i_t}, i.e., $\text{IOU}(b_{i_t}, b_{i_p}) \leq N_t \vee k_{i_p} = 0$. Therefore, if there is an arc from i_p to i_t, then $\delta(i_p) = k_{i_p} = 0$, which indicates $\delta(i_t) = 1$. Similarly, we can prove that $\delta(i_t) = 0$ if $k_{i_t} = 0$.

By mathematical induction, the statement above is true for $1, 2, \ldots, n$. □

B.3 Proof of Theorem 2

Proof. We denote $|\mathcal{B}|$ by n, we prove Theorem 2 by mathematical induction.

For $n = 1$, the statement is true since there is only one node in both binary trees.

Suppose that the statement holds for all sets with size $1, 2, \ldots, t$. Consider any set $\mathcal{B}$ with $n = (t+1)$ elements.

The root of $QT(\mathcal{B})$ corresponds b_v with maximum s_v in $\mathcal{B}$ ($QT(\emptyset) = \emptyset$). And the root of the Cartesian tree corresponds the maximum element of the sequence B, which implies it's also b_v indexed by m, i.e., $i_m = v$.

According to Definition 2, the left child of $QT(\mathcal{B})$'s root is $\mathcal{B}_l$'s root which satisfies

$$c_u \preceq_\mathcal{C} c_v \text{ for all } b_u \in \mathcal{B}_l,$$

where each c_u is the centroid of the corresponding box b_u, and c_v is the centroid of b_v.

Notice that B is sorted in ascending order of centroids according to the order $\preceq_\mathcal{C}$, which demonstrates that the elements in the set $\mathcal{B}_l$ are the same to that in the sequence $B[1 : m-1]$. Formally, $|\mathcal{B}_l| = m-1$ and $\forall b_v \in \mathcal{B}_l$, there exists one and only one $x \in \{1, 2, \ldots, m-1\}$ such that $v = i_x$, i.e., $b_v = b_{i_x}$.

According to the hypothesis, $QT(\mathcal{B}_l)$ is a Cartesian tree for $B[1 : m-1]$. Similarly, we can prove $QT(\mathcal{B}_r)$ is a Cartesian tree for $B[m+1 : t+1]$.

Hence, the statement is true for $n = (t+1)$, which completes the proof. □

B.4 Proof of Theorem 3

Proof. We can take b as the frame of reference. As illustrated in Figure 5, the blue box is b, and the green one is b^* which is respectively positioned to the left (Figure 5(a)) or right (Figure 5(b)) of the

16

(a) Left   (b) Right   (c) Above   (d) Below

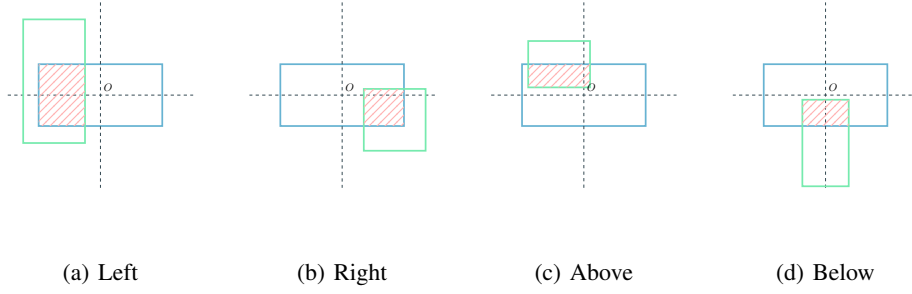



Figure 5: Four positions of b^* relative to b.

vertical dashed line, or above (Figure 5(c)) or below (Figure 5(d)) the horizontal dashed line. The intersections are filled with red north east lines. Hence, we have

$$\text{IOU}(b^*, b) = \text{IOU}(b, b^*)$$
$$= \frac{\text{Area(red)}}{\text{Union}(b, b^*)}$$
$$\leq \frac{1/2 \text{Area}(b)}{\text{Area}(b)}$$
$$= \frac{1}{2}.$$

$\square$

B.5 Proof of Theorem 4

We prove Theorem 4 by demonstrating the inequalities in the following two lemmas.

Lemma 1. *Given positive real numbers θ and β, for any $x_1, x_2, y_1, y_2 \in \mathbb{R}$, the following inequality holds:*
$$\theta(x_2 - x_1) + \beta(y_2 - y_1) \geq (\theta + \beta)(\min\{x_2, y_2\} - \max\{x_1, y_1\}).$$

Lemma 2. *Given a positive real number γ, for any $x_1, x_2, y_1, y_2 \in \mathbb{R}$ such that*
$$(\gamma + 1)x_2 + (1 - \gamma)x_1 \leq y_1 + y_2,$$

the following inequality holds:
$$x_2 + y_2 - x_1 - y_1 \geq (2 + \gamma)(\min\{x_2, y_2\} - \max\{x_1, y_1\}).$$

Proof of Lemma 1. Notice that $\min\{x_2, y_2\} \leq x_2$ and $\min\{x_2, y_2\} \leq y_2$, we have:
$$\min\{x_2, y_2\} \leq \frac{\theta}{\theta + \beta} x_2 + \frac{\beta}{\theta + \beta} y_2.$$

For $\max\{x_1, y_1\}$, we have:
$$\max\{x_1, y_1\} \geq \frac{\theta}{\theta + \beta} x_1 + \frac{\beta}{\theta + \beta} y_1.$$

Therefore,
$$RHS = (\theta + \beta)(\min\{x_2, y_2\} - \max\{x_1, y_1\})$$
$$\leq (\theta + \beta)(\frac{\theta}{\theta + \beta} x_2 + \frac{\beta}{\theta + \beta} y_2 - \frac{\theta}{\theta + \beta} x_1 - \frac{\beta}{\theta + \beta} y_1)$$
$$= (\theta + \beta)(\frac{\theta}{\theta + \beta}(x_2 - x_1) + \frac{\beta}{\theta + \beta}(y_2 - y_1))$$
$$= \theta(x_2 - x_1) + \beta(y_2 - y_1)$$
$$= LHS.$$

$\square$

17

Proof of Lemma 2. Similar to the approach used in the proof above, for $\max\{x_1, y_1\}$, we have

$$\max\{x_1, y_1\} \geq \frac{\gamma}{2+\gamma}x_1 + \frac{2}{2+\gamma}y_1.$$

Notice that $\min\{x_2, y_2\} \leq x_2$, then we have

$$\begin{aligned}
LHS &= x_2 - x_1 + (y_1 + y_2) - 2y_1 \\
&\geq x_2 - x_1 + (\gamma + 1)x_2 + (1 - \gamma)x_1 - 2y_1 \\
&= (2 + \gamma)x_2 - \gamma x_1 - 2y_1 \\
&= (2 + \gamma)(x_2 - \frac{\gamma}{2+\gamma}x_1 - \frac{2}{2+\gamma}y_1) \\
&\geq (2 + \gamma)(\min\{x_2, y_2\} - \max\{x_1, y_1\}) \\
&= RHS.
\end{aligned}$$

$\square$

We next use these two inequalities to prove Theorem 4.

