Mobile-X: Dedicated FPGA Implementation of the MobileNet Accelerator Optimizing Depthwise Separable Convolution

Hyeonseok Hong, *Student Member, IEEE*, Dahun Choi, *Student Member, IEEE*, Namjoon Kim, *Student Member, IEEE*, and Hyun Kim, *Senior Member, IEEE*

***Abstract*—MobileNet proposed depthwise separable convolution (DSC) as a replacement for standard convolution (SC), achieving significant reductions in parameters and computational complexity compared with traditional convolutional neural network (CNN) models. Recently, there has been a growing trend of deploying MobileNet on various edge devices by implementing accelerators. However, the distinctive computational patterns of depthwise convolution (DWC) and pointwise convolution (PWC) in MobileNet pose challenges for FPGA and ASIC accelerator implementations. In this brief, we propose DSC-dedicated processing engine (PE) designs specialized for DWC and PWC operations and an SC reordering module for only the first convolution layer. In addition, we introduce the pipeline DSC processing called pipelining separable convolution (PSC) and tiled-convolution (TC) techniques that consider the computational load of PWC. Our proposed 8-bit quantization in the accelerator causes only a negligible accuracy drop (*i.e.*, 0.68%) compared with full precision, yet it enables hardware-friendly operations with only a single fixed-point multiplication. On the ZCU-102 platform, the proposed accelerator achieves 190.9 FPS and 108.3 GOPS using minimal hardware resources. Consequently, we achieve 18.20 GOPS/W, showing a 3.7× power efficiency compared to the A-100 GPU.**

***Index Terms*—Convolutional neural network (CNN), FPGA, hardware accelerator, MobileNet.**

I. Introduction

RECENTLY, convolutional neural networks (CNNs) have made remarkable advancements in solving various computer vision tasks, such as image classification, object detection, and segmentation [1], [2], [3]. However, deploying CNNs in resource-constrained mobile/edge devices for real-time operation is challenging because of their mega-scale parameter counts and giga-scale calculations [4], [5]. To address this issue, extensive research has been conducted on various lightweight techniques and compact models along with the development of hardware (HW) accelerators [6]. MobileNet [7] is a representative compact model, which achieves an 86% reduction in parameters and an 88% decrease in operations compared to traditional CNNs by replacing standard convolution (SC) with depthwise separable convolution (DSC). DSC consists of depthwise convolution (DWC) for learning spatial information and pointwise convolution (PWC) for learning correlations between channels, with each convolution having different computational patterns. Recently, various efforts have been made to implement dedicated accelerators and integrate them into different edge devices to utilize MobileNet efficiently with real-time and low-power processing [8].

Most MobileNet inference accelerators maximize the flexibility between different operations by processing three types of convolutions (*i.e.*, SC, DWC, and PWC) using a general convolution engine. Liu et al. [9] implemented an operation engine for handling 3D inputs and weights, achieving a performance of 206 GOPS for VGG-16. However, they achieved a throughput of only 13 GOPS for MobileNet. Liao et al. [10] implemented MobileNet using only 109 DSPs and 110.5 BRAMs, but achieved only 0.35 FPS, falling short of real-time processing. Yan et al. [11] achieved a high throughput of 181.8 GOPS by processing SC and DSC using a single convolution engine, but the relatively large resource consumption of 2160 DSPs makes it unsuitable for low-power operation. Su et al. [12] proposed redundancy-reduced MobileNet by applying compression to the original MobileNet, achieving a throughput of 91.2 GOPS, but showing a relatively large performance drop (64.6% in Top-1 accuracy). As a result, the existing MobileNet accelerator research has not been optimized in terms of the trade-off between throughput and HW resources. Therefore, it is important to design a dedicated accelerator with pipelined processing units (PUs) optimized for specific layers by considering this trade-off [13], [14].

