Scalable Hardware Acceleration of Non-Maximum Suppression

Chunyun Chen*, Tianyi Zhang†, Zehui Yu*, Adithi Raghuraman*, Shwetalaxmi Udayan*,
Jie Lin† and Mohamed M. Sabry Aly*
* *Nanyang Technological University*, Singapore, {CHUNYUN001,YUZE0004,ADITHI001,SHWETALA001,msabry}@ntu.edu.sg
† *Institute for Infocomm Research, A*STAR*, Singapore, {Zhang_Tianyi,lin-j}@i2r.a-star.edu.sg

***Abstract*—Non-maximum Suppression (NMS) in one- and two-stage object detection deep neural networks (*e.g.*, SSD and Faster-RCNN) is becoming the computation bottleneck. In this paper, we introduce a hardware acceleration for the scalable PSRR-MaxpoolNMS algorithm. Our architecture shows $75.0\times$ and $305\times$ speedups compared to the software implementation of the PSRR-MaxpoolNMS as well as the hardware implementations of GreedyNMS, respectively, while simultaneously achieving comparable Mean Average Precision (mAP) to software-based floating-point implementations. Our architecture is $13.4\times$ faster than the state-of-the-art NMS one. Our accelerator supports both one- and two-stage detectors, while supporting very high input resolutions (*i.e.*, FHD)—essential input size for better detection accuracy.**

***Index Terms*—deep learning, Non-maximum Suppression, parallel computing, object detection**

I. Introduction

Deep neural networks (DNNs, *a.k.a.*, deep learning) [1] is becoming the de-facto standard in modern artificial intelligence (AI) application workloads, thanks to their high detection accuracy [2]–[5]. For computer vision applications, such as autonomous driving, object-detection DNNs can be grouped into one-stage paradigms like SSD [3] and YOLO [4] and two-stage paradigms like Faster-RCNN [5]. Such DNNs require fast Non-Maximum Suppression (NMS) algorithms as NMS is mandatory and has become the major performance bottleneck [6]. While convolution operations have undergone massive performance improvement, thanks to both hardware scalable architectures (*e.g.*, GPUs and DNN accelerators) and algorithmic optimizations (*e.g.*, depth-wise, pruning), the commonly-adopted GreedyNMS has not witnessed significant improvement and thus has started to dominate the inference time, as illustrated in Fig. 1. It is then imperative to develop hardware accelerators for efficient NMS algorithms that are scalable.

In this paper, we present ShapoolNMS—a scalable and parallelizable hardware accelerator for PSRR-MaxpoolNMS [8]—to speed up the NMS in both one- and two-stage detectors. Unlike GreedyNMS [9], whose time complexity is $\mathcal{O}(n\log(n))$ (sorting) + $\mathcal{O}(nm)$ (nested loops) [6], PSRR-MaxpoolNMS owns a time complexity $\mathcal{O}(n)$, where n is the number of boxes before NMS and m is the number of boxes after NMS. Moreover, ShapoolNMS achieves $13.36\times$ and $59.18\times$ speedups over state-of-the-art NMS architectures, while achieving $714.7\times$ speedup over PSRR-MaxpoolNMS. Our contributions are:

- A detailed architecture of ShapoolNMS, highlighting the detailed flow of each module.
- A scalability analysis of ShapoolNMS.
- A performance analysis of ShapoolNMS versus state-of-the-art software and hardware accelerations of NMS.

This research is supported in part by the NTUITIVE Gap fund (NGF-2019-07-019) and the NRF AME programmatic fund titled Hardware-Software Co-optimisation for Deep Learning (A1892b0026).

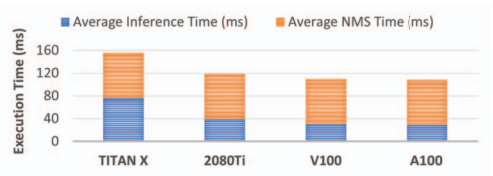


Fig. 1. Execution time of inference on different GPU platforms and GreedyNMS on Intel(R) Core(TM) i9-10900X CPU. Time was measured when the GPU was launched to run a Faster R-CNN detection network with ResNet-50 backbone [7] on PASCAL VOC benchmarking dataset. Only top 6,000 bounding boxes are selected in stage 1.

