Accelerating Non-Maximum Suppression: A Graph Theory Perspective

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

[1]School of Computer Science and Technolog, Xi'an JiaoTong University
[2]Institute of Artificial Intelligence (TeleAI), China Telecom

Introduction

- CNN-based object detection models [2, 3] consist of two parts:

Introduction

- CNN-based object detection models [2, 3] consist of two parts:
  1. model inference

Introduction

- CNN-based object detection models [2, 3] consist of two parts:
    1. model inference
    2. post-processing

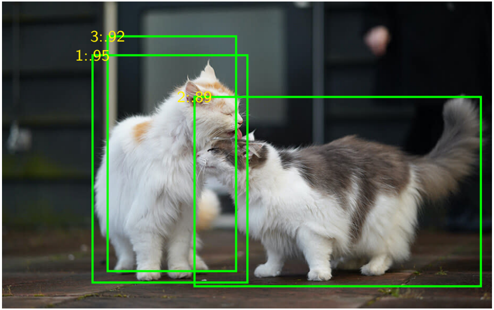
Introduction

- CNN-based object detection models [2, 3] consist of two parts:
  1. model inference
  2. post-processing
- Non-Maximum Suppression (NMS) is an indispensable post-processing step in object detection

Introduction

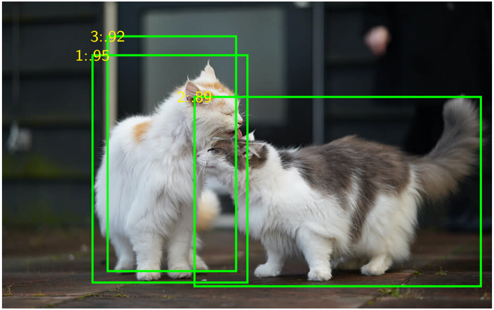
- CNN-based object detection models [2, 3] consist of two parts:
  1. model inference
  2. post-processing
- Non-Maximum Suppression (NMS) is an indispensable post-processing step in object detection
- NMS gradually becomes a bottleneck in the pipeline of object detection [4]

Introduction



- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

Introduction



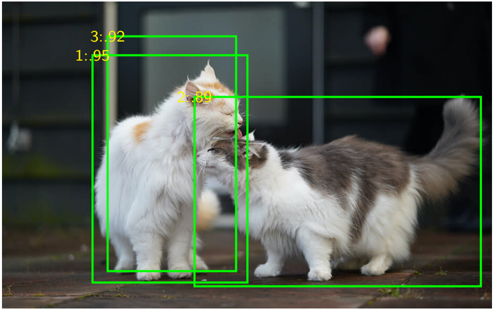
1	3	2

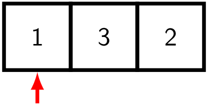
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

retained

suppressed

Introduction



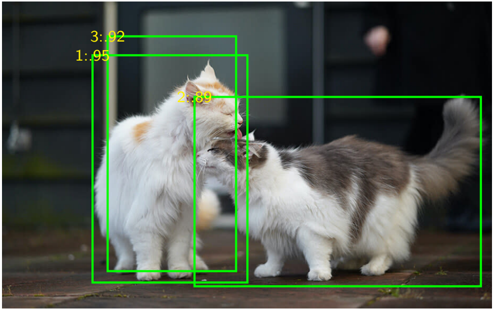


- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

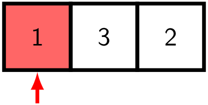
retained

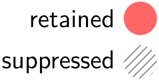
suspended

Introduction

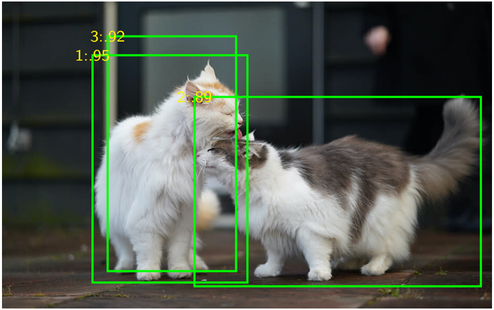


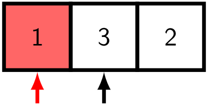
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order





Introduction



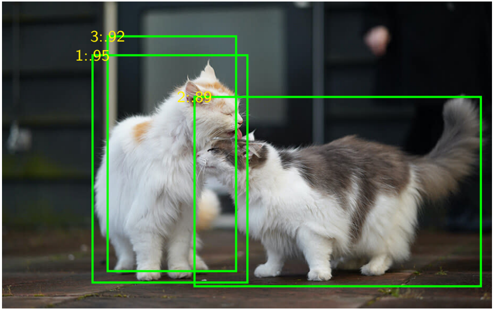


- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

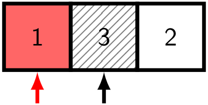
retained

suppressed

Introduction



- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

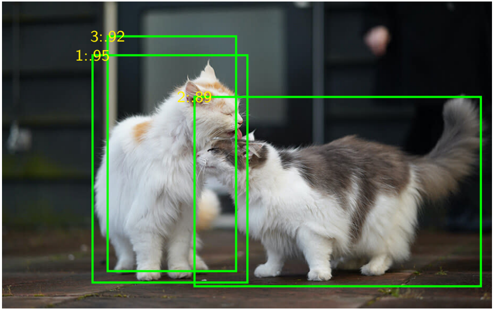


$IOU = .8 > N_t$

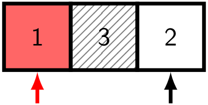
retained

suppressed

Introduction



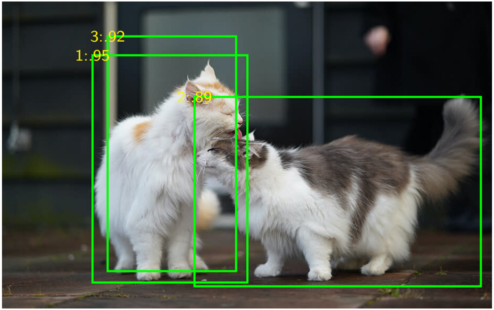
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

