JNPU: A 1.04TFLOPS Joint-DNN Training Processor with Speculative Cyclic Quantization and Triple Heterogeneity on Microarchitecture / Precision / Dataflow

Je Yang, Sukbin Lim, Sukjin Lee, Jae-Young Kim and Joo-Young Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea
{yangje, sukbin.lim, ssjjlee, jykim1109, jooyoung1203}@kaist.ac.kr

***Abstract*—This paper presents JNPU, a 1.04TFLOPS joint-DNN accelerator that can simultaneously run joint-DNN (MobileNet + GoogLeNet) models with 245FPS (inference) and 1.26TFLOPS/W (training). It proposes speculative cyclic quantization that enables integer-dominant operations and reduces external memory access by 87.5%. Its tangram dataflow mapper provides optimized sets of heterogeneous stationary types for both forward and backward propagation, enhancing efficiency up to 71.6%. Lastly, its novel processing cluster leverages triple heterogeneity on INT8 arrays and FP16 vector processor, saving 56.3% and 26.9% of computing area and power, respectively.**

***Index Terms*—Metaverse, Deep neural network (DNN) training processor, Multiple DNN acceleration, Dataflow, Quantization**

I. Introduction

Metaverse is an upcoming technological megatrend aiming to create an immersive virtual world facilitated by augmented and virtual reality as well as internet technology. Emerging metaverse applications require various cognitive sub-tasks such as eye tracking, pose estimation, and object detection as basic functionality, but each sub-task is not trivial [1, 2]. With the recent advancements in machine learning technology, these primitive sub-tasks include multiple deep neural networks (DNNs) to run in real-time, demanding unprecedented computational requirements in handheld or wearable devices. In addition, user-specific service is crucial in metaverse, thus requiring on-device training of DNN models using private user data. However, inference/training multiple DNNs on a single chip poses new computational challenges as shown in Figure 1: 1) excessive external memory access (EMA) while switching multiple models. 2) limited reusability and efficiency of fixed dataflow due to various computational characteristics (e.g., operation types, layer sizes). 3) sufficient bit-precision for training accuracy.

In this paper, we present Joint Neural Processing Unit (JNPU) that highlights three main features for efficient multi-DNN inference/training. First, it introduces on-chip speculative cyclic quantization that enables integer-based operation and external/on-chip memory access reduction. Second, its tangram dataflow mapper supports all three output/weight/input stationary (OS/WS/IS) dataflow in inference and training to maximize data reuse. Lastly, it proposes a novel processing cluster that leverages heterogeneity in architecture, precision, and dataflow with INT8 arrays and FP16 vector processor for high computing efficiency. As a result, it shows high performance of 1.04TFLOPS on joint-DNN processing.

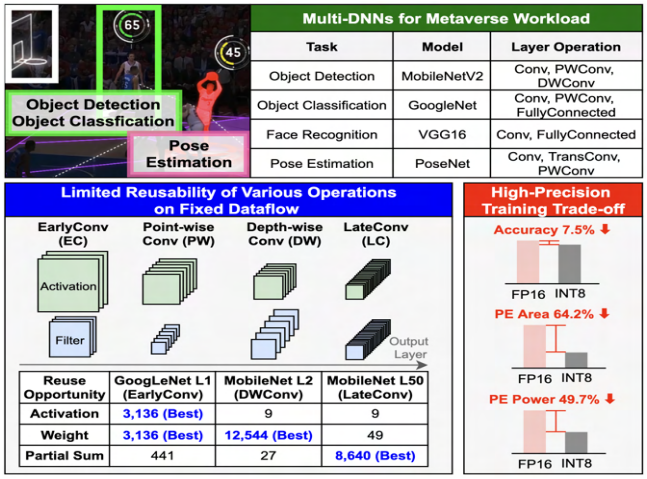


Fig. 1. Motivation of Joint-DNN Training Processor

II. JNPU Processor

Figure 2 illustrates the overall architecture of JNPU. It consists of 9 triple heterogeneity training clusters (THTC) connected by 3x3 level-2 (L2) mesh network. Single or multiple THTCs are responsible for running sub-layer tasks divided from the layer-wise joint-DNN workloads. THTC includes four BIG-INT8 array processors and one little-FP16 vector processor shared by the four arrays through spatiotemporal multiplexing, along with shared memory and cluster controller. The array processor is a key component for high throughput joint network training by supporting all three data stationary types through tangram dataflow mapper (TDM) and 8x8 INT8 heterogeneous dataflow PE array (IHPE Array). After convolution or matrix operation, the results are accumulated among other arrays in the same or adjacent clusters through the level-1 (L1) network and then sent to the vector processor. The vector processor takes charge of post-layer processing, such as batch-normalization (BN), non-linear activation (ReLU), and speculative cyclic quantization. It converts the INT8 data to FP16 data for high-precision accumulation and post-processing, then scales back to the INT8 type through a speculative cyclic quantization unit (SCQU).