Proof of Theorem 4. For convenience, we denote a bounding box b by a 4-tuple $(x_{lt}^{(b)}, x_{rb}^{(b)}, y_{lt}^{(b)}, y_{rb}^{(b)})$, where $(x_{lt}^{(b)}, y_{lt}^{(b)})$ and $(x_{rb}^{(b)}, y_{rb}^{(b)})$ represent the left-top and right-bottom corners of b, respectively. This says

$$\begin{cases} x_{lt}^{(b)} \leq x_{rb}^{(b)}, \\ y_{lt}^{(b)} \leq y_{rb}^{(b)}. \end{cases}$$

Given bounding boxes b^* and b, their intersection can be calculated as follows:

$$\begin{aligned}
\text{Inter}(b^*, b) = &\max\{0, \min\{x_{rb}^{(b^*)}, x_{rb}^{(b)}\} - \max\{x_{lt}^{(b^*)}, x_{lt}^{(b)}\}\} \\
&\times \max\{0, \min\{y_{rb}^{(b^*)}, y_{rb}^{(b)}\} - \max\{y_{lt}^{(b^*)}, y_{lt}^{(b)}\}\}.
\end{aligned}$$

Here, b^* is represented by $(x_{lt}^{(b^*)}, x_{rb}^{(b^*)}, y_{lt}^{(b^*)}, y_{rb}^{(b^*)})$ and b by $(x_{lt}^{(b)}, x_{rb}^{(b)}, y_{lt}^{(b)}, y_{rb}^{(b)})$. Let

$$\begin{cases} I_x = \min\{x_{rb}^{(b^*)}, x_{rb}^{(b)}\} - \max\{x_{lt}^{(b^*)}, x_{lt}^{(b)}\}, \\ I_y = \min\{y_{rb}^{(b^*)}, y_{rb}^{(b)}\} - \max\{y_{lt}^{(b^*)}, y_{lt}^{(b)}\}. \end{cases}$$

We observe that if $I_x \leq 0$ or $I_y \leq 0$, then $\text{Inter}(b^*, b) = 0$, resulting in IOU $= 0$. This is a trivial case, and Theorem 4 holds. Therefore, we only need to consider the case where $I_x > 0$ and $I_y > 0$. In this case,

$$\text{Inter}(b^*, b) = I_x I_y.$$

Then the union can be expressed as:

$$\begin{aligned}
\text{Union}(b^*, b) &= \text{Area}(b^*) + \text{Area}(b) - \text{Inter}(b^*, b) \\
&= L_x^{(b^*)} L_y^{(b^*)} + L_x^{(b)} L_y^{(b)} - I_x I_y,
\end{aligned}$$

where

$$\begin{cases} L_x^{(b^*)} = x_{rb}^{(b^*)} - x_{lt}^{(b^*)}; \\ L_y^{(b^*)} = y_{rb}^{(b^*)} - y_{lt}^{(b^*)}; \\ L_x^{(b)} = x_{rb}^{(b)} - x_{lt}^{(b)}; \\ L_y^{(b)} = y_{rb}^{(b)} - y_{lt}^{(b)}, \end{cases}$$

represent the width and height of b^* and b, respectively. We then have the following inequality holds:

$$\frac{L_x^{(b^*)} L_y^{(b^*)} + L_x^{(b)} L_y^{(b)}}{I_x I_y} \geq \frac{1}{N_t} + 1. \tag{1}$$

Let $\theta = L_x^{(b^*)} > 0, \beta = L_x^{(b)} > 0$, according to Lemma 1 and $I_y > 0$, then we have

$$\frac{L_x^{(b^*)} L_y^{(b^*)} + L_x^{(b)} L_y^{(b)}}{I_y} \geq L_x^{(b^*)} + L_x^{(b)}. \tag{2}$$

18

Furthermore, according to the proof of Theorem 3, $\alpha(b^*, 1/N_t - 1)$ is respectively positioned to the left or right of the vertical dashed line, or above or below the horizontal dashed line (see Figure 5). Without loss of generality, we can assume that $\alpha(b^*, 1/N_t - 1)$ is to the left of vertical dashed line passing through the centroid of b. This means:

$$
\begin{aligned}
\frac{(x_{lt}^{(b)} + x_{rb}^{(b)})}{2} &\geq x_c^{(b^*)} + s \times |x_{rb}^{(b^*)} - x_c^{(b^*)}| \\
&= (x_{lt}^{(b^*)} + x_{rb}^{(b^*)})/2 + (1/N_t - 1) \times (x_{rb}^{(b^*)} - x_{lt}^{(b^*)})/2 \\
&= \frac{(1/N_t)x_{rb}^{(b^*)} + (2 - 1/N_t)x_{lt}^{(b^*)}}{2}.
\end{aligned}
$$

let γ be $1/N_t - 1 > 0$, according to Lemma 2 and $I_x > 0$, then we have

$$
\frac{L_x^{(b^*)} + L_x^{(b)}}{I_x} \geq 2 + (\frac{1}{N_t} - 1) = \frac{1}{N_t} + 1. \tag{3}
$$

Combining inequalities (2) and (3), inequality (1) is proven.