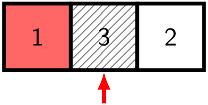


$IOU = .16 \leq N_t$

retained
suppressed

Introduction



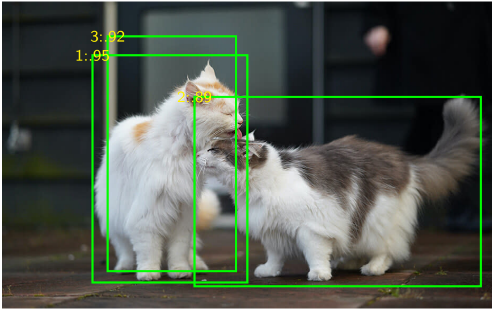


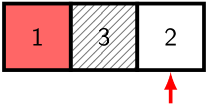
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

retained

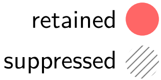
suppressed

Introduction

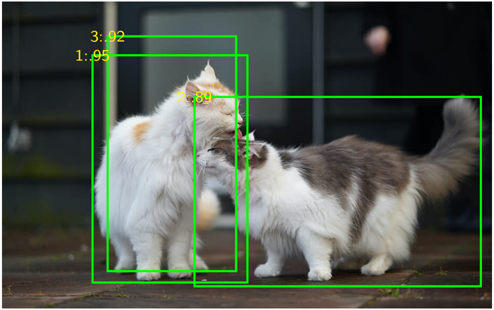


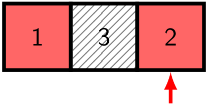


- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order

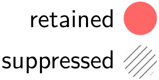


Introduction

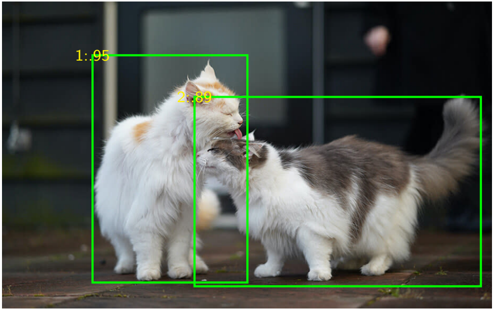




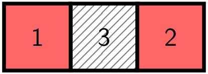
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order



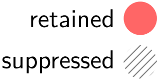
Introduction



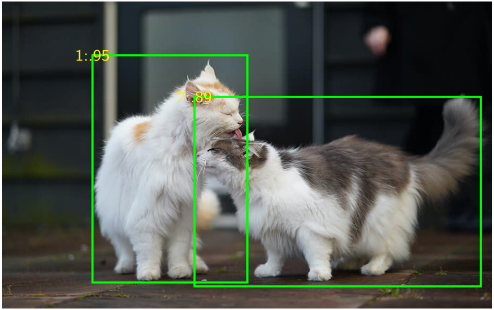
- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order



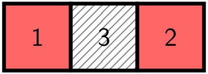
retained boxes $= \{1, 2\}$



Introduction



- set a threshold, e.g., $N_t = .7$
- sort by confidence in descending order
- $\mathcal{O}(n^2)$



retained boxes $= \{1, 2\}$

retained
suppressed

A Graph Theory Perspective

- original NMS: too many calculations of IOUs

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

Proposition

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

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

Proposition

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

- dynamic programming in topological sorting

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

Proposition

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

- dynamic programming in topological sorting

Corollary

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

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

Proposition

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

- dynamic programming in topological sorting

Corollary

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

- sorting by confidence is not necessary

A Graph Theory Perspective

- original NMS: too many calculations of IOUs
- the input of NMS can be regarded as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- v suppresses $u \Leftrightarrow (v, u) \in \mathcal{E}$

Proposition

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

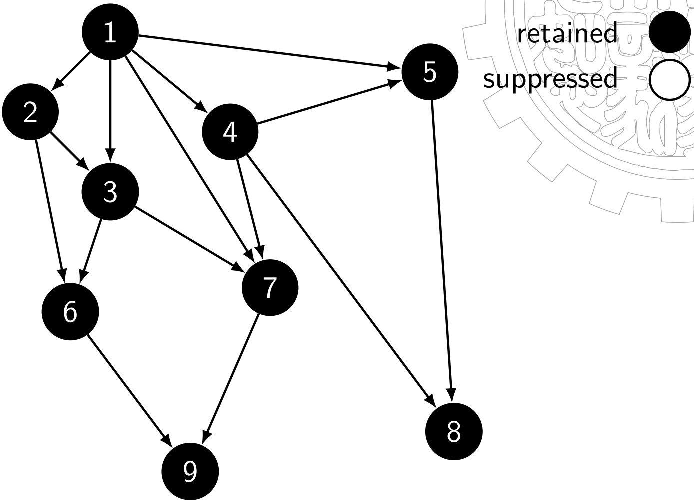
- dynamic programming in topological sorting