II. Related Work

Object-detection DNNs generate highly overlapped bounding boxes around the ground-truth objects. A post-processing step—Non-maximum Suppression (NMS)—is introduced to filter the overlapping detections to reduce false positives.

GreedyNMS is the commonly-used NMS [9] with high computational complexity. Many NMS algorithms [10]–[13] are proposed to reduce the complexity and increase the Mean Average Precision (mAP). Hardware-accelerated NMS methods are less explored. CPU-NMS [14] and GPU-NMS [15], [16] target on CPU and GPU platforms. GPU-NMS V2 [16] could process 1027 bounding boxes in 0.324 ms on GeForce GTX 1060 at 1.70 GHz. Inspired by GPU-NMS [15]. Shi *et al.* [17] implemented a power-efficient NMS accelerator which merges 1000 bounding boxes in 12.79 μs at 400MHz. MaxpoolNMS [6] and PSRR-MaxpoolNMS [8] reformulate NMS as MaxPool operation which is inherently parallel and friendly to the hardware. However, the accelerated NMS still occupies a significant time compared with the execution time of convolution operations. Additionally, these solutions do not support high image resolution.

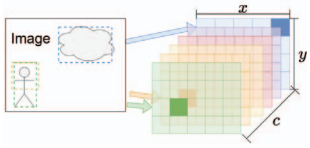


Fig. 2. RR to project the bounding boxes (dashed rectangles) with different scales, ratios and spatial location into the 3D score map (highlighted rectangles)

III. PSRR-MaxpoolNMS Algorithm Review

PSRR-MaxpoolNMS (PSRR-MNMS) [1] [8] approximates GreedyNMS in both one- and two-stage detectors via **Relationship Recovery** (RR). RR projects the boxes into a 3D confidence score map [2] and **Pyramid Shifted MNMS** (PS MNMS) eliminates more overlapped boxes.

Channels in the 3D confidence score map represent different scales s_0 and ratios r_0 of anchor boxes[3], while each cell in the score map indicates the score and the spatial information (x, y) of a bounding box (bbox). In this paper, there are 4 scales and 3 ratios with a downsampling ratio $\beta = 16$. s_0 and r_0 are:

$$s_0 \in [64^2, 128^2, 256^2, 512^2] \tag{1}$$

$$r_0 \in [1:2, 1:1, 2:1] \tag{2}$$

Hence, the confidence score map consists of 12 channels. Its width and height are $round(\frac{W}{\beta})$ and $round(\frac{H}{\beta})$ respectively, where W and H are the width and height of images.

A. Relationship Recovery

Relationship Recovery (RR) projects the bounding boxes to the 3D score map after downsampling of spatial locations, as Fig. 2 illustrates, to solve the mismatch problem [8].

1) Spatial Recovery: computes the spatial location (x, y) of the bounding box in the 3D score map based on their center position $[X_c, Y_c]$ and the downsampling ratio β:

$$x = [\frac{X_c}{\beta}],\ y = [\frac{Y_c}{\beta}] \tag{3}$$

2) Channel Recovery: projects the bbox to a channel $c(s_0, r_0)$, where s_0 and r_0 are its scale and ratio that are Euclidean-nearest to the box scale s' and ratio r'.

3) Score Assignment: each cell in the score map only keeps the box with the highest score to remove highly similar boxes.