Finally, according to the calculation method of IOU, we have:

$$
\begin{aligned}
\text{IOU}(b^*, b) &= \frac{\text{Inter}(b^*, b)}{\text{Union}(b^*, b)} \\
&= \frac{I_x I_y}{L_x^{(b^*)} L_y^{(b^*)} + L_x^{(b)} L_y^{(b)} - I_x I_y} \\
&= \frac{1}{\frac{L_x^{(b^*)} L_y^{(b^*)} + L_x^{(b)} L_y^{(b)}}{I_x I_y} - 1} \\
&\leq \frac{1}{(\frac{1}{N_t} + 1) - 1} \\
&= N_t.
\end{aligned}
$$

$\square$

B.6 Proof of Proposition 3

Proof. Since a point can be considered as an interval of length 0, we can construct the set $\mathcal{M}'$ in $\mathcal{O}(n)$ time:

$$
\mathcal{M}' = \{[m_i, m_i] \mid m_i \in \mathcal{M}\}.
$$

Suppose there is an algorithm A that can solve problem X. This means:

$$
\mathcal{Q}_X = A(\mathcal{I}, [l, r]).
$$

Then, we can spend $\mathcal{O}(1)$ time to call A once, using $\mathcal{M}'$ and $[l, r]$ as its inputs. Therefore:

$$
\mathcal{Q}_Y = A(\mathcal{M}', [l, r]).
$$

Therefore, problem Y can be reduced to X in polynomial time, i.e., $Y \leq_P X$. $\square$

19

C Discussion

From a graph theory perspective, we revisited the NMS algorithm and discovered statistical properties of the weakly connected components in graph $\mathcal{G}$: they are numerous overall but small in size locally. Leveraging these two characteristics, we proposed QSI-NMS and BOE-NMS. We then validated the efficiency and accuracy of QSI-NMS and BOE-NMS on NMS-Bench, showing significant improvements over the original NMS, parallelized Fast NMS, and Cluster-NMS. However, there are several areas for improvement in our work.

First, our NMS algorithms can be combined with other accuracy-enhancing NMS methods, such as Soft-NMS [3], to address the negligible accuracy loss introduced by QSI-NMS and eQSI-NMS. Since our algorithms serve as general frameworks, allowing other methods to be implemented by modifying the dynamic programming equation. Second, our algorithms can be further parallelized to improve efficiency. QSI-NMS uses a divide-and-conquer recursive strategy, enabling parallel processing of different subproblems, while BOE-NMS can adopt the approach of Fast NMS, where each box computes local IOUs in parallel. Third, the distribution characteristics of bounding boxes can be further studied to obtain more detailed information about graph $\mathcal{G}$, aiding in the analysis of accuracy loss in Fast NMS, QSI-NMS, and eQSI-NMS. These can be explored in future work.

In the latter part of this section, we delve deeper into our work. In Subsection C.1, we conduct a thorough graph theory analysis of Fast NMS and Cluster-NMS; in Subsection C.2, we compare our methods with MaxpoolNMS, ASAP-NMS, and PSRR-MaxpoolNMS; in Subsection C.3, we explain and analyze the reasons behind the slight mAP drop of QSI-NMS; in Subsection C.4, we explore some implementation details of QSI-NMS; and in Subsection C.5, we discuss the advantages of the BOE-NMS implementation.

C.1 Discussion of Fast NMS and Cluster-NMS

Discussion of Fast NMS In Fast NMS [4], the computation of IOU is parallelized, and when the IOU exceeds a threshold N_t, the bounding box with the higher confidence score always suppresses the one with the lower confidence score. This allows the results for all bounding boxes to be computed in parallel without depending on the results of previous boxes. Formally, in Fast NMS, the set $\mathcal{B}$ is first sorted into B by confidence scores in descending order, and then an $n \times n$ matrix $\mathbf{Y}$ is defined as follows:

$$y_{i,j} = \begin{cases} \text{IOU}(b_i, b_j) & \text{if } i < j; \\ 0 & \text{otherwise.} \end{cases}$$

For a bounding box b_j, if there exists a bounding box b_i with higher confidence $(i < j)$ such that $y_{i,j} > N_t$, then b_j is not retained; otherwise, it is retained. Since N_t effectively classifies all IOUs into two categories, for ease of analysis, we define a matrix $\mathbf{X}$ where:

$$x_{i,j} = \begin{cases} 1 & \text{if } i < j \text{ and } y_{i,j} > N_t; \\ 0 & \text{otherwise.} \end{cases}$$

Thus, we have

$$k_j = \neg \left(\bigvee_{i<j} x_{i,j} \right).$$

We can observe that $\mathbf{X}$ is the adjacency matrix of $\mathcal{G}$. Fast NMS can also obtain the same result using dynamic programming, but the dynamic programming equation is as follows:

$$\delta(u) = \begin{cases} 0 & \text{if } d^-(u) > 0; \\ 1 & \text{otherwise.} \end{cases}$$

It is evident that Fast NMS suppresses more bounding boxes compared to original NMS, which explains the loss in mAP observed with Fast NMS. Additionally, this demonstrates that Fast NMS can achieve its results without the need for topological sorting, explaining why Fast NMS is parallelizable. However, the actual implementation of Fast NMS has a time complexity of $\Theta(n^2)$, which is, in fact, greater than that of original NMS. In original NMS, suppressed boxes are not used in subsequent IOU calculations, whereas in Fast NMS, every box calculates IOUs with all preceding boxes. This highlights the heavy reliance of Fast NMS on efficient parallel computing.

20

Discussion of Cluster-NMS In Cluster-NMS [46], the issue of accuracy loss is addressed by iterating Fast NMS multiple times. We use a binary vector $\boldsymbol{r}$ to represent the result of applying Fast NMS to $\mathbf{X}$, denoted as $\boldsymbol{r} = F(\mathbf{X})$, where:

$$\boldsymbol{r}_j = \neg \left(\bigvee_{i<j} x_{i,j} \right).$$

The iterative process of Cluster-NMS is as follows:

1. $\mathbf{X}^{(0)} = \mathbf{X}$;
2. $\boldsymbol{r}^{(t)} = F(\mathbf{X}^{(t)})$, for $t \geq 0$;
3. $\mathbf{X}^{(t+1)} = \mathrm{diag}(\boldsymbol{r}^{(t)}) \times \mathbf{X}$;
4. Iterate until $\boldsymbol{r}^{(t)}$ converges to $\boldsymbol{r}^*$, at which point $\|\boldsymbol{r}^{(t)} - \boldsymbol{r}^*\| = 0$.

In step 3, the iterative method for $\mathbf{X}^{(t+1)}$ involves left-multiplying $\mathbf{X}$ by a diagonal 01 matrix, meaning rows corresponding to 1 are retained and those corresponding to 0 are discarded. Formally expressed as follows:

$$x_{i,j}^{(t+1)} = \begin{cases} x_{i,j} & \text{if } \boldsymbol{r}_i^{(t)} = 1; \\ 0 & \text{otherwise.} \end{cases}$$

This shows that only boxes not suppressed in the current iteration can cause a suppression effect in the next iteration. Thus, bounding boxes incorrectly suppressed in this round will be re-evaluated with truly retained boxes in the next iteration to determine if they should indeed be suppressed.