Corollary

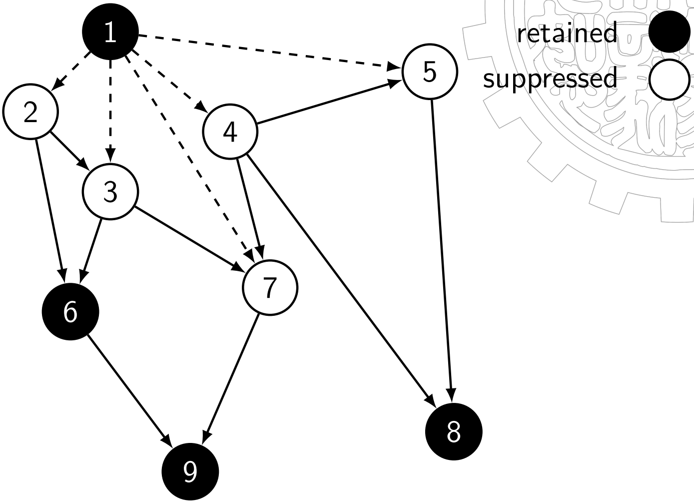
If v and u are in two different weakly connected components (WCCs) of $\mathcal{G}$, then $\delta(v)$ and $\delta(u)$ are independent.

- sorting by confidence is not necessary
- idea: to construct $\mathcal{G}$ quickly

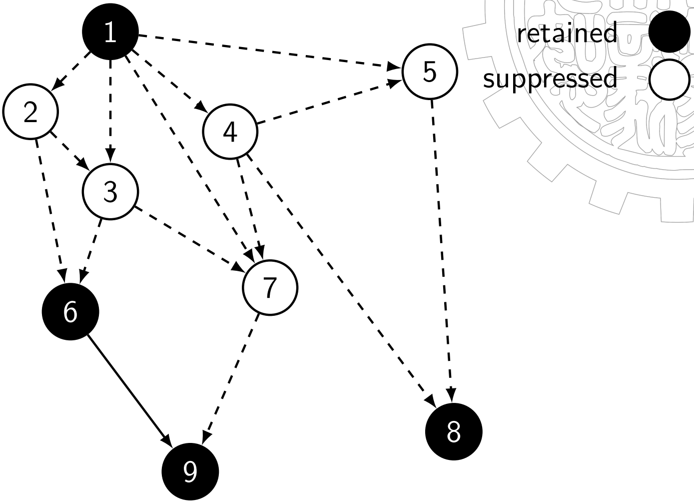
A Graph Theory Perspective



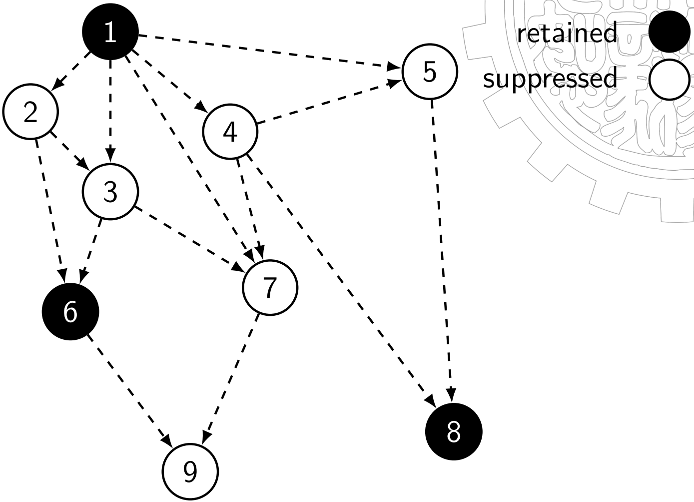
A Graph Theory Perspective



A Graph Theory Perspective



A Graph Theory Perspective



QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$
- a binary tree?

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$
- a binary tree?
- treap / Cartesian tree

QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$
- a binary tree?
- treap / Cartesian tree
- v can only be influenced by its ancestors

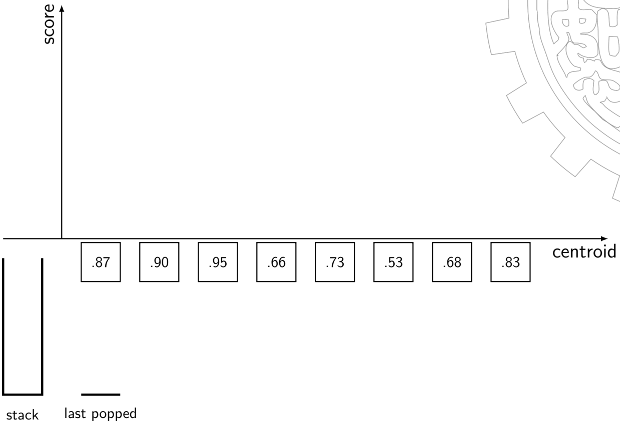
QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$
- a binary tree?
- treap / Cartesian tree
- v can only be influenced by its ancestors
- sort boxes by centroid ($\mathcal{O}(n \log n)$) and construct Cartesian tree ($\mathcal{O}(n)$)