To address the limitations of previous studies, we introduce Mobile-X, a high-throughput and power-efficient MobileNet accelerator that employs a variety of techniques spanning algorithms and architectures tailored to the characteristics of MobileNet. Consequently, our accelerator, implemented on the Xilinx ZCU-102 platform, utilizes only 69 DSPs and achieves a performance of 18.20 GOPS/W, demonstrating the most outstanding power efficiency compared to prior research

Manuscript received 14 March 2024; revised 15 July 2024; accepted 6 August 2024. Date of publication 7 August 2024; date of current version 30 October 2024. This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119, and in part by the Industrial Fundamental Technology Development Program (Development of Low Power AI Architecture for AIoT) funded by the Ministry of Trade, Industry and Energy (MOTIE) of Korea under Grant 20019367. This brief was recommended by Associate Editor V. Gaudet. *(Corresponding author: Hyun Kim.)*

The authors are with the Department of Electrical and Information Engineering and the Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea (e-mail: hs_hong@seoultech.ac.kr; dahun926@seoultech.ac.kr; rlarla2626@seoultech.ac.kr; hyunkim@seoultech.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2024.3440884.

Digital Object Identifier 10.1109/TCSII.2024.3440884

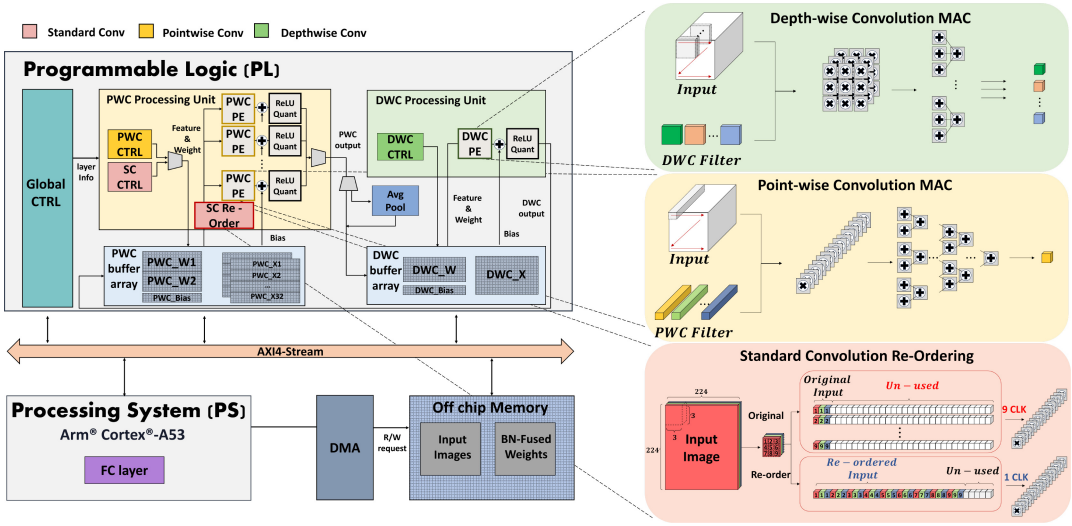


Fig. 1. Design Overview of the Proposed Accelerator.

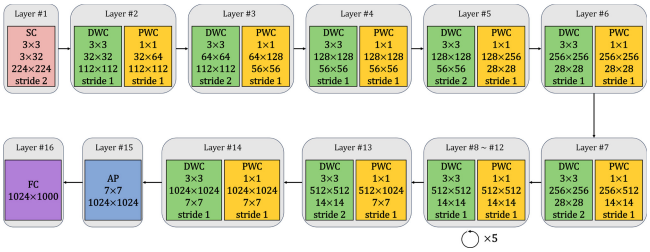


Fig. 2. Illustration of MobileNet-V1 Architecture Including Layer Information.

[9], [15], [16]. Particularly noteworthy is its superior power efficiency, outperforming both the A-100 GPU and previous accelerators by up to 3.7× and 8.8×, respectively.

The contributions of this brief are summarized as follows.

- Our proposed IP includes dedicated PU designs for efficient DWC and PWC operations, along with a reordering module for single SC operation. We introduce a PSC operation technique utilizing these components, achieving a 1.82× reduction in DWC latency through DWC hiding.
- To reduce the dominant latency caused by the proportion of PWC operations within MobileNet, our proposed IP suggests tiled convolution (TC), which partitions input feature maps of both PWC and SC into multiple input tiles for simultaneous processing. This approach achieves a throughput enhancement of 24.5∼32×.
- Our proposed IP optimizes the trade-off between throughput, HW resources, and accuracy by incorporating a quantizer that enables 8-bit quantization with only a single fixed-point multiplication while minimizing accuracy degradation (0.685% of Top-1 accuracy drop compared to full precision).

