A 0.95 mJ/frame DNN Training Processor for Robust Object Detection with Real-World Environmental Adaptation

Donghyeon Han, Dongseok Im, Gwangtae Park, Youngwoo Kim, Seokchan Song, Juhyoung Lee and Hoi-Jun Yoo
School of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea
E-mail: hdh4797@kaist.ac.kr

***Abstract*—A DNN training processor with a maximum of 332 TOPS/W is proposed for efficient and robust object detection. The proposed processor is able to support both quantization and pruning-based personalization to make a user-optimized lightweight network. In addition to personalization, it supports real-time adaptation to compensate for accuracy degradation caused by environmental changes or unpredictable situations. It maintains conventional input slice skipping architecture and stochastic rounding-based computing for the efficient acceleration of the DNN training. It further improves efficiency by removing pseudo-RNGs during the stochastic rounding and adding blocks to pruning-aware training. Moreover, it suggests an LT-flag-based reconfigurable accumulation network and enables multi-learning-task-allocation for low-latency DNN training with the backward unlocking solution. Fabricated in 28-nm technology, the proposed processor demonstrates 46.6 FPS object detection with 0.95 mJ/frame energy consumption which is the state-of-the-art performance compared with the previous processors.**

Keywords—Deep Neural Network, Back-propagation, Object Detection, Quantization, Pruning, Backward Locking

I. Introduction

Object detection is one of the essential applications for the edge devices such as drones or robots. For example, object detection is the basic sensing method for drone navigation or human-robot interaction. Recently, deep neural network (DNN) becomes the mainstream of the object detection algorithm and its performance continues to be improved by increasing the size of the network. However, smarter DNNs require more parameters, making them difficult to be used on mobile/edge devices that have limited hardware resources. Edge devices inevitably have to use mobile-oriented DNNs which have fewer parameters and computational amounts. One of the trials is the network architecture modification by using simple operations such as 1×1 CONV but it is still not enough to be realized with the limited power budget.

Another solution is personalization-based network optimization which restricts the functionality of the network only for the user environment. Although the pre-trained DNN targets broad applications, actual usage at the edge device is narrow and biased. It gives an opportunity to reduce the network capacity while maintaining its accuracy on the local dataset. Many DNN training processors [1-8] already adopted personalization-based optimization through the quantization or weight pruning so that they could reduce the computational cost of the DNN dramatically on the edge.

Personalization-based lightweight DNN training has shown great success but it suffers from significant accuracy degradation when it is applied to the new environment. In other words, the lightweight DNN obtained through user customization loses generality due to its low network capacity. In particular, the mobile-oriented DNNs do not work properly in unexpected situations such as camera malfunction as described in Fig. 1. Accuracy compensation after an unpredictable accident is important to prevent critical system damage. Real-time online DNN tuning [9] is a promising solution to compensate accuracy of the lightweight network while maintaining its hardware benefits.

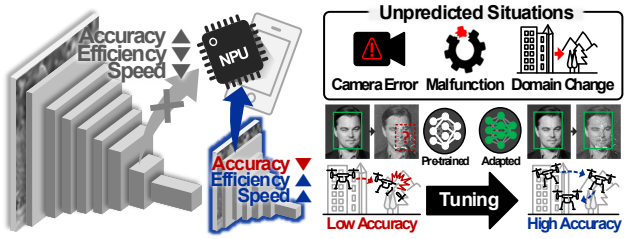


Fig. 1. Problems of lightweight DNN and necessity of online DNN tuning for robust object detection.

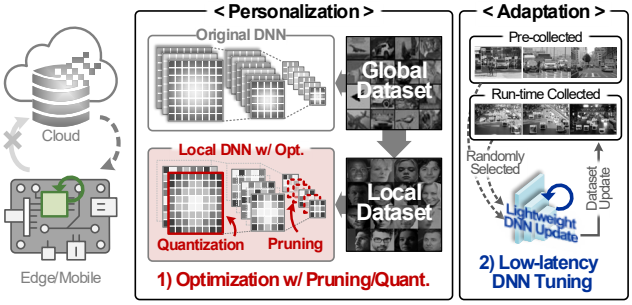


Fig. 2. Two types of DNN training supported by the proposed processor.