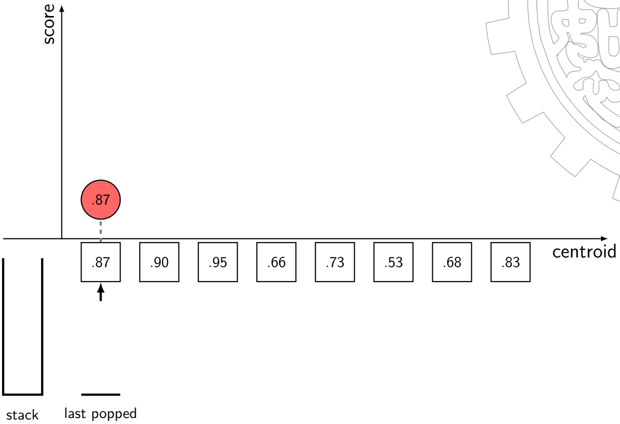
QSI-NMS

- key insight: $\mathcal{G}$ contains many small WCCs
- divide and conquer
- quicksort induced NMS
  1. select the box b^* with the highest confidence score as the pivot
  2. boxes are divided into two unrelated subproblems according to the preorder defined on $\mathbb{R}^2$
- a binary tree?
- treap / Cartesian tree
- v can only be influenced by its ancestors
- sort boxes by centroid ($\mathcal{O}(n \log n)$) and construct Cartesian tree ($\mathcal{O}(n)$)
- total complexity: $\mathcal{O}(n \log n)$

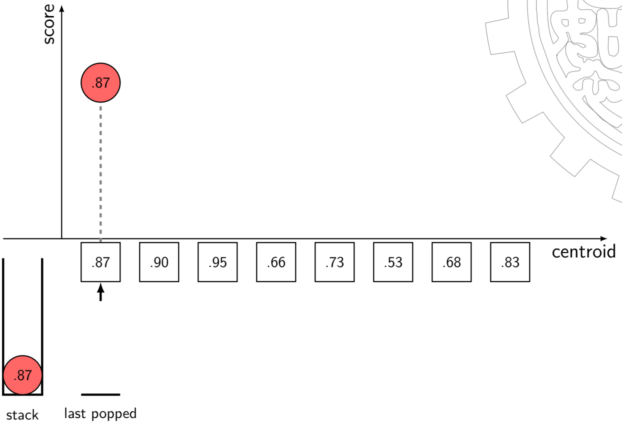
eQSI-NMS



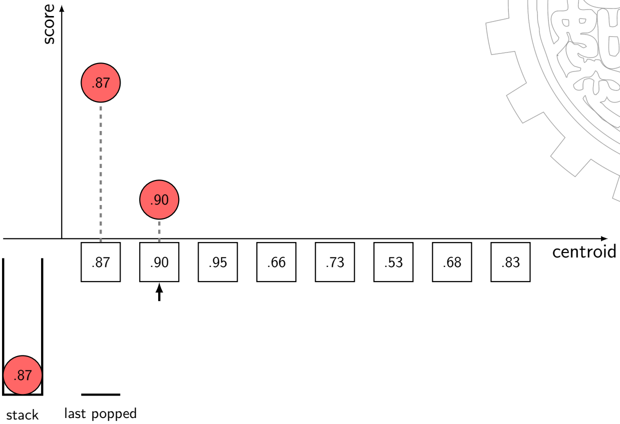
eQSI-NMS



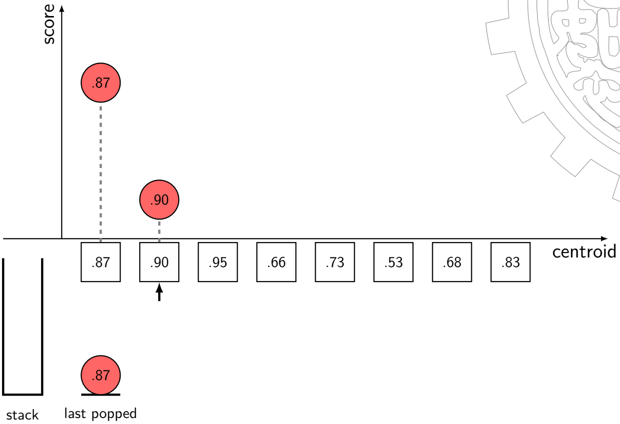
eQSI-NMS



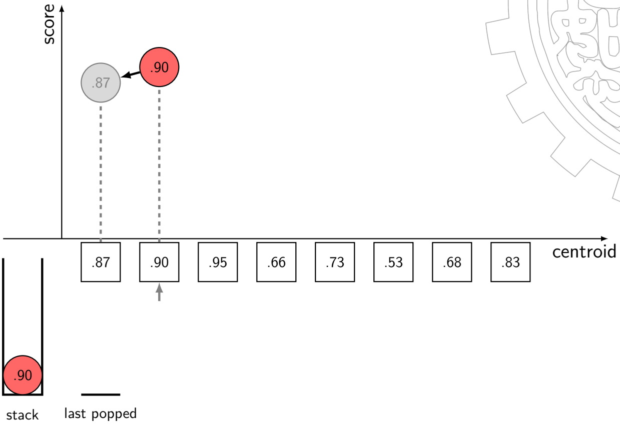
eQSI-NMS


score
.87
.90
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.87
stack
last popped

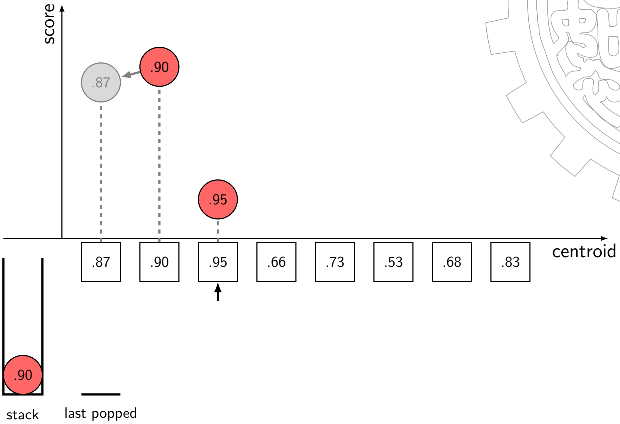
eQSI-NMS


score
.87
.90
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.87
stack
last popped