[46] proves that its results are equivalent to original NMS and usually requires only a few iterations. To discuss the number of iterations, [46] defines a cluster:

Definition 9 (cluster in Cluster-NMS [46]). *A subset $\mathcal{U} = \{b_{j_1}, b_{j_2}, \ldots, b_{j_{|\mathcal{U}|}}\}$ of $\mathcal{B}$ is a cluster if and only if for all $b_{j_t} \in \mathcal{U}$, there exists $i \in \{j_1, j_2, \ldots, j_{|\mathcal{U}|}\} \setminus \{j_t\}$ such that $IOU(b_{j_t}, b_i) > N_t$ and for all $b \in \mathcal{B} \setminus \mathcal{U}$, $IOU(b_{j_t}, b) \leq N_t$.*

[46] proves that the number of iterations does not exceed the size of the largest cluster.

Analyzing Cluster-NMS from the perspective of graph $\mathcal{G}$, each iteration essentially determines whether all current nodes with an in-degree of 0 should be retained, i.e., the value of $\delta(\cdot)$; then it traverses all outgoing arcs of some node v with in-degree 0 to decide if a successor node u should be suppressed, updating $\delta(u)$ as follows:

$$\delta(u) \leftarrow \delta(u) \wedge \neg\delta(v).$$

Finally, all outgoing arcs are deleted, and the next iteration begins. Hence, the essence of Cluster-NMS is parallel topological sorting within each WCC. In fact, we can see that the boxes in a cluster correspond to a WCC in $\mathcal{G}$. This explains why the number of iterations until convergence does not exceed the size of the largest cluster: in each iteration, at least one node with an in-degree of 0 is expanded and added to the topological sort within each WCC, reducing its size by at least one per iteration. Although Cluster-NMS is correct, the matrix $\mathbf{X}$ obtained by parallel IOU computation already encodes all the information of $\mathcal{G}$, which indicates a single dynamic programming can quickly produce results identical to original NMS.

C.2 Discussion of MaxpoolNMS, ASAP-NMS, and PSRR-MaxpoolNMS

Discussion of MaxpoolNMS and ASAP-NMS MaxpoolNMS [5] and ASAP-NMS [38] take into account the information of pre-defined "anchors" in the RPN and leverage the locality suppression characteristics of NMS to achieve impressive performance improvements. However, they have the following limitations:

- MaxpoolNMS and ASAP-NMS can only be used in the first stage of two-stage detectors. Our methods, however, can be used in any stage of any detector because we address the most general case (see Section 2).

21

- The complexity of MaxpoolNMS is $\mathcal{O}(n_s n_r \lfloor \frac{W}{\beta} \rceil \lfloor \frac{H}{\beta} \rceil + n \log n)$, where n_s and n_r represent the number of anchor box scales and ratios, respectively. The complexity of ASAP-NMS is $\mathcal{O}(n^2)$. Neither of these methods are more efficient than eQSI-NMS.

- These methods involve many manually defined hyperparameters and are complex to implement, which limits their generalization across different models and datasets. In contrast, our methods require no additional parameters beyond those in original NMS and are easy to implement.

- These methods are not rigorous and can lead to a certain degree of mAP degradation, whereas BOE-NMS is rigorously proven to cause no mAP loss.

Discussion of PSRR-MaxpoolNMS PSRR-MaxpoolNMS [43] introduces Relationship Recovery to address the issue of regression box and score map mismatch in MaxpoolNMS, allowing it to be used at any stage of all anchor-based detectors, and claims to achieve a time complexity of $\mathcal{O}(n)$. However, we do not believe that PSRR-MaxpoolNMS outperforms our methods for the following two reasons:

First, we implement PSRR-MaxpoolNMS and conduct efficiency comparisons. To better illustrate the impact of bounding box count on runtime, we perform experiments on YOLOv5-N, which has the highest number of bounding boxes, as shown in Figure 6(a). Original NMS has the highest time cost due to its quadratic growth. For a clearer comparison between our methods and PSRR-MaxpoolNMS, we exclude original NMS, as shown in Figure 6(b). As the number of boxes increases, PSRR-MaxpoolNMS is faster than BOE-NMS and QSI-NMS but consistently slower than eQSI-NMS.

Second, strictly speaking, the complexity of PSRR-MaxpoolNMS is not $\mathcal{O}(n)$. PSRR-MaxpoolNMS requires prior knowledge of the input image size and generates confidence score maps related to the size of the image. This aspect is not considered in the complexity analysis (whereas our methods are designed and analyzed independently of the image size). During the Channel Recovery stage of PSRR-MaxpoolNMS, the complexity of computing the nearest distances for channel mapping is $\mathcal{O}(n \times n_s \times n_r)$, where n_s and n_r represent the number of anchor box scales and ratios, respectively. These quantities vary with different datasets and detectors and increase as image size and object count increase. The remaining stages: Spatial Recovery, Pyramid MaxpoolNMS, and Shifted MaxpoolNMS can all be completed in $\mathcal{O}(n)$ time. Thus, the overall complexity of PSRR-MaxpoolNMS is $\mathcal{O}(n_s n_r \lfloor \frac{W}{\beta} \rceil \lfloor \frac{H}{\beta} \rceil + n_s n_r n)$.

(a) (b)

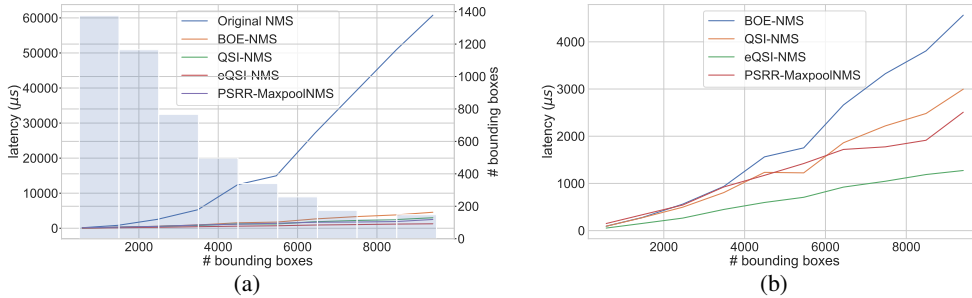



Figure 6: The line plot of the runtime of different methods as the number of bounding boxes varies in YOLOv5-N. 6(a) The histogram with a bin width of 1000 representing the number of bounding boxes in each interval. The input images are divided into 10 intervals based on the number of bounding boxes: $(0, 1000], (1000, 2000], \ldots, (9000, 10000]$. The line plot is drawn with the average number of boxes per interval as the x-coordinate and the average time cost of the NMS algorithms as the y-coordinate. 6(b) The line plot of the runtime of different methods without original NMS.