Conventional DNN training processors can support training functionality to realize online DNN tuning but they cannot achieve real-time training. The majority of DNN training processors [2-4] achieved high throughput by utilizing data sparsity but their sparsity exploitation was effective only when DNN adopts the ReLU nonlinear function. As more and more DNNs adopt non-ReLU functions, the word-level sparsity exploitation becomes useless in popular object detection networks such as YOLO. Moreover, they cannot realize low-latency DNN training because of the backward locking problem [10]. Only one processor [1] tried to solve this problem by adopting a backward unlocking solution (BU) but it was optimized only for fully-connected (FC) layers. It shows low reconfigurability so suffers from a core utilization drop in other layer configurations such as the CONV layer.

As shown in Fig. 2, the proposed processor can support both high-speed personalization and low-latency adaptation with three key features. 1) The intrinsic TRNG (iTRNG) generates truly random bit-streams in order to support

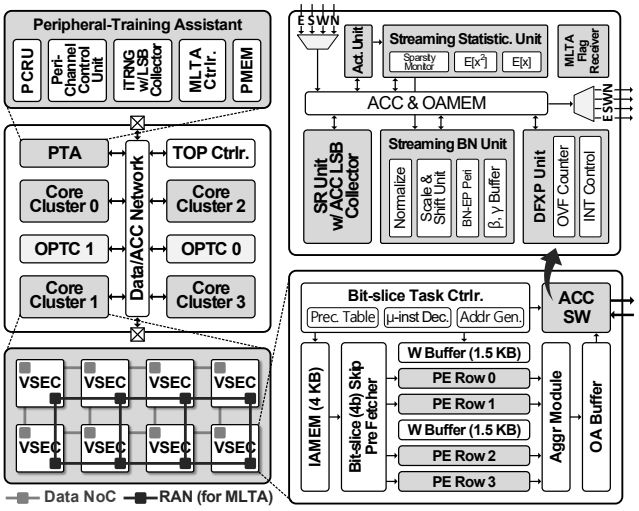


Fig. 3. Overall architecture of the proposed processor

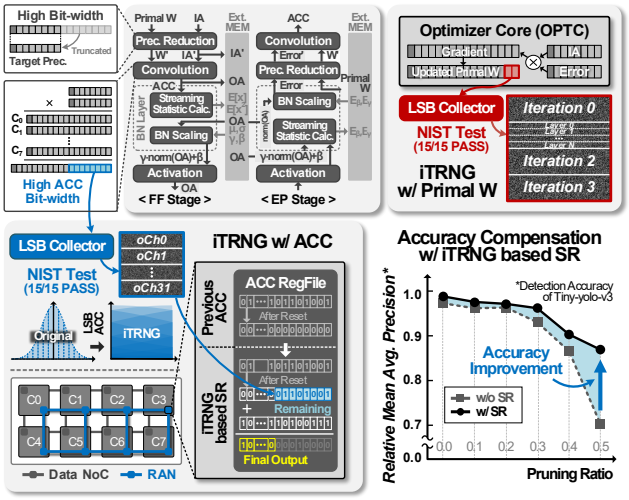


Fig. 4. Details of two intrinsic-TRNGs (iTRNG). (1) Primal weight (PW) based iTRNG and (2) accumulation (ACC) based iTRNG.

stochastic rounding (SR, [11]) for required precision reduction. 2) Versatile sparsity exploitation core (VSEC) with pruning-aware channel reordering unit (PCRU) enables both input-slice skipping (ISS) and useless channel removal to increase throughput. 3) LT-flag-based reconfigurable accumulation network (RAN) control enables multi-learning-task-allocation (MLTA) and parallel processing of three different training stages to support BU-based training.

II. Proposed Processor

Fig. 3 describes the overall architecture of the proposed processor. The basic architecture of the proposed processor follows the previous DNN training processor, HNPU [6] for slice-level sparsity exploitation. It includes a total of 32 VSECs and they are connected with both data NoC and RAN. A single VSEC contains the bit-slice (4b) skip pre-fetcher to omit zero-slice located in the input activation (IA). Since it computes 4b × 4b at a time, the bit-slice task controller divides computations into multiple μ-instructions to support the high-bit-precision computing. It affects the address generation of IAMEM, weight buffer, and OA buffer. Accumulation switch (ACC SW) placed in every VSEC executes post-processing to computation results. It includes streaming statistics and batch-normalization (BN) units. They minimize the throughput degradation caused by external memory access during the BN layer computation. It also includes dynamic fixed-point (DFXP) and iTRNG based SR units for low-bit-precision training. Peripheral training assistant (PTA) consists of a PCRU and an MLTA controller.