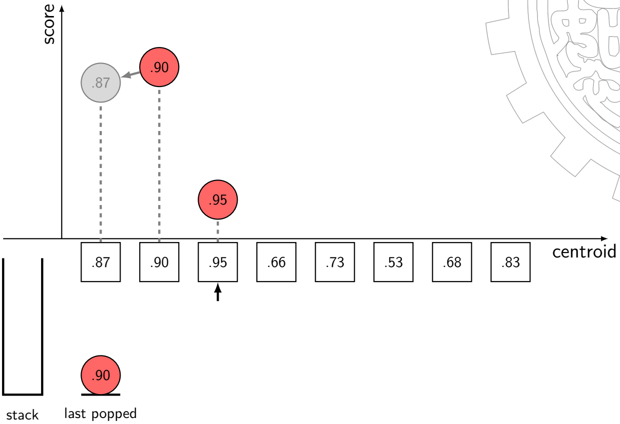
eQSI-NMS


score
.87
.90
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.90
stack
last popped

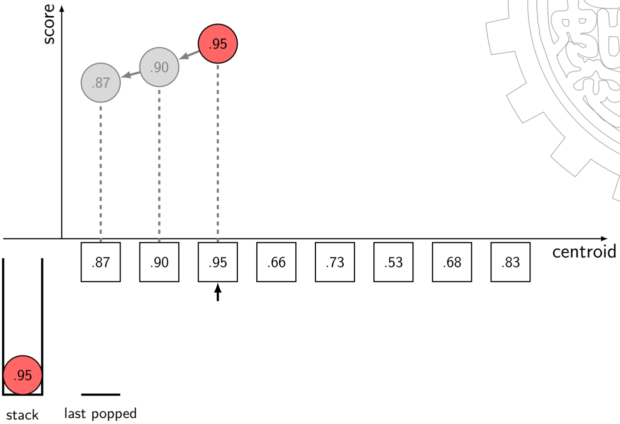
eQSI-NMS


score
.87
.90
.95
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.90
stack
last popped

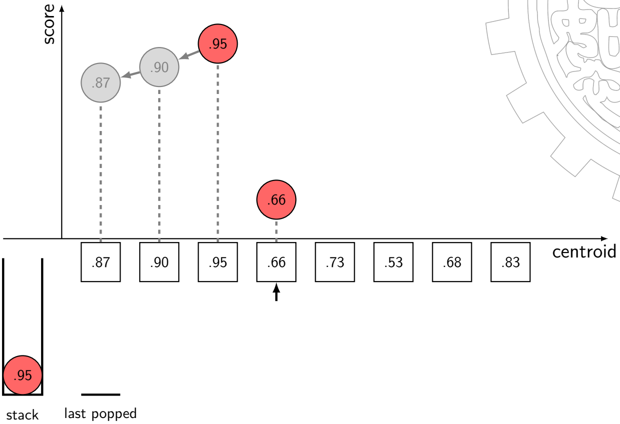
eQSI-NMS


score
.87
.90
.95
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.90
stack
last popped

eQSI-NMS

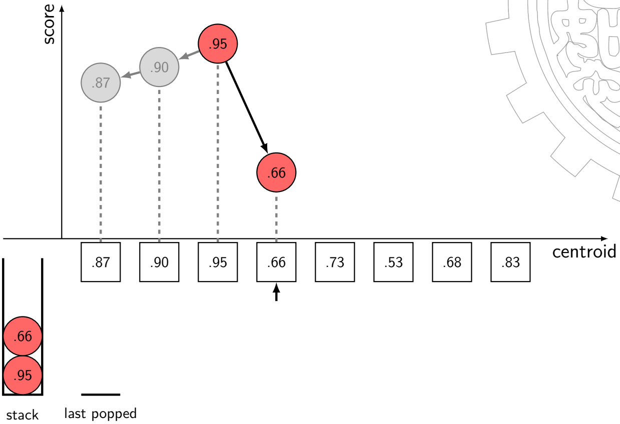

score
.95
.90
.87
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.95
stack
last popped

eQSI-NMS



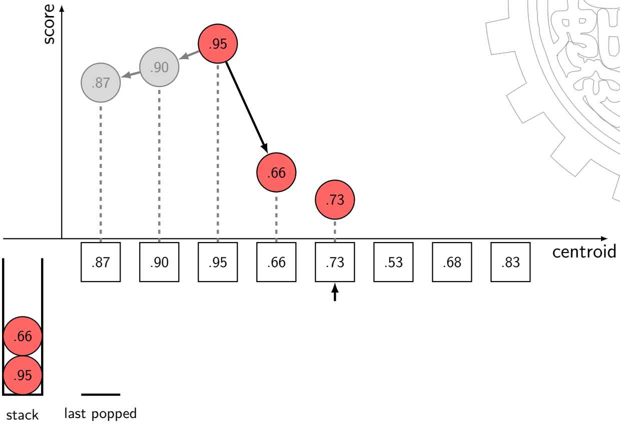
score
.95
.90
.87
.66
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.95
stack
last popped

eQSI-NMS

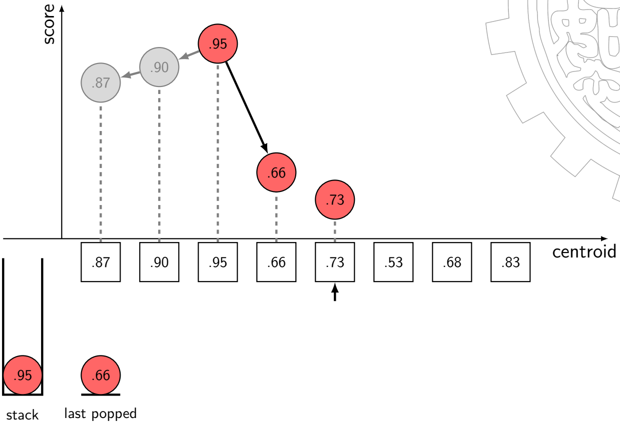


score
.87
.90
.95
.66
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.66
.95
stack
last popped