C.3 Discussion on mAP loss of QSI-NMS

In QSI-NMS, we use a divide-and-conquer strategy, which means that bounding boxes in different subproblems do not affect each other. In some special cases, QSI-NMS may assign nodes from

22

the same WCC to different subproblems, potentially causing some nodes that should have been suppressed to be retained.

We provide a case study with results from the YOLOv8-M model on MS COCO 2017. In Figure 7(a), the blue boxes represent the outputs of original NMS/BOE-NMS, while Figure 7(b) shows the outputs of QSI-NMS, with red boxes indicating additional boxes retained by QSI-NMS. It can be seen that QSI-NMS retains four additional boxes.

For example, consider the box/node numbered 188. The WCC containing this node is shown in Figure 8(a). All other nodes in the WCC would suppress node 188, but when we use $\preceq_M$ to define the partitioning criterion, node 188 ends up in a different subproblem than other nodes, as shown in Figure 8(b). The figure shows a partial structure of the QSI-tree: solid lines indicate parent-child relationships, and dashed lines indicate ancestor-descendant relationships. The red nodes are nodes from the WCC, while the black node 148 is the lowest common ancestor (LCA) of nodes 188 and 201; node 156 is the LCA of nodes 194 and 193. Since each node in this WCC can only be suppressed by its red ancestor nodes, node 188 is not suppressed. However, node 194 is still suppressed because node 201 is its ancestor.

This example highlights the core of QSI-NMS design: the pivot selection and the partitioning criterion. If we choose these two appropriately, the accuracy loss of QSI-NMS can be negligible. In our algorithm design: pivot selection chooses the most representative nodes (with the highest confidence scores), so node 194 is correctly suppressed by node 201 even after being placed in a different subproblem from node 193. The partitioning criterion aims to assign nodes from the same WCC to the same subproblem as much as possible, which helps reduce cases like node 188 being incorrectly retained. We also discuss other partitioning criteria in Appendix C.4.

(a)   (b)

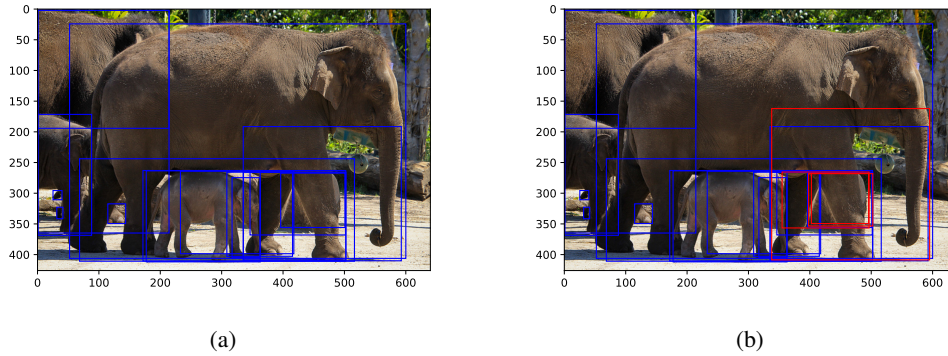



Figure 7: The output bounding boxes of original NMS (7(a)) and QSI-NMS (7(b)) in YOLOv8-M on the MS COCO 2017 image "000000057027.jpg".

(a)   (b)

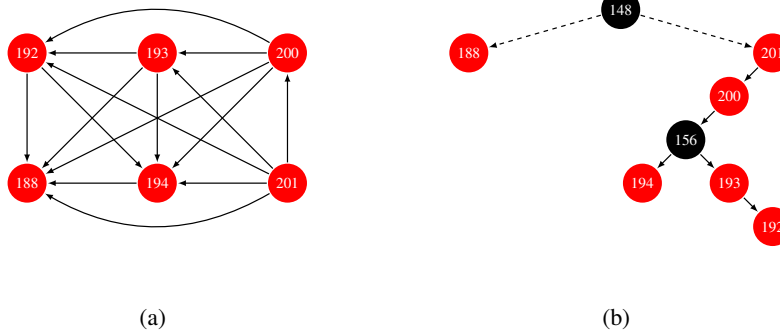



Figure 8: WCC in graph $\mathcal{G}$ contains node 188 (8(a)), along with a partial structure of the QSI-tree (8(b)).

23

C.4 Further Discussion of QSI-NMS and eQSI-NMS

Algorithm 1 and Algorithm 2 describe QSI-NMS and eQSI-NMS respectively, which we use C++ to implement in NMS-Bench.

In the QSI-NMS and eQSI-NMS, we define the order $\preceq_{\mathcal{C}}$, where $\mathcal{C}$ represents the set of centroids of the original boxes. The preorder $\preceq_{\mathcal{C}}$ is defined on $\mathbb{R}^2$. Different preorders result in slight variations in the final graph $\tilde{\mathcal{G}}$, which in turn cause minor differences in the NMS results. Therefore, we conduct comparative experiments on different orders.

We explore three classical orders ($\preceq_L, \preceq_M, \preceq_E$), defined as follows:

$$(x_1, y_1) \preceq_L (x_2, y_2) \Leftrightarrow (x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 \leq y_2)$$
$$(x_1, y_1) \preceq_M (x_2, y_2) \Leftrightarrow |x_1| + |y_1| \leq |x_2| + |y_2|$$
$$(x_1, y_1) \preceq_E (x_2, y_2) \Leftrightarrow \sqrt{x_1^2 + y_1^2} \leq \sqrt{x_2^2 + y_2^2}$$

We conduct tests on the MS COCO 2017 using different weights of YOLOv8, with the results shown in Table 4. We find orders $\preceq_M$ and $\preceq_E$ outperform $\preceq_L$.