Thanks to the ISS, both HNPU and the proposed processor can exploit slice-level sparsity but the proposed processor can also support pruning-aware training [12] thanks to the PCRU. Instead, the proposed processor does not include the adaptive precision scaling unit of the HNPU. That's because we verified that the runtime precision search rather degrades the accuracy during the online DNN-tuning based object detection. At last, the RAN is modified to receive LT-flag and it becomes programmable to support MLTA.

III. Key Features of Proposed Processor

A. Intrinsic True Random Number Generator

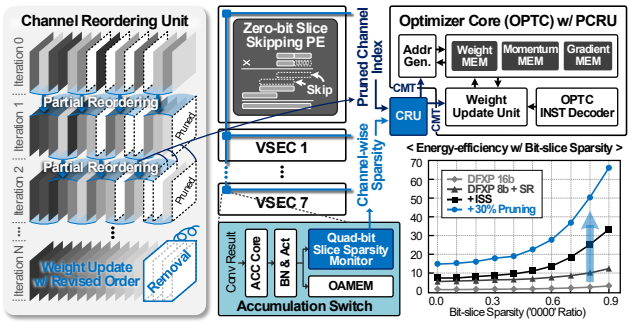
Random number generation has an important role during the DNN training. For example, when the new layers or new categories are needed, corresponding weights should be initialized with the random values. Even in online DNN tuning cases, the training data is randomly selected for stochastic gradient descent. As depicted in Fig. 4, the proposed processor is designed with extremely simple random number generators but they show truly random patterns.

Th DFXP based DNN training, it needs primal weight (PW, [6]) which has higher bit-precision compared with the normal weight used in the feed-forward (FF) and error propagation (EP) stages. In other words, it adopts higher bit-precision only for the weight-gradient-update (WG) stage. Although the MSB bits of the PW are updated infrequently and show the normal distribution, LSB bits of PW are updated every training iteration and bit-streams look like truly random patterns. The LSB collector placed in the PTA extracts and cumulates LSB bits of the PW. When 16-bit primal weight is used and the LSB collector extracts LSB 4-bits, it passes all fifteen tasks of the NIST SP-800.22 test [13] with a threshold of P-value > 0.01 (significance level).

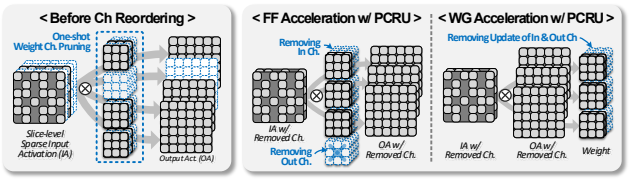
In addition to iTRNG in the PTA, another iTRNG is placed in the ACC SW and it realizes SR by remaining the LSB bits of the prior ACC results instead of adding random noise. Compared with the conventional SR method designed by the HNPU, it removes all pseudo-RNG (PRNG) circuits from the PEs so that it reduces dynamic power caused by multiple PRNGs. Furthermore, it realizes higher virtual representation resolution compared with the previous linear-feedback-shift-register-based SR. The iTRNG based SR finally achieves 16-bit-level accuracy by using only 8-bit-precision and removes all PRNGs required for SR operation.

B. Versatile Sparsity Exploitation Core

As described in Fig. 5, the VSEC with the PCRU enables ISS and it also removes useless channels caused by the narrow distribution or structured pruning. Thanks to the slice-level (4b) sparsity exploitation, the performance improvement is no matter restricted by ReLU. However, the ISS can suffer from the core utilization drop because of the unbalanced workload. The PCRU solves this problem by utilizing sparsity sorting and spreading. Thanks to the workload balancing with PCRU, it shows a maximum of 21.0% higher throughput.



(a) Bit-slice sparsity monitor and pruning-aware channel reordering unit.



(b) Pruning-aware DNN inference and training after channel reordering.

Fig. 5. Versatile sparsity exploitation core with channel reordering unit for pruning-aware DNN inference and training.

The VSEC monitors the ratio of the bit-slice sparsity through the ACC SW and it generates a channel mapping table (CMT). Generally, the number of channels in the single CONV layer is more than 64, the CMT generation for the entire channels induces high routing congestion. Instead, the PCRU defines 8-input and 8-output channels as a single sorting group and shuffles allocated channels every iteration. The PCRU can minimize the routing congestion through the simple sorting unit and it can align entire channels according to the sparsity ratio at the end of each training iteration. Compared with the channel reordering unit proposed by the previous processor [6], the PCRU additionally receives pruned channel index and considers it as 100% sparsity. The optimizer cores (OPTCs) receive CMT and update new weights with the changed order or removed channel. The updated weights are immediately applied from the next iteration. Useless channel removal through the PCRU finally improves the performance of the entire three training stages, FF, EP, and WG.