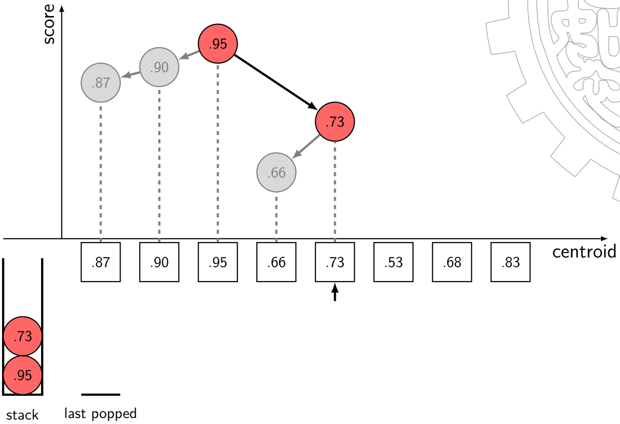
eQSI-NMS


score
.95
.90
.87
.66
.73
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.66
.95
stack
last popped

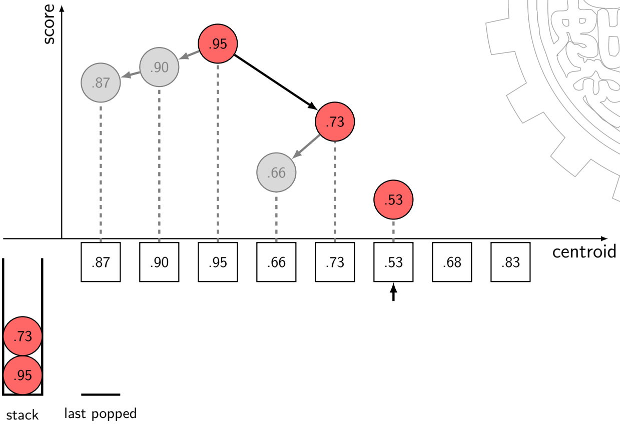
eQSI-NMS


score
.95
.90
.87
.66
.73
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.95
.66
stack
last popped

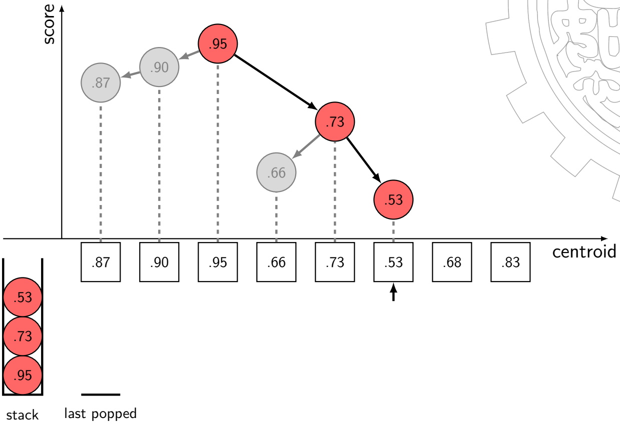
eQSI-NMS


score
.95
.90
.87
.73
.66
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.73
.95
stack
last popped

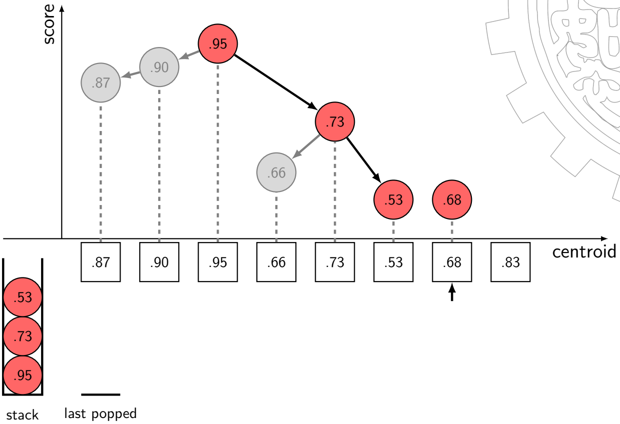
eQSI-NMS


score
.95
.90
.87
.73
.66
.53
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.73
.95
stack
last popped

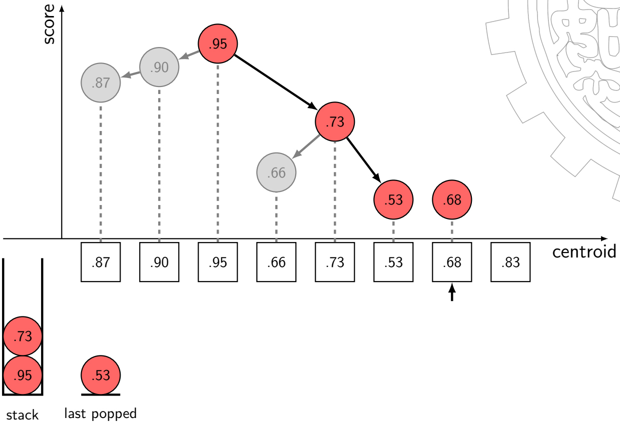
eQSI-NMS


score
.95
.90
.87
.73
.66
.53
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.53
.73
.95
stack
last popped