II. Proposed Architecture

A. Design Overview

The design overview of the proposed accelerator and the structure of the MobileNet-V1 model are shown in Fig. 1 and Fig. 2, respectively. As depicted in Fig. 1, contrary to most single-engine accelerators [9], [10], [12], our design incorporates specialized PUs for PWC and DWC, labeled as PWC PU and DWC PU in Fig. 1. Within each PU, the results from each processing engine (PE) are passed through ReLU and quantizer modules. In addition, each PU is equipped with on-chip buffer arrays to handle the input, filter, and bias independently, thereby enabling concurrent convolution operations. In particular, the double buffering of the PWC filter buffer utilizes a ping-pong technique to eliminate the latency caused by weight loading. In addition, as shown in Fig. 2, the feature maps of MobileNet-V1, excluding the input image channels, are composed of channel sizes that are multiples of 32. Hence, within the on-chip buffer, one data word is structured to store 32 data items bundled channel-wise, enabling 32-channel unrolling in MAC operations to enhance throughput.

The global controller (Global CTRL) disseminates layer information to each PU, such as the input feature map dimensions and number of filters. It regulates the data flow such that the layer structure of MobileNet-V1 can be processed sequentially using the proposed accelerator. In addition, it controls the off-chip memory access through direct memory access (DMA). After all the convolutions are completed, the average pooling (Avg Pool) operation, which is the last operation in programmable logic (PL), is designed to receive an intermediate output from the PWC PU for computation. The FPGA processing system (PS) performs fully connected (FC) layer operations, softmax, and other post-processing tasks in parallel with the PL to produce the final output.

B. Dedicated Processing Unit

1) DWC Processing Unit: DWC operates by sliding a 3×3 kernel independently across each channel of the input feature map, performing multiply and accumulate (MAC) operations. In our design, each DWC PE is optimized for this operation by arranging MAC units in parallel for each channel and employing a weight stationary technique to push in the 3×3 window input, thus realizing an optimized DWC operation. The proposed PWC PU can perform parallel DWC operations using several parallel MAC units. In this design, 32 parallel

MAC units are used for parallel processing. DWC CTRL in Fig. 1 receives the input resolution and stride information from Global CTRL, controls the data flow, reads the data from the DWC buffer array, and delivers it to the PEs.

2) PWC Processing Unit: PWC involves computing the correlations between all channels of the input feature map, necessitating a different approach from DWC. Hence, within each PWC PE, we sequentially arrange multipliers to compute the channel-wise input feature multiplied by the corresponding filter, followed by accumulation through an adder tree to perform PWC operations. PWC PEs utilize an input stationary technique for holding the input feature map in registers and changing the filters during operations. Similar to DWC PEs, we employ 32 MAC units within PWC PEs. PWC CTRL in Fig. 1 receives the input resolution and channel size information from the global CTRL, controls the PWC data flow, and reads data from the buffer array.

3) SC Reorder Module: As shown in Fig. 2, because MobileNet contains a single SC operation only at the beginning, designing a separate PU for SC is highly inefficient in terms of HW resources. Therefore, as illustrated in Fig. 1, we introduce the SC CTRL and SC reorder module to implement SC operations through the PWC PEs. The traditional SC involves MAC operations between a 3×3×3 (kernel width, kernel height, and channel) filter and the input window. However, constructing data words in the conventional channel direction for use as PWC PE operands results in inefficient HW utilization, with only three out of 32 data words and MAC units being utilized, leading to increased latency owing to the data indexing of nine clock cycles per SC operation. To address this issue, we first construct data words for the input image in a row-wise manner, rather than channel-wise, and store them in buffers. The SC reorder module then loads row-wise input data, flattens the original 3×3×3 input window into a 1×27 operand, and forwards it to the PWC PEs. This enables the utilization of all 27 MAC units per pixel, achieving SC operation in only one clock cycle and providing a 9× improvement in HW utilization and throughput compared to conventional SC operations.