C. Multi-learning Task Allocation with LT-flag based RAN

Conventional DNN training processor supports multi-DNN allocation through the RAN [3] but it suffers from the backward locking problem. One processor [1] adopts BU but it can be applied only for fully-connected layers because of the low programmability caused by heterogeneous core architecture. As shown in Fig. 6, the proposed processor enables not only multi-DNN allocation but also MLTA through the LT-flag-based programmable RAN. The LT-flag includes the information of input/output size, bit-precision, and target training stage. The LT-flag receiver is included in the ACC SW to support dynamic core allocation. It pre-defines data movement among the PMEMs, which are distributed caches of the chip. The LT-flag-based RAN not only controls the accumulation path but also controls data movement among each core to support BUs with various network configurations.

To realize low-latency online DNN tuning, the proposed processor adopts one of the typical BU, direct feedback alignment (DFA, [1, 10]). Unlike the previous processor [1] the proposed processor adopts DFA-based training to the CONV layers. As shown in Fig. 7, the processor constructs a

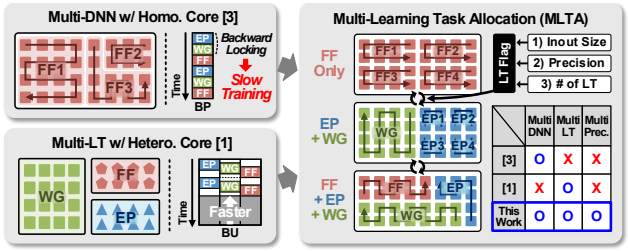


Fig. 6. LT-flag based RAN control for multi-learning task allocation.

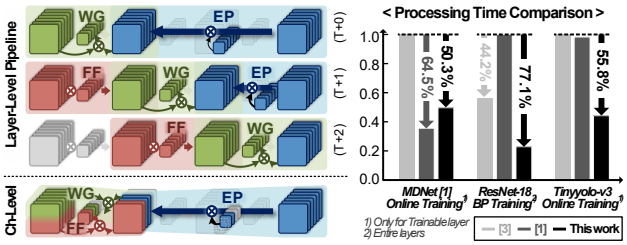
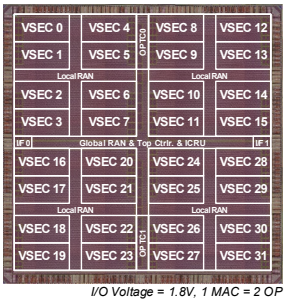


Fig. 7. DFA based channel-level pipelined DNN training.



Specifications				
Technology	Samsung 28nm 1P8M CMOS			
Die Area	3.6mm × 3.6mm (12.96mm^2)			
Supporting BU	BP, DFA [6], ... (Programmable)			
Op. Condition	0.63V (@ 10MHz) ~ 1.0V (@ 250MHz)			
Data Type	DFXP + SR (4/8/12/16)-bit × (4/8/12/16)-bit			
Area Efficiency	59-to-9334 GOPS/mm^2			
	Sparsity (IS, WS) [%]	(0,0)	(50,50)	(90,90)
Power [mW]	250MHz, 1.0V	1032	850	616
	10MHz, 0.63V	24.1	19.8	14.6
Energy Efficiency @ 10MHz [TOPS/W]	16b×16b	2.04	6.01	98.1
	8b×8b	6.81	19.9	220
	4b×4b	20.4	49.8	332

Fig. 8. Chip photograph and performance summary.

channel-level pipeline among the three training stages by utilizing DFA-based direct error propagation. As a result, it shows a 50.3-to-55.8% processing time reduction compared with the back-propagation-based DNN training.

IV. Measurement Results

A. Implementation Results of Proposed Processor

As shown in Fig. 8, a 12.96 mm^2 chip is fabricated in 28-nm CMOS technology and it operates at 10-to-250 MHz core frequency with 0.63-to-1.0 V supply voltage. The proposed processor achieves a maximum of 332 TOPS/W energy efficiency by utilizing both quantization and weight-pruning. Thanks to its lower hardware logic complexity, it also achieves higher efficiency compared to the HNPU [6] regardless of the sparsity condition. As summarized in Table I, the proposed processor achieves the best throughput and energy efficiency during the online DNN tuning-based object detection scenario. Unlike the conventional DNN training processors, the proposed processor shows high throughput even with the leaky-ReLU thanks to the slice-level sparsity exploitation. It can further improve its efficiency by utilizing weight sparsity induced by pruning.