Table 4: AP 0.5:0.95 (%) of QSI-NMS and eQSI-NMS under Different Orders on MS COCO 2017

Model	Size	Methods	$\preceq_L$	$\preceq_M$	$\preceq_E$
YOLOv8	N	QSI-NMS	37.0	37.1	37.1
		eQSI-NMS	36.8	36.9	36.9
	S	QSI-NMS	44.5	44.6	44.6
		eQSI-NMS	44.4	44.5	44.5
	M	QSI-NMS	49.9	50.0	50.0
		eQSI-NMS	49.7	49.9	49.9
	L	QSI-NMS	52.5	52.7	52.7
		eQSI-NMS	52.3	52.5	52.5
	X	QSI-NMS	53.6	53.8	53.8
		eQSI-NMS	53.4	53.6	53.6

In the case where weakly connected components are independent of each other, $\preceq_M$ and $\preceq_E$ can better maintain the neighborhood consistency between the boxes. In contrast, $\preceq_L$ may order the boxes of different weakly connected components closer together, thereby disrupting the hierarchical nature of QSI-NMS and resulting in more accuracy loss. Therefore, when selecting the order, it is necessary to consider whether the construction of the order can retain the positional information of the original boxes as much as possible.

C.5 Further Discussion of BOE-NMS

An intuitive conclusion is that if two bounding boxes do not overlap at all, their IOU is necessarily 0. This is a weaker conclusion than Theorems 3 and 4, but it is actually more challenging to implement.

Without loss of generality, we consider the one-dimensional case. Given a set of n intervals $\mathcal{I} = \{[l_1, r_1], [l_2, r_2], \dots, [l_n, r_n]\}$, where $l_i \leq r_i$ for $i = 1, 2, \dots, n$.

We define the following two problems:

Definition 10 (Problem X). *Given $\mathcal{I}$ and an arbitrary interval $[l, r]$ where $l \leq r$, find all intervals in $\mathcal{I}$ that intersect with $[l, r]$, i.e., determine the set $\mathcal{Q}_X$:*

$$\mathcal{Q}_X = \{i \mid 1 \leq i \leq n \wedge [l, r] \cap [l_i, r_i] \neq \emptyset\}.$$

Definition 11 (Problem Y). *Given $\mathcal{I}$ and an arbitrary interval $[l, r]$ where $l \leq r$, find all intervals in $\mathcal{I}$ whose midpoints lie within $[l, r]$, i.e., determine the set $\mathcal{Q}_Y$:*

$$\mathcal{Q}_Y = \{i \mid 1 \leq i \leq n \wedge l \leq \frac{l_i + r_i}{2} \leq r\}.$$

24

In problem X, we need to find all intervals in $\mathcal{I}$ that intersect with the current interval, while in problem Y, we need to find all intervals whose midpoints lie within the current interval. In Problem Y, we only need to consider the set of midpoints, which can be represented as:

$$\mathcal{M} = \{m_i \mid m_i = \frac{l_i + r_i}{2}, \forall i = 1, 2, \ldots, n\}.$$

For problem X, if $i \in \mathcal{Q}_X$, meaning that interval $[l_i, r_i]$ intersects with $[l, r]$, then we have

$$l_i \leq r \wedge r_i \geq l.$$

For problem Y, if $i \in \mathcal{Q}_Y$, then we have

$$l \leq m_i \leq r.$$

Therefore, we find that problem X is more complex than problem Y because, in problem X, we need to maintain the partial order of both endpoints, whereas in Problem Y, we only need to maintain the partial order of the midpoints. Formally, we have the following proposition:

Proposition 3. $Y \leq_P X$.

We prove Proposition 3 in Appendix B.6. Proposition 3 indicates that problem Y can be reduced to problem X. Therefore, X is at least as hard as problem Y. Through the above analysis, we can see that although this intuitive conclusion seems straightforward, it is a more difficult problem than determining whether a point lies within a bounding box. Actually, problem X can essentially be equivalent to a 2D plane point filtering problem. Some algorithms can solve this problem by maintaining data structures with large constants, such as persistent segment trees. However, these methods are complex and offer limited optimization. For BOE-NMS, the comparison between points and segments is quite special. We can use sorting and preprocessing of the point set, and then use binary search based on monotonicity to determine the point set corresponding to the query interval. If the size of the point set corresponding to the query interval is k, it requires $\mathcal{O}(k + \log(n))$ time complexity to obtain the result, avoiding redundant traversal of invalid points. In the other word, our proposed method BOE-NMS more profoundly exploits the properties of weakly connected components.

25

D   Experimental Details

D.1   More Information about NMS-Bench

Table 5 and Table 6 respectively present the number of original bounding boxes after inferences of different models on the MS COCO 2017 and Open Images V7 datasets. A larger number of bounding boxes indicates weaker filtering capabilities of the model, leading to longer post-processing times required for NMS.

Table 5: Number of Bounding Boxes on MS COCO 2017

Model	Size	Number of Bounding Boxes
YOLOv8	N	4,040,118
	S	3,052,794
	M	2,627,367
	L	2,058,360
	X	1,955,594
YOLOv5	N	14,489,236
	S	9,871,207
	M	9,051,451
	L	7,986,223
	X	7,227,635
Faster R-CNN R50-FPN	-	1,256,090
Faster R-CNN R101-FPN	-	1,181,563
Faster R-CNN X101-FPN	-	1,068,974

Table 6: Number of Bounding Boxes on Open Images V7

Model	Size	Number of Bounding Boxes
YOLOv8	N	50,389,365
	S	43,062,828
	M	39,894,713
	L	38,126,739
	X	37,113,747

D.2   Experimental Environment and Settings

Our experimental environment is shown as the Table 7.

Table 7: Experimental Environment

Component	Specification
CPU	
Model	Intel Xeon Gold 6226
Total Cores	12
Total Threads	24
Max Turbo Frequency	3.70 GHz
GPU	
Model	NVIDIA RTX 4090 $\times 1$
VRAM	24 GB GDDR6X

For the hyperparameter settings, we set the NMS threshold N_t to 0.7 in our experiments.

26

D.3 More Results

Experiments in Torchvision Library To compare with the CUDA NMS from torchvision [28], we implement our methods as C++ operators under the torchvision library. We then fairly replace the different NMS operator modules for testing. We test on the MS COCO 2017 using different weights of YOLOv8. The experimental setup is the same as previously described, and we set bench size as 20. The experimental results are shown in the Table 8. This demonstrates that our methods provide performance improvements even when compared to highly optimized parallel implementations.