C. Pipelining Separable Convolution

DSC in MobileNet comprises consecutive PWC and DWC operations, with DWC using the output of PWC as its input. Moreover, because DWC operates independently on each channel, it can perform its operation using intermediate outputs from PWC, even if not all PWC outputs are available. In this brief, we propose a pipelining separable convolution (PSC) technique leveraging this characteristic of DSC. Fig. 3 compares a naïve DSC with the proposed PSC. In a naïve DSC, the DWC PU waits for PWC operations to be completed for each output channel (*i.e.*, OC) before proceeding, resulting in the sequential processing of all convolution layers. In contrast, in PSC, when the filtering operation corresponding to pipelining (*i.e.*, PPL) is completed within the PWC PU, the PPL signal is forwarded to the DWC PU, allowing DWC operations to commence using intermediate PWC outputs of size PPL channels. Consequently, pipelining enables simultaneous PWC and DWC operations. By repeating PSC for the *OC/PPL* filter bundles, we achieve the same output as the conventional DSC while hiding most of the DWC operations, leading to a reduction in latency. In this design, we set the size of PPL

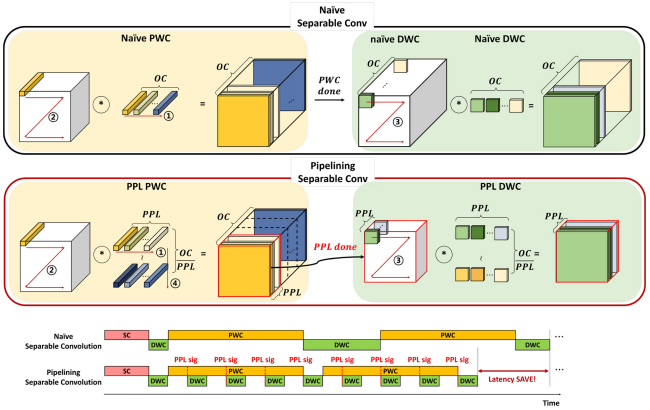
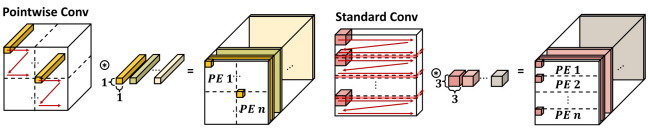


Fig. 3. Comparison of the Naive DSC and the Proposed PSC.

TABLE I
PROPORTION OF MACS AND PARAMETERS WITHIN MOBILENET

	MAC (%)		Parameters (%)	
PWC	539,492,352	(94.86%)	3,139,584	(75.59%)
DWC	17,385,984	(3.06%)	44,640	(1.06%)
SC	10,838,016	(1.19%)	864	(0.02%)
FC	1,024,000	(0.18%)	1,024,000	(24.33%)



(a) PWC Tiled Convolution (b) SC Tiled Convolution

Fig. 4. Illustration of the Proposed Tiled Convolution.

to 32, which is the most common divisor of the feature map channels in MobileNet-V1, thus enabling the application of PPL operations to all layers. This approach hides 7.6 million operations, thereby reducing DWC latency by 1.82×.

D. Tiled Convolution

As shown in Table I, the proportion of MAC and parameters with respect to operations within MobileNet is overwhelmingly dominated by PWC. This indicates that during MobileNet inference, PWC dominates in terms of latency. To achieve high throughput in PWC operations, we propose the TC technique, which divides the input feature map into multiple tiles stored in distributed buffers and assigns them to multiple PEs. Fig. 4(a) presents an overview of PWC TC. PWC operates using a 1×1 kernel, indicating that there is no spatial overlap between the feature maps. Thus, each tile can independently perform full PWC operations, enabling parallel processing equal to the number of divided tiles. Additionally, because the resolution of the input feature maps decreases towards the latter layers of CNNs, to divide them into more tiles, we split them in both the row and column directions. In this design, we divide the input feature map of PWC into an 8×4 grid for the TC, resulting in a 24.5∼32× throughput enhancement.