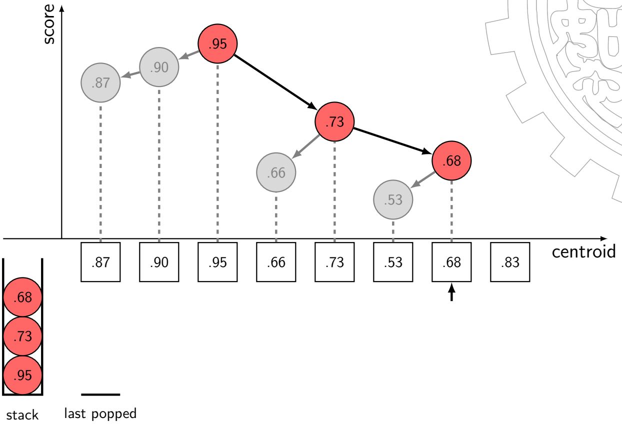
eQSI-NMS


score
.95
.90
.87
.73
.66
.53
.68
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.53
.73
.95
stack
last popped

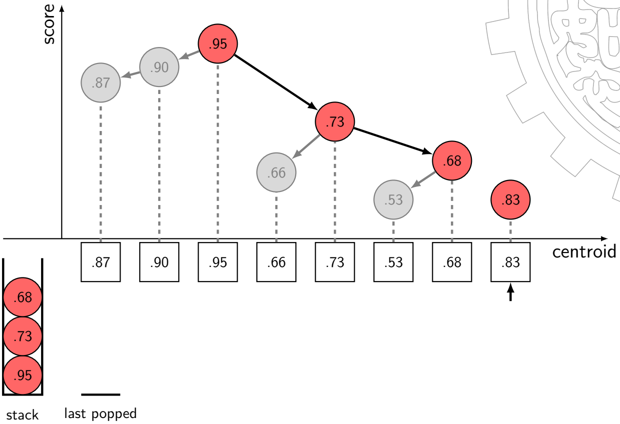
eQSI-NMS


score
.95
.90
.87
.73
.66
.53
.68
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.73
.95
.53
stack
last popped

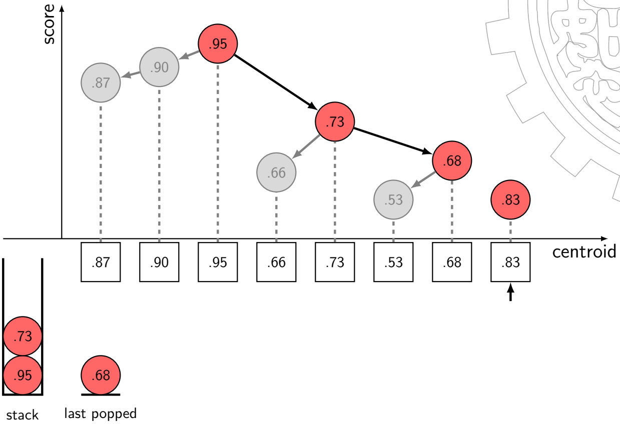
eQSI-NMS


score
.95
.90
.87
.73
.66
.68
.53
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.68
.73
.95
stack
last popped

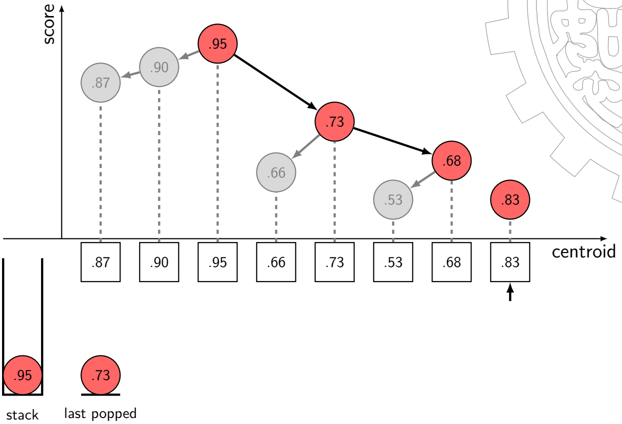
eQSI-NMS


score
.95
.90
.87
.73
.66
.68
.53
.83
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.68
.73
.95
stack
last popped

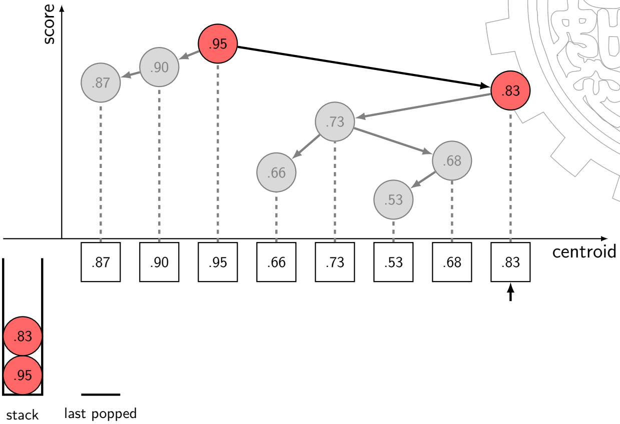
eQSI-NMS


score
.95
.90
.87
.73
.66
.68
.53
.83
.87
.90
.95
.66
.73
.53
.68
.83
centroid
.73
.95
stack
.68
last popped