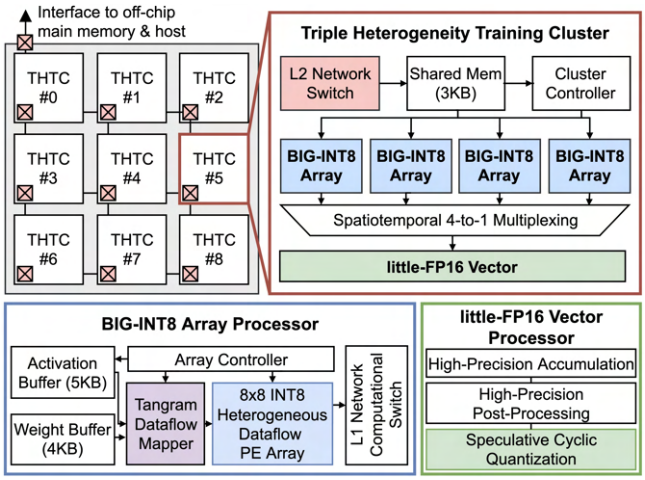


Fig. 2. Overall Architecture of Proposed JNPU Processor

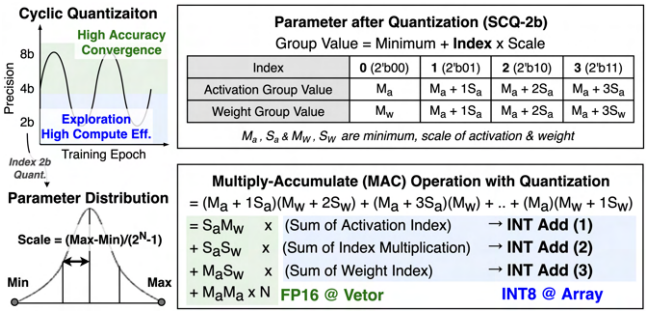
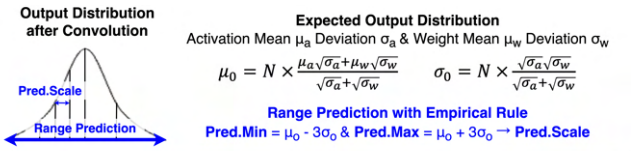


Fig. 3. Cyclic Quantization Algorithm and its Implementation in JNPU



Training Accuracy of SCQ

Network	MobileNet		GoogLeNet		VGG16		PoseNet
Dataset	Cifar-100	ImageNet	Cifar-100	ImageNet	Cifar-100	ImageNet	MPII
Baseline FP16	75.70	69.30	78.03	67.87	70.48	67.71	83.82
w/o Speculation	75.65	67.09	77.36	65.49	72.52	65.56	81.19
w/ Speculation	**74.40**	**67.18**	**77.15**	**65.51**	**72.55**	**65.53**	**80.9**

Fig. 4. Speculative Cyclic Quantization (SCQ) and Training Accuracy

III. Microarchitecture Design

A. Speculative Cyclic Quantization (SCQ)

JNPU adapts cyclic quantization algorithm [3], which uniformly divides all parameters in the layer into 2^N groups, where N is cyclically changing among 2, 4, and 8, exploiting both training accuracy at high-precision and computation efficiency at low-precision (Figure 3). It quantizes the parameters in each group to a common group value of *minimum+index×scale* in floating-point 16-bit format (FP16). JNPU minimizes the number of FP16 MAC operations by factoring out the index terms that can be done in integer 8-bit domain (INT8), resulting in only four FP16 multiplications with minimum and scale value for final output. While these calculated FP16 results should be quantized back to indexes for the next layer processing, the minimum and maximum value of the result cannot be fixed until the entire layer is finished, which makes it difficult to process quantization in place.

To address this problem, JNPU predicts the min/max of upcoming results, which is required to calculate scale value, based on the mathematical property that the convolution of two Gaussian-distributed inputs (activation and weight) also follows Gaussian distribution, as shown in Figure 4. By using the mean and deviation of the current layer's activation and weight ($\sigma_a, \mu_a, \sigma_w, \mu_w$), JNPU estimates those of the convolution result (σ_o and μ_o). It can further calculate the range and scale of the result, hence directly applying the quantization scheme inline. As a result, SCQ shows 99.3% hit rate and maintains the training accuracy, losing only 1.32% on average compared to the FP16 baseline. Moreover, it reduces EMA by up to 87.5% compared to the FP16, and reduces operation time up to 47.7% and 79.8% for compute- and memory-intensive layer, respectively.

B. Hierarchical Computational Network and Reuse-aware Heterogeneous Dataflow

JNPU employs computational on-chip network with hierarchy for efficient data transfer among THTCs in joint-DNN training. Figure 5 shows an example of the adaptive workload allocation with heterogeneous data stationary using the hierarchical computational network. The L2 network is responsible for data transfer at the THTC level, while the L1 network accumulates the partial sums among the array processors under the THTCs. The top controller selects the optimized dataflow for the model's target layer and allocates the required number of THTCs and accumulation path. This layer-wise allocation with computational on-chip network increases joint-DNN processing throughput by 18.4%.