Furthermore, because even a single SC is performed within the PWC PU in our accelerator, we also apply TC to the SC. However, for SC, it is essential to consider tile division at the row level, as it loads input data from off-chip memory at the row level. Contrary to PWC, overlap

TABLE II
PERFORMANCE COMPARISON WITH PREVIOUS WORKS

	GPU	[9]	[10]	[12]	[17]	[15]	[18]	[16]	[19]	**Proposed**
Model	MobileNetV1	MobileNetV1	MobileNetV1	RR-MobileNet	MobileNetV1-SSD	MobileNetV2	MobileNetV2	MobileNetV2	MobileNetV2-SSD	MobileNetV1
Platform	A-100	Xilinx XC7Z045	Xilinx XC7Z045	Xilinx XCZU9EG	Xilinx XCZU2EG	Xilinx XC7Z020	Xilinx XC7Z020	Xilinx XC7Z020	Xilinx XC7Z045	Xilinx XCZU9EG
Freq.(MHz)	-	150	100	150	430	150	150	100	100	150
BRAMs	-	537	110.5	864.5	145	136	119.5	126.5	311	506.5
DSPs	-	787	109	1452	212	206	176	214	728	**69**
LUTs / FFs (k)	- / -	154 / 128	9.2 / 16.9	139 / 55	31.2 / 46.8	- / -	36.3 / -	39.1 / -	148.4 / 437.2	214.9 / 166.7
Image Size	224×224	224×224	224×224	224×224	-	224×224	224×224	224×224	224×224	224×224
Data format	Floating-32	Fixed-8	Fixed-16	Fixed-8(W),4(A)	Fixed-8	Fixed-16	Fixed-8	Mixed	Mixed	Fixed-8
Speed (FPS)	358.4	5.5	0.35	127.4	31.0	17.0	70.94	49.2	64.8	**190.9**
Throughput (GOPS)	203.6	13.0	-	91.2	-	10.86	-	29.3	-	**108.3**
Power (W)	73	6.24	2.15	-	-	3.11	-	3.0	9.9	5.95
Power Efficiency (GOPS/W)	4.91	2.08	-	-	-	3.49	-	9.77	-	**18.20**
HW Efficiency (GOPS/DSP)	-	0.017	-	0.063	-	0.053	-	0.137	-	1.570

occurs between input feature maps owing to the 3×3 kernel operation. Therefore, in this design, as shown in Fig. 4(b), we divide the SC at the row level. This tile composition allows for the efficient storage of input data fetched from DMA at the row level into distributed buffers and minimizes the overlap of input data, thereby achieving efficient parallel SC operations. In our accelerator, we divide the input feature map into 28 tiles for the TC to achieve a 28× throughput enhancement.

E. Fused Quantization

In this design, all weights and activations used in MobileNet inference are quantized to a fixed-point 8-bit, minimizing the usage of DSPs during MAC operations. Moreover, by quantizing the feature maps, which originally occupied 25.6 Mb of floating-point 32-bit memory, into 6.4 Mb of fixed-point 8-bit memory, we not only reduce on-chip memory usage but also implement MobileNet inference without off-chip memory access for feature maps, leading to advantages in power consumption and latency. However, the conventional quantization [20] involves two multiplications for dequantization and a division operation along with rounding for quantization, utilizing the floating-point scaling factors obtained from the activations and weights for the nth convolution output (x_n, w_n). The multiplication of floating-point scaling factors requires relatively large resource consumption compared to fixed-point multiplication and registers to store intermediate results, leading to disadvantages in HW resource utilization and latency.

To optimize HW resources and the latency caused by the quantization process, this design employs fused quantization. Fused quantization approximates the three floating-point scaling factors into a single fixed-point fused scaling factor ($FS_{x_n} \approx S_{x_n} * S_{w_n} / S_{x_{n+1}}$), replacing conventional quantization with only one fixed-point multiplication as follows:

$$FQ(x_{n+1}) = \left[x_n * FS_{x_n}\right]. \tag{1}$$

In this context, it is essential to identify fixed-point values that closely resemble floating-point values to minimize the approximation error. We employ a greedy search algorithm to explore the optimal solutions for a and b for a fixed point scaling factor ($a * 2^{-b}$). We limit the search range of a to 64 and b to 10. Through this exploration, we obtain fixed-point scaling factors with a maximum approximation error of 1e-4, resulting in negligible accuracy degradation. Ultimately, we implement (1) as an integer multiplication for the a value and a shift operation for the b value.