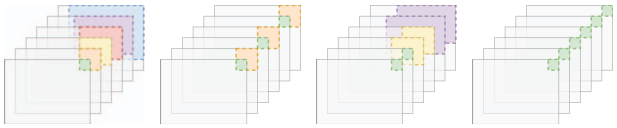
B. Pyramid Shifted MaxpoolNMS

Pyramid Shifted MaxpoolNMS (PS MNMS) consists of Pyramid MNMS and Shifted MNMS. As Fig. 3 shows, **Pyramid MNMS** consists of four steps to suppress the overlapped boxes. **Shifted MNMS** addresses the edge effect problem

[1]As the inputs of stage 1 in two-stage detectors are cells of the confidential score map, PSRR-MNMS is simplified as Single-Channel MNMS without RR and Shifted MNMS, which is called **partial PSRR-MNMS** . While the PSRR-MNMS in one-stage detectors or stage 2 of two-stage detectors is called **full PSRR-MNMS**, which includes both RR and PS MNMS.

[2]Confidence score is the probability that an anchor box contains an object.

[3]Its scale equals its height times its width, its ratio equals its height divided by its width.



(a) Single-Channel (b) Cross-Ratio (c) Cross-Scale (d) Cross-Channel

Fig. 3. The four steps of Pyramid MaxpoolNMS. The 6 channels in the score map represent 2 scales and 3 ratios. Different MaxPool kernels are represented in different colorful dashed rectangles.

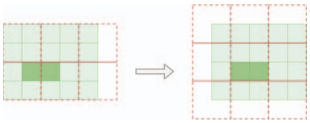


Fig. 4. Shifted MNMS addresses the edge effect problem by padding $\frac{k}{2}$ zeros for a kernel size k, followed by the same Maxpool step in Pyramid MNMS. Two adjacent cells are kept after PS MNMS and only the higher one is kept after Shifted MNMS. MaxPool kernels are represented as red dashed rectangles.

shown in Fig. 4. They reduce maximum number of overlapped boxes via a sequence of MaxPool operations with fine-tuned kernel sizes and strides:

$$h = \sqrt{r \cdot s},\ w = \frac{s}{h} \tag{4}$$

$$k_x = s_x = max(round(\frac{\alpha w}{\beta}), 1) \tag{5}$$

$$k_y = s_y = max(round(\frac{\alpha h}{\beta}), 1) \tag{6}$$

where r and s are the ratio and scale of each channel which belong to Eq. (2) and Eq. (1). h and w denote the height and width of anchor boxes in the channel. k_x, k_y and s_x, s_y are kernels and strides in x and y direction. $\alpha \in (0, 1)$ represents the overlap threshold. A larger α would remove more overlapped boxes while missing the objects. The $round$ function rounds the values to the nearest integer. In this paper, α equals 0.75 and 0.25 in full and partial PSRR-MNMS respectively as [6], [8] suggest.

IV. ShapoolNMS Architecture

Fig. 5 illustrates the architecture of ShapoolNMS, including Relationship Recovery Ways (RRWs), MaxPool Kernel Index Compute Units (MKICUs), XBar and Score Map Memories (SMMs).

A. ShapoolNMS Controller

As shown in Fig. 6, the four stages coincide with the four steps in full PSRR-MNMS, where each stage is further split into two sub-stages—Pyramid MNMS and Shifted MNMS.

Below are steps of full PSRR-MNMS: 1) N boxes are input into the RRWs parallelly whose results are written into SMM A via the address computed by MKICU B; 2) When all boxes are fed, N cells in SMM A are read simultaneously and MKICU B computes the address for SMM B; 3) When SMM A read finishes, SMM B is read and MKICU A computes the address for SMM A; 4) Repeat step 2 and 3 until all stages finish.

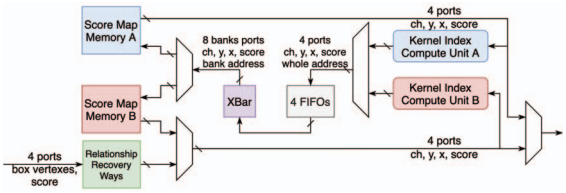


Fig. 5. System architecture which is illustrated with 4 RRWs ($N = 4$) and 8 memory banks in SMMs ($M = 8$). The depth of the FIFOs is 4.