Table 8: NMS Methods Performance under Torchvision Implementation

Model	Size	Target	CUDA NMS	BOE-NMS	QSI-NMS	eQSI-NMS
YOLOv8	N	Average Latency (μs)	343.4	204.2	185.2	**136.3**
		AP 0.5:0.95 (%)	37.4	37.4	37.3	37.1
	S	Average Latency (μs)	302.1	158.3	160.1	**109.4**
		AP 0.5:0.95 (%)	45.0	45.0	44.8	44.7
	M	Average Latency (μs)	301.0	136.7	141.4	**101.3**
		AP 0.5:0.95 (%)	50.3	50.3	50.2	50.1
	L	Average Latency (μs)	284.4	109.1	111.7	**85.2**
		AP 0.5:0.95 (%)	53.0	53.0	52.8	52.7
	X	Average Latency (μs)	284.1	109.9	106.5	**83.5**
		AP 0.5:0.95 (%)	54.0	54.0	53.9	53.8

Statistics of IOU Calculations During the NMS algorithm process, the computational cost of numerous IOU calculations is a performance bottleneck. We compare the number of IOU calculations between our methods and the original NMS. Figure 9 shows the relationship between the number of boxes and the number of IOU calculations for different methods. It can be observed that our methods significantly reduce the number of IOU calculations compared to original NMS, demonstrating the superiority.

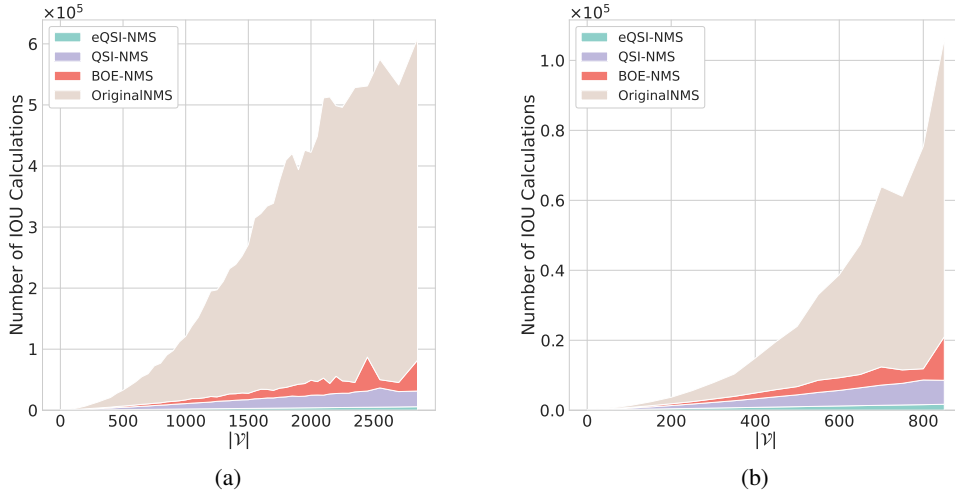



Figure 9: Stackplots of IOU calculations for different methods. 9(a) shows the results of YOLOv8-N on MS COCO 2017, while 9(b) shows the results of Faster R-CNN X101-FPN on MS COCO 2017.

27

Results of Instance Segmentation Tasks We also evaluate our methods on instance segmentation tasks using Mask R-CNN [15] and YOLOv8, where they demonstrate significant superiority over other methods. Please refer to Table 9 for details.

Table 9: NMS Methods on Instance Segmentation Tasks

Model	Size	Target	original NMS	Fast NMS	Cluster-NMS	BOE-NMS	QSI-NMS	eQSI-NMS
Mask R-CNN R50-FPN	-	Average Latency (μs)	48.8	105.7	222.8	31.9	30.3	**21.6**
		AP^{Box} 0.5:0.95 (%)	41.0	40.4	40.9	41.0	40.8	40.4
		AP^{Mask} 0.5:0.95 (%)	37.2	36.9	37.1	37.2	36.9	36.7
Mask R-CNN R101-FPN	-	Average Latency (μs)	45.3	113.1	205.0	32.3	29.8	**21.5**
		AP^{Box} 0.5:0.95 (%)	42.9	42.2	42.9	42.9	42.7	42.3
		AP^{Mask} 0.5:0.95 (%)	38.6	38.4	38.6	38.6	38.4	38.2
Mask R-CNN X101-FPN	-	Average Latency (μs)	40.3	105.6	189.2	26.7	26.6	**19.3**
		AP^{Box} 0.5:0.95 (%)	44.3	43.6	44.2	44.3	44.1	43.7
		AP^{Mask} 0.5:0.95 (%)	39.5	39.3	39.5	39.5	39.3	39.1
YOLOv8	N-SEG	Average Latency (μs)	1265.3	366.4	859.4	219.5	153.4	**85.4**
		AP^{Box} 0.5:0.95 (%)	36.7	36.5	36.7	36.7	36.6	36.4
		AP^{Mask} 0.5:0.95 (%)	30.4	30.4	30.4	30.4	30.3	30.2
	S-SEG	Average Latency (μs)	740.0	269.2	736.2	158.6	115.8	**61.9**
		AP^{Box} 0.5:0.95 (%)	44.7	44.5	44.7	44.7	44.6	44.4
		AP^{Mask} 0.5:0.95 (%)	36.7	36.6	36.7	36.7	36.5	36.5

28

E Pseudo-Codes

E.1 Pseudo-Code for QSI-NMS

Algorithm 1: QSI-NMS

Input : $\mathcal{B} = b_1, \cdots, b_n, \mathcal{C} = c_1, \cdots, c_n, \mathcal{S} = s_1, \cdots, s_n, N_t$
$\mathcal{B}$ is the list of initial detection boxes, $\mathcal{C}$ contains the centroids of the boxes in $\mathcal{B}$, $\mathcal{S}$ contains corresponding detection scores, N_t is the NMS threshold.

Output : $\mathcal{D}$: boxes to be retained.