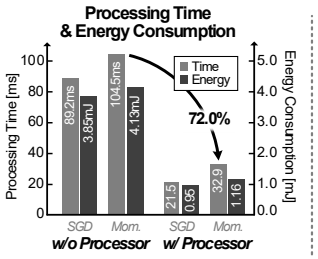
B. Object Detection Demonstration Results

The proposed processor is demonstrated with a lightweight object detection network, Tinyyolo-v3. Before it is tested for new video sequences, the network is pre-trained with the COCO dataset and the proposed processor performs both SR-based 8-bit quantization and pruning-aware training. The online DNN tuning is tested in YouTube-Objects (YTO)

TABLE I. DNN TRAINING PROCESSOR COMPARISON TABLE

		[1]	[2]	[3]	[4]	[5]	[6]	This Work
Backward Unlocking		O	X	X	X	X	X	O
Low-precision Training		DFXP	Q-Learning	X	X	FP8-SEB	SDFXP	DFXP+SR[11]
Supporting Sparsity		X	I/O	I/O	I/O/W	X	IS[2]	IS / OG[3] / W
Robustness for Non-ReLU		X	X	X	X	X	Zero-slice Skip	Zero-slice Skip
Technology		65nm	28nm	65nm	65nm	40nm	28nm	28nm
MAX Core Frequency		200MHz	268MHz	200MHz	200MHz	180MHz	250MHz	250MHz
Supporting Precision		FXP13/16	FXP2/4/8	FP8/16	FP8/16	FP8	FXP4/8/12/16	FXP4/8/12/16
Throughput [GOPS]	@ 0% Prune	155	137	1080	610	567	4526	4526
	@ 20% Prune				763		4526	7072
Area Efficiency[1] [(GOPS or GFLOPS)/mm²]	@ 0% Prune	26.9	24.3	33.3	38.1	77.8	349	349
	@ 20% Prune				47.7		349	545
Energy Efficiency[1] [(TOPS or TFLOPS)/W]	@ 0% Prune	0.62	3.81	1.67	1.44	2.47	4.74	5.05
	@ 20% Prune				1.79		4.74	7.89

1) Op. condition: MAX throughput, avg. value during Tiny-yolo-v3 online tuning (8-bit, Sparsity by ReLU: 0%, Slice-level Sparsity: 40%)
2) Input Slice (4b), 3) Output Gradient



< Object Detection Comparison Table >

	[7]	[1]	[8]	[6]	This Work
# of Detection	1	1	>1	>1	>1
Backward Unlocking	X	O	X	X	O
Operating Frequency	200 MHz	200 MHz	200 MHz	200 MHz	100 MHz
Framerate (FPS)	30.4	34.4	25.2	26.7	46.6
Energy Efficiency [mJ/frame]	4.14	4.88	2.13	1.68	0.95

Fig. 9. (1) Processing time and energy consumption of the proposed processor. (2) Object detection performance comparison table.

dataset and online DNN tuning is performed only when the detection probability is lower than a threshold. Thanks to the PCRU based pruning-aware acceleration, the energy efficiency can be improved by 1.6 × higher. The training efficiency is increased up to 3.9 × with the help of SR and ISS. The MLTA over programmable RAN further improves efficiency by 1.7 × higher and enables low-latency training with DFA, significantly lowering the required operating frequency. The proposed processor finally consumes an average of 0.95 mJ/frame which is the lowest energy consumption compared with the state-of-the-art processors as explained in Fig. 9. It also demonstrates the fastest object detection with 46.6 FPS compared to existing processors.

As described in Fig. 10, the proposed processor is able to reduce both processing time and energy consumption while maintaining high detection accuracy through online DNN tuning. It shows 15.6%p higher detection accuracy compared with the DNN without tuning case. The proposed processor is also examined with the broken lens environment and the results are shown in Fig. 11. User-customized lightweight DNN shows a low detection rate because it loses detection generality during personalization. Whenever the lens is broken, the proposed processor recognizes low detection probability and begins online DNN tuning. We verified that it can compensate for accuracy degradation caused by an unpredictable accident such as a camera malfunction. To sum up, the proposed processor recovers its detection accuracy and realizes robust object detection against environmental changes or incidents.