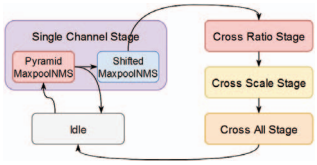


Fig. 6. The Controller FSM simplified state transition diagram

The process of partial PSRR-MNMS is simpler: 1) The spatial location of N boxes are written into SMM B via the addresses computed by MKICU B without Relationship Recovery; 2) When all boxes are input, N cells in SMM B are read out in parallel as outputs.

B. Relationship Recovery Ways (RRWs)

As Fig. 7 illustrates, this module contains N RRWs, where each RRW could process one box and compute its $channel$, x and y in the confidential score map per clock cycle, *i.e.*, Spatial Recovery and Channel Recovery in the algorithm. Unlike PSRR-MNMS, Score Assignment is processed together with Single-Channel MNMS in ShapoolNMS as it is treated as 1×1 MaxPool operation, which is fused with Single-Channel MNMS to speedup and reduce energy consumption.

C. MaxPool Kernel Index Compute Unit (MKICUs)

Fig. 8 illustrates the MaxPool operations in ShapoolNMS which are done by mapping the score map cells within the same MaxPool kernel into a memory cell so that only the one with the highest score is kept. MKICUs are used to compute the kernel index, *i.e.*, the physical memory address of the bounding boxes. There are two MKICUs in ShapoolNMS for Pyramid MaxpoolNMS and Shifted MaxpoolNMS. As Fig. 9 shows, each MKICU contains four submodules for four PS MNMS steps as introduced in Section III-B.

D. XBar

XBar is used to send $score$, $channel$, x and y to the corresponding memory bank based on the address computed by MKICU. Its ratio is $N \times M$, *i.e.*, its input ports number equals the number of ways (N in Fig. 7) and output ports number equals the number of memory banks in Score Map Memories (M). When multiple bounding boxes target the same memory bank per cycle, conflict happens. The XBar also acts as an Arbiter with round-robin scheduling to resolve the conflicts. The address from the FIFOs is specially designed to reduce the conflicts, *e.g.*, the 3 least significant bits of the address are used as bank selection for $M = 8$.

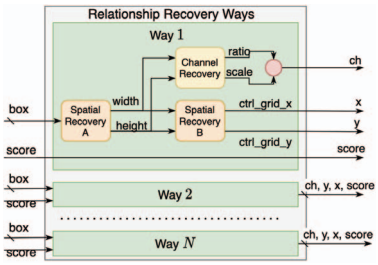


Fig. 7. Relationship Recovery Ways

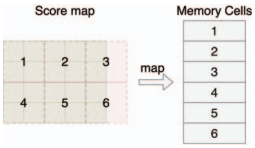


Fig. 8. The MaxPool operation in ShapoolNMS is done by mapping the score map cells with same ID into the physical memory cells, *e.g.*, the four score map cells in MaxPool kernel 1 are mapped into memory cell 1.

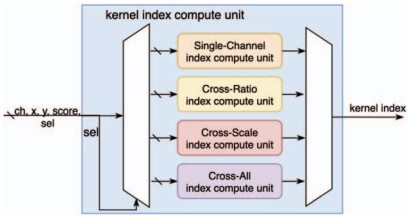


Fig. 9. MaxPool Kernel Index Compute Unit

E. Score Map Memory (SMMs)

There are two SMMs: SMM A is used to store the $channel$, x, y and $score$ of the bounding boxes after Pyramid Maxpool-NMS while SMM B stores that of Shifted MNMS. The score of a new box will be compared with the existing one in the memory cell, and the one with the highest score will be kept.

V. Evaluation

We quantify the speedup, accuracy, and scalability of ShapoolNMS with common NMS algorithms, as well as state-of-the-art designs. PSRR-MNMS and GreedyNMS software implementations are measured on CPU-based execution, while we develop a detailed cycle-accurate model of ShapoolNMS.