III. EXPERIMENTAL RESULTS

A. Experimental Environment

We trained MobileNet-V1 using the ILSVRC 2012 dataset [21] on an RTX 2080 Ti GPU. All the hyperparameters were implemented using PyTorch, following the guidelines in [7]. For accelerator implementation, we utilized the Xilinx Vivado 19.1 and SDK 2019.1 tools targeting Xilinx XCZU9EG FPGA. Power results present total on-chip power, which is the sum of static and dynamic power in the Vivado synthesis report.

B. Performance Comparison With Various Accelerator Designs

Table II presents a comparison of the proposed accelerator with the Nvidia A-100 GPU and existing MobileNet accelerators in terms of performance and resource usage. The GPU performance was directly measured for latency and power, and information not provided in previous research is denoted by a dash (-). The proposed accelerator achieved a remarkable throughput of 190.9 FPS and 108.3 GOPS, outperforming previous accelerators [9], [10], [12], [15], [16], [17], [18], [19]. Furthermore, it exhibited superior HW utilization with only 69 DSPs, which is the least number among accelerator studies. Although the Nvidia A-100 GPU demonstrated a high throughput of 203.6 GOPS, its power efficiency was 4.91 due to a power consumption of 73 W. In contrast, the power efficiency of the proposed accelerator is 18.20, which is 3.7× and 1.9× superior to the power efficiency of GPU and the most power-efficient existing study [16], respectively.

C. Ablation Studies

Table III compares the proposed fused quantization (FQ) with the naïve quant using floating-point scaling factors [22]. The naïve quant involves a significant latency consumption of 26 cycles and the utilization of 1,130 LUTs and 630 FFs owing to the conversion between fixed-point and floating-point representations of the convolution output, along with floating-point multiplication and division. In contrast, the FQ simplifies quantization using only one integer multiplication and shift operation, resulting in only two cycles of latency (13× saving) and the utilization of 75 LUTs and 55 FFs (15× and 11× saving, respectively). Consequently, the FQ achieves MobileNet inference with a negligible accuracy drop of 0.685 and 0.297 compared to the full precision and naïve quant, respectively.

Table IV presents the performance change according to each proposed technique. The existing naïve accelerator shows a throughput of only 4.8 GOPS. However, applying TC improves

TABLE III
PERFORMANCE COMPARISON ACCORDING TO QUANTIZATION METHODS

	Acc.(%)	DSP	LUTs	FFs	CARRY8	Latency(μs)
naïve quant	71.762	20	1130	632	143	210
FQ(ours)	71.465	**1**	**75**	**55**	**4**	**2**

TABLE IV
PERFORMANCE CHANGE ACCORDING TO EACH METHOD

	Latency (μs)	Throughput (GOPS)	BRAMs	DSPs	LUTs (k)	FFs (k)
Naive	118060.3	4.8	**404.5**	102	**116.2**	**68.0**
+TC	5947.3	95.4	506.5	102	249.7	185.7
+TC, +PSC	5271.8	107.6	506.5	102	250.8	185.7
+TC, +PSC, +FQ (ours)	**5238.34**	**108.3**	506.5	**69**	214.9	166.7

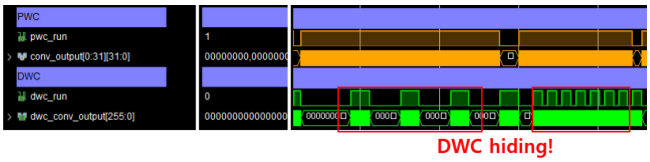


Fig. 5. Waveform of RTL simulation for Pipelining Separable Convolution.

performance by approximately 19.9 ×, with only the addition of 133.5k LUTs and 117.7k FFs. Furthermore, by applying PSC, a performance improvement of 12.2 GOPS is achieved with only a modification in the controller, consuming merely 1.1k LUTs. The waveform of PSC in Fig. 5 confirms the efficient execution of PSC through the proposed dedicated PUs. Through PSC, DWC latency hiding is implemented, reducing the DWC-induced latency by 1.8 × and increasing the throughput of our accelerator by 13%. Finally, by applying the proposed FQ instead of the naïve quantization, a slight performance improvement of 0.7 GOPS is achieved, along with a reduction in resource consumption, including 33 DSPs, 35.9k LUTs, and 19.2k FFs. These results demonstrate the effectiveness of each proposed design technique in enhancing the accelerator performance.