eQSI-NMS

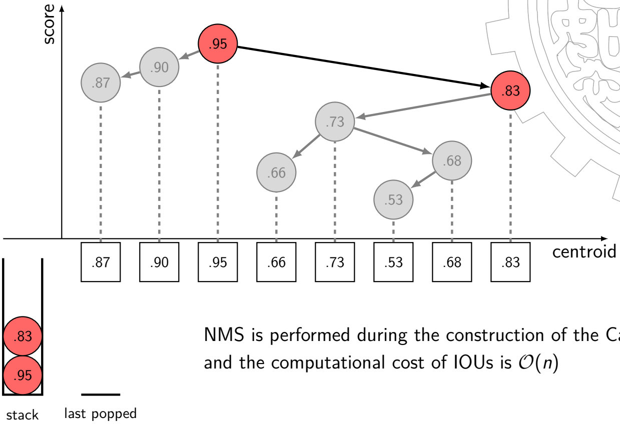

score
.95
.90
.87
.73
.66
.68
.53
.83
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.95
stack
.73
last popped

eQSI-NMS



score
.95
.90
.87
.83
.73
.66
.68
.53
centroid
.87
.90
.95
.66
.73
.53
.68
.83
.83
.95
stack
last popped

eQSI-NMS



NMS is performed during the construction of the Cartesian tree, and the computational cost of IOUs is $\mathcal{O}(n)$

BOE-NMS

- key insight: $\mathcal{G}$ is a sparse graph

BOE-NMS

- key insight: $\mathcal{G}$ is a sparse graph
- many IOU calculations are unnecessary

BOE-NMS

- key insight: $\mathcal{G}$ is a sparse graph
- many IOU calculations are unnecessary

Theorem

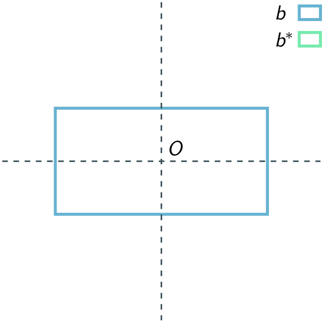
Given a bounding box $b^* \in \mathcal{B}$, $\forall b \in \mathcal{B}$, we have $IOU(b^*, b) \leq \frac{1}{2}$ if the centroid of b does not lie within b^*. Formally,

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

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.

BOE-NMS

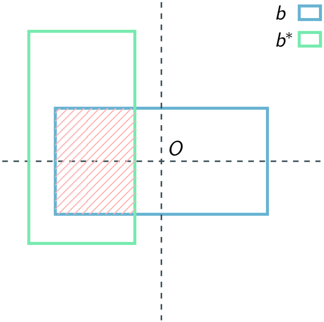
a sketch of proof.



□

BOE-NMS

a sketch of proof.

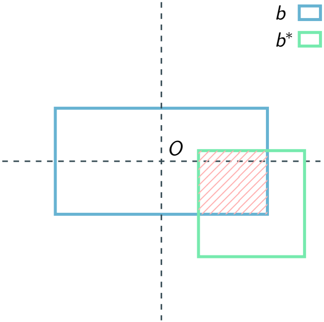


$$
\begin{aligned}
\mathrm{IOU}(b^*, b) &= \mathrm{IOU}(b, b^*) \\
&= \frac{\mathrm{Area}(\mathrm{red})}{\mathrm{Union}(b, b^*)} \\
&\le \frac{1/2\mathrm{Area}(b)}{\mathrm{Area}(b)} \\
&= \frac{1}{2}.
\end{aligned}
$$

□

BOE-NMS

a sketch of proof.

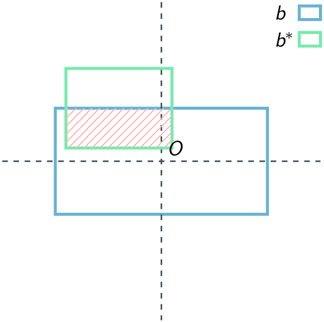


$$
\begin{aligned}
\mathrm{IOU}(b^*, b) &= \mathrm{IOU}(b, b^*) \\
&= \frac{\mathrm{Area}(\mathrm{red})}{\mathrm{Union}(b, b^*)} \\
&\le \frac{1/2\mathrm{Area}(b)}{\mathrm{Area}(b)} \\
&= \frac{1}{2}.
\end{aligned}
$$

□

BOE-NMS

a sketch of proof.

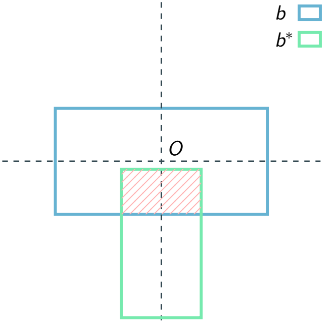


$$
\begin{aligned}
\mathrm{IOU}(b^*, b) &= \mathrm{IOU}(b, b^*) \\
&= \frac{\mathrm{Area(red)}}{\mathrm{Union}(b, b^*)} \\
&\leq \frac{1/2\mathrm{Area}(b)}{\mathrm{Area}(b)} \\
&= \frac{1}{2}.
\end{aligned}
$$

□

BOE-NMS

a sketch of proof.



$$
\begin{aligned}
\mathrm{IOU}(b^*, b) &= \mathrm{IOU}(b, b^*) \\
&= \frac{\mathrm{Area}(\mathrm{red})}{\mathrm{Union}(b, b^*)} \\
&\leq \frac{1/2\mathrm{Area}(b)}{\mathrm{Area}(b)} \\
&= \frac{1}{2}.
\end{aligned}
$$

□

Results

Table 1: NMS Methods Performance on MS COCO 2017 [1]

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

[1] 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.

[2] 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.

[3] 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.

[4] 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.

Thanks!