A. Detection Accuracy

Table II summarizes the mAP of ShapoolNMS with different scores and spatial location bit width. Intuitively, better precision yields higher mAP. Our results also indicate that the mAP of PSRR-MNMS is 0.81% less compared with GreedyNMS. Our 32-bit ShapoolNMS is 1.75% less than GreedyNMS and 0.94% less than PSRR-MNMS. This is acceptable and caused mainly by the difference in data format and precision—IEEE 754 in the software implementations and 32-bit fixed-point in ShapoolNMS.

TABLE I
EXECUTION TIME OF PS MNMS AND THE CAPACITY OF SCORE MAP MEMORY IN DIFFERENT IMAGE RESOLUTIONS

Image Resolution	PS MNMS Cycles[a]	Memory Capacity (KB) [b]
720x480 (480p)	1,455	7.03
1280x720 (720p)	3,713	19.8
1920x1080 (1080p)	8,069	44.94
2048x1080 (2K)	8,610	47.96
3840x2160 (4K)	31,536	190.22

[a] This cycles is near constant for any number of bounding boxes.
[b] The memory size of two Score Map Memory when the score is 8 bit.

TABLE II
THE MAP OF SHAPOOLNMS, PSRR-MNMS AND GREEDYNMS ON PASCAL VOC DATASET WITH FASTER-RCNN AND SSD FRAMEWORKS

Detection Pipeline	GreedyNMS	PSRR-MaxpoolNMS	ShapoolNMS		
			Score and Box bit width		
			32	*16*	*8*
Faster-RCNN ResNet-50	78.4	77.59	76.65	75.55	74.88
Faster-RCNN ResNet-152	78.7	78.4	77.32	76.45	75.59
SSD VGG-16	77.3	76.1	75.28	74.83	74.31

TABLE III
EXECUTION TIME AND SPEEDUP OF CLUSTERING 1000 AND 8000 BOUNDING BOXES IN DIFFERENT METHODS COMPARED WITH GREEDYNMS

Method	NMS Execution Time (μs)		Speedup	
	n=1,000	*n=8,000*	*n=1,000*	*n=8,000*
GreedyNMS[a]	35,000	512,000	1	1
PSRR-MaxpoolNMS [8][a]	18,000	89,000	1.944	5.753
GPU-NMS V2 [16][b]	324	—	108.025	—
Shi *et al*. [17][c]	12.79	102.32	2,736.513	5,003.909
ShapoolNMS[d]	5.69	11.99	6,154.495	42,713.830

[a] The data is reported in [8].
[b] Executed on GeForce GTX 1060. The execution time of $n = 8,000$ is not reported.
[c] The operation frequency is 400MHz. The speedup for $n = 8,000$ is projected from the reported execution time of $n = 1,000$ and $n = 10,000$ in linear trend.
[d] The operation frequency is 400MHz. The shown values is the execution time of a 4-way ShapoolNMS to process full PSRR-MNMS. The partial one is $4.2\mu s$ faster.

B. Execution Time

Fig. 10 illustrates the execution cycles of 4-way ShapoolNMS (*i.e.*, $N = 4$ in Fig. 7) and the 64-parallel compute components GreedyNMS hardware utilizing Bubble and Bitonic sorting algorithms (sorting is a key component in GreedyNMS). ShapoolNMS increasingly outperforms GreedyNMS hardware, GPU-NMS V2 [16] and Shi *et al*. [17] with an increased number of boxes. This is due to the time complexity of GPU-NMS and Shi *et al*. is at least $\mathcal{O}(nm)$ [17], while that of ShapoolNMS is $\mathcal{O}(n)$, as summarized in Table III. At the system-level, executing Faster-RCNN with Resnet-50 backbone and ShapoolNMS achieves $3.76\times$ speedup compared to executing Faster-RCNN with GreedyNMS[4].

Table IV lists the execution time of PS MNMS and the capacity requirements in different image resolutions.