IV. CONCLUSION

In this brief, we proposed PSC and TC optimized for DSC in MobileNet, enhancing throughput by 1.8× and 24.5∼32×, respectively, and achieved the best performance among existing DSC-based network accelerators. Furthermore, we achieved a 3.7× higher power efficiency than GPU, demonstrating its superiority as a mobile and edge inference platform. By implementing a sparse network with low precision, we can achieve better power efficiency in the accelerator with only an acceptable accuracy drop. Based on this brief, we enable CNN inference on mobile/edge devices and expect further expansion into MobileViT [23] accelerator research.

REFERENCES

[1] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[2] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 502–511.
[3] S. I. Lee and H. Kim, "GaussianMask: Uncertainty-aware instance segmentation based on Gaussian modeling," in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*, 2022, pp. 3851–3857.
[4] E. Kristiani, C.-T. Yang, and C.-Y. Huang, "iSEC: An optimized deep learning model for image classification on edge computing," *IEEE Access*, vol. 8, pp. 27267–27276, 2020.
[5] N. J. Kim and H. Kim, "FP-AGL: Filter pruning with adaptive gradient learning for accelerating deep convolutional neural networks," *IEEE Trans. Multimedia*, vol. 25, pp. 5279–5290, 2023.
[6] D. T. Nguyen, H. Kim, and H.-J. Lee, "Layer-specific optimization for mixed data flow with mixed precision in FPGA design for CNN-based object detectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 6, pp. 2450–2464, Jun. 2021.
[7] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[8] S. Ryu, Y. Oh, and J.-J. Kim, "MobileWare: A high-performance MobileNet accelerator with channel stationary dataflow," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2021, pp. 1–9.
[9] W. Liu, Y. Li, Y. Yang, J. Zhu, and L. Liu, "Design an efficient DNN inference framework with PS-PL synergies in FPGA for edge computing," in *Proc. China Autom. Congr. (CAC)*, 2022, pp. 4186–4190.
[10] J. Liao, L. Cai, Y. Xu, and M. He, "Design of accelerator for MobileNet convolutional neural network based on FPGA," in *Proc. IEEE 4th Adv. Inf. Technol., Electron. Autom. Control Conf.*, 2019, pp. 1392–1396.
[11] S. Yan et al., "An FPGA-based MobileNet accelerator considering network structure characteristics," in *Proc. 31st Int. Conf. Field-Program. Logic Appl. (FPL)*, 2021, pp. 17–23.
[12] J. Su et al., "Redundancy-reduced MobileNet acceleration on reconfigurable logic for ImageNet classification," in *Proc. 14th Int. Symp. Appl. Reconf. Comput., Archit., Tools, Appl.*, 2018, pp. 16–28.
[13] S. Ki, J. Park, and H. Kim, "Dedicated FPGA implementation of the Gaussian TinyYOLOv3 accelerator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 10, pp. 3882–3886, Oct. 2023.
[14] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1861–1873, Aug. 2019.
[15] X. Sang, T. Ruan, C. Li, H. Li, R. Yang, and Z. Liu, "A real-time and high-performance MobileNet accelerator based on adaptive dataflow scheduling for image classification," *J. Real-Time Image Process.*, vol. 21, no. 1, p. 4, 2024.
[16] M. Sun et al., "Film-QNN: Efficient FPGA acceleration of deep neural networks with intra-layer, mixed-precision quantization," in *Proc. ACM/SIGDA Int. Symp. FPGA*, 2022, pp. 134–145.
[17] D. Wu et al., "A high-performance CNN processor based on FPGA for MobileNets," in *Proc. 29th Int. Conf. Field Program. Logic Appl. (FPL)*, 2019, pp. 136–143.
[18] Y. Lin, Y. Zhang, and X. Yang, "A low memory requirement MobileNets accelerator based on FPGA for auxiliary medical tasks," *Bioengineering*, vol. 10, no. 1, p. 28, 2022.
[19] H. Fan et al., "A real-time object detection accelerator with compressed SSDLite on FPGA," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, 2018, pp. 14–21.
[20] S. Kim and H. Kim, "Zero-centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors," *IEEE Access*, vol. 9, pp. 20828–20839, 2021.
[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1–9.
[22] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020, *arXiv:2004.09602*.
[23] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer," 2021, *arXiv:2110.02178*.