Function Partition(l, r)
 | $\mathcal{S}' \leftarrow \mathcal{S}_{l \sim r}$;
 | $m \leftarrow \arg\max \mathcal{S}'$;
 | swap(b_m, b_r); swap(s_m, s_r); swap(c_m, c_r);
 | $p \leftarrow l$;
 | **for** $i \in [l, r-1]$ **do**
  | **if** $c_i \preceq_{\mathcal{C}} c_r$ **then**
   | swap(b_p, b_i);
   | swap(s_p, s_i);
   | swap(c_p, c_i);
   | $p \leftarrow p + 1$;
  | **end**
 | **end**
 | swap(b_p, b_r); swap(s_p, s_r); swap(c_p, c_r);
 | **return** p
begin
 | $\mathcal{D} \leftarrow \{\}; \delta \leftarrow \{\text{True}\}^n$;
 | QSI-NMS$(1, n, \delta)$;
 | **return** $\mathcal{D}$
end

Procedure QSI-NMS(l, r, δ)
 | **if** $l \geq r$ **then**
  | **return**
 | **end**
 | $p \leftarrow$ Partition(l, r);
 | **if** $\delta(b_p)$ **then**
  | **for** $i \in [l, r] \setminus \{p\}$ **do**
   | **if** IOU$(b_p, b_i) > N_t$ **then**
    | $\delta(b_p) \leftarrow$ False;
   | **end**
  | **end**
  | $\mathcal{D} \leftarrow \mathcal{D} \cup b_p$;
 | **end**
 | QSI-NMS$(l, p-1, \delta)$;
 | QSI-NMS$(p+1, r, \delta)$;

29

E.2 Pseudo-Code for eQSI-NMS

Algorithm 2: eQSI-NMS

Input $:\mathcal{B} = b_1, \cdots, b_n, \mathcal{C} = c_1, \cdots, c_n, \mathcal{S} = s_1, \cdots, s_n, i_n, N_t$
  $\mathcal{B}$ is the list of initial detection boxes, $\mathcal{C}$ contains the centroids of the boxes in $\mathcal{B}$, $\mathcal{S}$
contains corresponding detection scores, N_t is the NMS threshold.
Output $:\mathcal{D}$: boxes to be retained.
Procedure Solve(I)

   Stack $\mathcal{L}^b \leftarrow [\,]; \mathcal{L}^s \leftarrow [\,]$;
   for $m = 1, 2, \ldots, n$ **do**
      $b^* \leftarrow b_{I_m}$;
      while $\mathcal{L}^s$ *is not empty* $\wedge$ $\mathcal{L}^b$ *is not empty* **do**
         if $\text{TOP}(\mathcal{L}^s) < s_m$ **then**
            if $\text{IOU}(b^*, \text{TOP}(\mathcal{L}^b)) > N_t$ **then**
               $\delta(\text{TOP}(\mathcal{L}^b)) \leftarrow \text{False}$;
            end
            $\text{POP}(\mathcal{L}^b); \text{POP}(\mathcal{L}^s)$;
         end
         else
            Break;
         end
      end
      $\text{PUSH}(\mathcal{L}^b, b^*); \text{PUSH}(\mathcal{L}^s, s_m)$;
   end

begin
   $\mathcal{D} \leftarrow \{\}$; $\delta \leftarrow \{\text{True}\}^n$;
   $C \leftarrow$ the sorted $\mathcal{C}$ in ascending order according to $\preceq_{\mathcal{C}}$;
   $I \leftarrow (i_1, i_2, \ldots, i_n)$ where $C = (c_{i_1}, c_{i_2}, \ldots, c_{i_n})$;
   Solve(I);
   $I \leftarrow \text{reverse}(I)$;
   Solve(I);
   for $b \in \mathcal{B}$ **do**
      if $\delta(b)$ **then**
         $\mathcal{D} \leftarrow \mathcal{D} \cup b$;
      end
   end
   return $\mathcal{D}$
end

30

E.3   Pseudo-Code for BOE-NMS

Algorithm 3: BOE-NMS

Input   : $\mathcal{B} = b_1, \cdots, b_n, \mathcal{M} = m_1, \cdots, m_n, \mathcal{S} = s_1, \cdots, s_n, \mathcal{I} = i_1, \cdots, i_n, N_t$

      $\mathcal{B}$ is the list of initial detection boxes, $\mathcal{M}$ contains the x-coordinates of the centroids of the boxes in $\mathcal{B}$, $\mathcal{S}$ contains corresponding detection scores, $\mathcal{I}$ contains the ranks of all boxes in $\mathcal{B}$, which is sorted by x-coordinate of the centroids of the boxes in ascending order, N_t is the NMS threshold.

Output: $\mathcal{D}$: boxes to be retained.

begin
    $\mathcal{D} \leftarrow \{\}$;
    while $\mathcal{B} \neq \emptyset$ **do**
        $m \leftarrow \arg\max \mathcal{S}$;
        $b^* \leftarrow b_m$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup b^*; \mathcal{B} \leftarrow \mathcal{B} - b^*; \mathcal{S} \leftarrow \mathcal{S} - s_m$;
        $x_l \leftarrow$ left x-coordinate(b^*); $x_r \leftarrow$ right x-coordinate (b^*);
        $l \leftarrow$ lowerbound $(\mathcal{I}, x_l)$; $\triangleright$ Find the rank of the first item i in $\mathcal{I}$, **s.t.**$m_i \geq x_l$
        $r \leftarrow$ upperbound $(\mathcal{I}, x_r)$; $\triangleright$ Find the rank of the first item i in $\mathcal{I}$, **s.t.**$m_i > x_r$
        $\mathcal{I}' \leftarrow \mathcal{I}_{l \sim r-1}$;
        for $i \in \mathcal{I}'$ **do**
            if $\text{IOU}(b^*, b_i) > N_t$ **then**
                $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$;
            end
        end
    end
    return $\mathcal{D}$
end

31

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to Section 1, Paragraph 3, 4, and 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Appendix C, Paragraph 2.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

32

Justification: Please refer to Appendix A and B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: For algorithm implementation details, please refer to Section 4 and Appendix E. And for experimental details, please refer to Section 5 and Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

33

Answer: [Yes]

Justification: We have included the code, related documentation, and licenses in the supplementary material. We will open-source our code and data after acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Section 5 and Appendix D. More details can be found in our source code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Please refer to Table 1, Table 2, Table 8, and Figure 9.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

34

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Table 7 for the CPU/GPU details.

   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in this article fully complies with the NeurIPS Code of Ethics.

   Guidelines:
   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our research on NMS is foundational work in the field of object detection, and does not have any societal impact.

    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

35

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper does not present any such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the relevant models and datasets. Additionally, we provide the corresponding copyrights and licenses in the supplementary material.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

36

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the corresponding instructional documentation and the license in the supplementary material.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

37