C. Hardware Utilization

We implemented ShapoolNMS in Verilog and synthesized it. A 2-way ShapoolNMS that supports up to FHD image resolution utilizes 14,188 LUTs, 2,924 FFs, 7,504 LUT RAMs and 11 BUFGs when mapped on Gensys 2 FPGA. It occupies a 0.31mm^2 footprint when synthesized with 28nm foundry PDK.

[4]The execution time of the system contains convolution operations and NMS. The convolution is accelerated by A100 GPU, see Fig.1

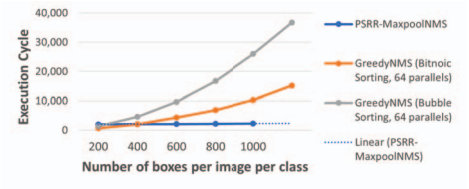


Fig. 10. The execution cycles for a 4-way ShapoolNMS and a 64-parallel compute components GreedyNMS hardware with Bubble sorting algorithm and Bitonic sorting algorithm to process various number of bounding boxes.

TABLE IV
THE EXECUTION TIME OF PS MNMS AND THE CAPACITY OF SMM IN DIFFERENT IMAGE RESOLUTIONS

Image Resolution	PS MNMS Cycles[a]	Memory Capacity (KB) [b]
720x480 (480p)	1,455	7.03
1280x720 (720p)	3,713	19.8
1920x1080 (1080p)	8,069	44.94
2048x1080 (2K)	8,610	47.96
3840x2160 (4K)	31,536	190.22

[a] This cycles is near constant for any number of bounding boxes.
[b] The memory size of two Score Map Memory when the score is 8 bit.

VI. CONCLUSION

We have presented a highly scalable and parallel NMS hardware architecture — ShapoolNMS — that clusters tens of thousands of bounding boxes in very low latency. ShapoolNMS achieves $8.54\times$ speedup than state-of-the-art and $3626\times$ speedup versus GreedyNMS software implementations. ShapoolNMS is suitable for both one- and two-stage detection frameworks with any number of bounding boxes and supports very high input resolutions for improved accuracy.

REFERENCES

[1] Y. LeCun *et al.*, "Deep learning," *nature*, vol. 521, pp. 436–444, 2015.
[2] J. Deng *et al.*, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019, pp. 4690–4699.
[3] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *ECCV*, 2016, pp. 21–37.
[4] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.
[5] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv Neural Inf Process Syst*, vol. 28, pp. 91–99, 2015.
[6] L. Cai *et al.*, "Maxpoolnms: getting rid of nms bottlenecks in two-stage object detectors," in *CVPR*, 2019, pp. 9356–9364.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[8] T. Zhang *et al.*, "Psrr-maxpoolnms: Pyramid shifted maxpoolnms with relationship recovery," in *CVPR*, 2021, pp. 15 840–15 848.
[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, 2005, pp. 886–893.
[10] J. Hosang, R. Benenson, and B. Schiele, "A convnet for non-maximum suppression," in *GCPR*, 2016, pp. 192–204.
[11] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms–improving object detection with one line of code," in *ICCV*, 2017, pp. 5561–5569.
[12] S. Liu, D. Huang, and Y. Wang, "Adaptive nms: Refining pedestrian detection in a crowd," in *CVPR*, 2019, pp. 6459–6468.
[13] N. O. Salscheider, "Featurenms: Non-maximum suppression by learning feature embeddings," in *ICPR*, 2021, pp. 7848–7854.
[14] R. Rothe *et al.*, "Non-maximum suppression for object detection by passing messages between windows," in *ACCV*, 2014, pp. 290–306.
[15] D. Oro *et al.*, "Work-efficient parallel non-maximum suppression for embedded gpu architectures," in *ICASSP*, 2016, pp. 1026–1030.
[16] ——, "Work-Efficient Parallel Non-Maximum Suppression Kernels," *The Computer Journal*, 08 2020.
[17] M. Shi *et al.*, "A fast and power-efficient hardware architecture for non-maximum suppression," *TCAS-II*, vol. 66, no. 11, pp. 1870–1874, 2019.