CONCLUSION

An energy-efficient and low-latency DNN training processor is proposed for robust object detection with real-world environmental adaptation. The iTRNG based SR enables 8-bit quantization and increases efficiency by 2.2 × higher. A combination of ISS and PCRU exploits sparsity even with the non-ReLU function and achieves 78.4% higher efficiency. When these features are combined with MLTA based BU supporting, it finally shows 6.6 × higher efficiency

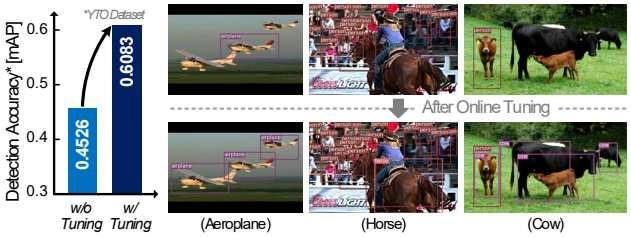


Fig. 10. Object detection accuracy and detection results in YTO dataset.

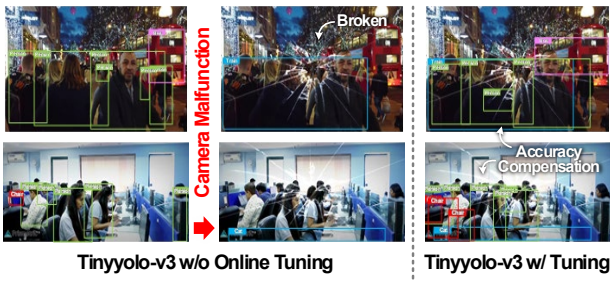


Fig. 11. Improved object detection performance in custom dataset (even with the camera malfunction).

and 4.2 × higher throughput compared with the baseline. In conclusion, the proposed processor successfully demonstrates the object detection application with the state-of-the-art energy efficiency, 0.95 mJ/frame, while maintaining 46.6 FPS real-time operation.

REFERENCES

[1] D. Han et al., "DF-LNPU: A Pipelined Direct Feedback Alignment-Based Deep Neural Network Learning Processor for Fast Online Learning," in IEEE JSSC, vol. 56, no. 5, pp. 1630-1640.

[2] F. Tu et al., "Evolver: A Deep Learning Processor With On-Device Quantization–Voltage–Frequency Tuning," in IEEE JSSC, vol. 56, no. 2, pp. 658-673, Feb. 2021.

[3] S. Kang et al., "7.4 GANPU: A 135TFLOPS/W Multi-DNN Training Processor for GANs with Speculative Dual-Sparsity Exploitation," IEEE ISSCC, 2020, pp. 140-142.

[4] S. Kim et al., "A 146.52 TOPS/W Deep-Neural-Network Learning Processor with Stochastic Coarse-Fine Pruning and Adaptive Input/Output/Weight Skipping," 2020 IEEE S. VLSI, 2020, pp. 1-2.

[5] J. Park et al., "9.3 A 40nm 4.81TFLOPS/W 8b Floating-Point Training Processor for Non-Sparse Neural Networks Using Shared Exponent Bias and 24-Way Fused Multiply-Add Tree," IEEE ISSCC, 2021, pp. 1-3.

[6] D. Han et al., "HNPU: An Adaptive DNN Training Processor Utilizing Stochastic Dynamic Fixed-Point and Active Bit-Precision Searching," in IEEE JSSC, vol. 56, no. 9, pp. 2858-2869, Sept. 2021.

[7] D. Han et al., "A Low-Power Deep Neural Network Online Learning Processor for Real-Time Object Tracking Application," in IEEE TCAS-I, vol. 66, no. 5, pp. 1794-1804.

[8] Y. Gong et al., "RAODAT: An Energy-Efficient Reconfigurable AI-based Object Detection and Tracking Processor with Online Learning," IEEE A-SSCC, 2021, pp. 1-3.

[9] M. Farhadi et al., "TKD: Temporal knowledge distillation for active perception," in Proc. IEEE WACV, Mar. 2020, pp. 942–951.

[10] D. Han et al., "Extension of Direct Feedback Alignment to Convolutional and Recurrent Neural Network for Bio-plausible Deep Learning," ArXiv: 2006.12830.

[11] S. Gupta et al., "Deep learning with limited numerical precision," in Proc. 32nd Int. Conf. ICML, vol. 37, 2015, pp. 1737–1746.

[12] Y. He et al., "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration," 2019 IEEE/CVF CVPR, 2019, pp. 4335-4344.

[13] L. Bassham et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST SP.