Figure 6 illustrates the heterogeneous dataflows in detail. Tangram dataflow mapper (TDM) schedules the optimized dataflows by operating in two modes: output stationary (OS) and weight stationary (WS) mode. These modes are named based on the stationary that has the highest data reusability during the inference of a single layer. Each mode involves an optimized set of dataflow sequences that exploit different stationary for individual stages of inference/training, including forward propagation (FW), backward propagation (BW) and gradient generation (GG), to maximize the data reusability. When the layer shows higher reusability on partial sum accumulation during FW, TDM operates in OS mode. For FW, OS mode tiles both activation and weight in channel direction. Then, it aggregates the partial sums by broadcasting the activations and passing down the weights like conventional OS dataflow. For BW, OS mode tiles error in spatial direction to enable the identical order of access on the weight data. In contrast, WS mode takes WS/IS in FW and OS in BW, which is good for short channel accumulation or matrix multiplication. Both types take OS in GG to multiply each channel's activation and error. To this end, TDM changes the mapping of activations while storing the weight in output channel direction, allowing parallel computation over multiple

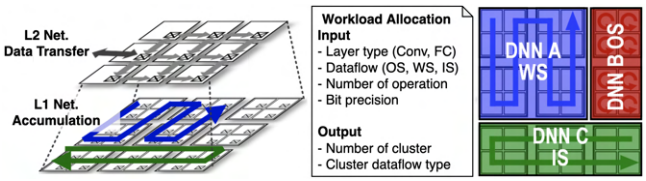


Fig. 5. Hierarchical Computational Network

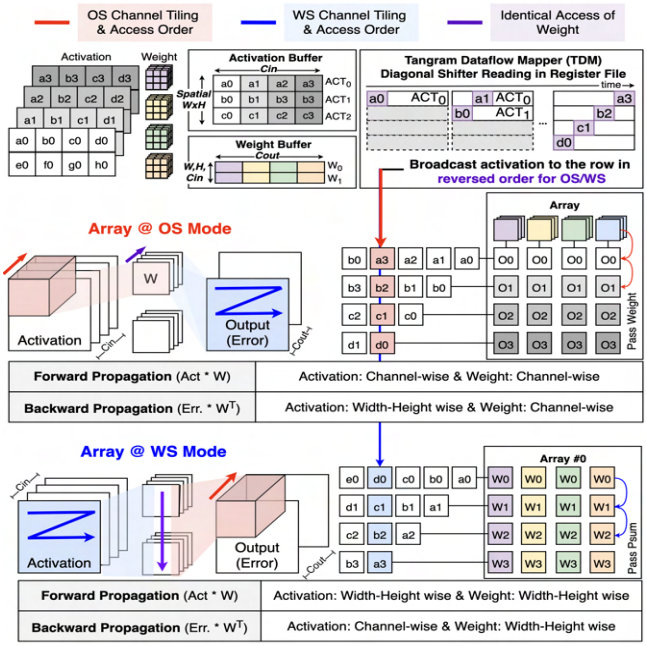


Fig. 6. Reuse-aware Heterogeneous Dataflow

kernels without dependency. It includes a register file (RF) that fetches data from the activation buffer and prepares the row data that enters into the array. It diagonally reads data from the RF with shifted addresses each time. Its data arrangement scheme supports both OS and WS/IS by simply reversing the read data and updating non-reusable row data to new ones. By supporting both heterogeneous dataflow and transpose operation with minimum hardware overhead, JNPU eliminates redundant data fetching/movement, increasing throughput up to 71.6%.

C. Triple Heterogeneity Training Cluster (THTC)

JNPU leverages a novel cluster that exploits triple heterogeneity in microarchitecture (array/vector), precision (INT8/FP16), and dataflow (OS/WS/IS). The BIG-INT8 array processor of Figure 7 employs an array of 8x8 INT8 heterogeneous dataflow PEs (IHPE) that performs the SCQ's MAC operation by separately accumulating the indexes of activation, weight, and product using integer operators. It supports heterogeneous dataflow through 2-to-1 muxes by keeping the partial sums in itself for OS or sending the results to the bottom of the array for WS/IS. The result of WS/IS enters the L1 network, where its computational switch adds or passes individually summated three 16-bit partial sums depending on the accumulation path programmed by the top controller. All accumulated results for convolution or matrix

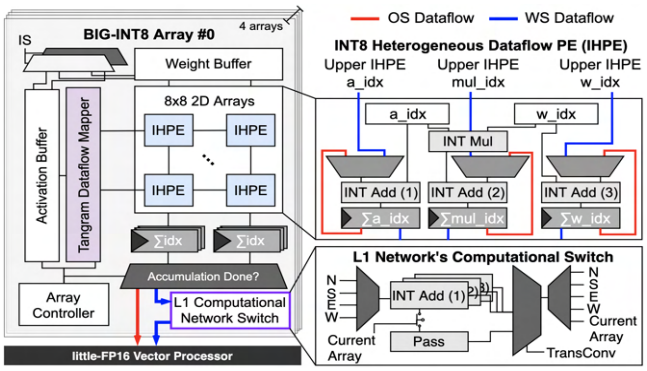


Fig. 7. BIG-INT8 Array Processor in THTC

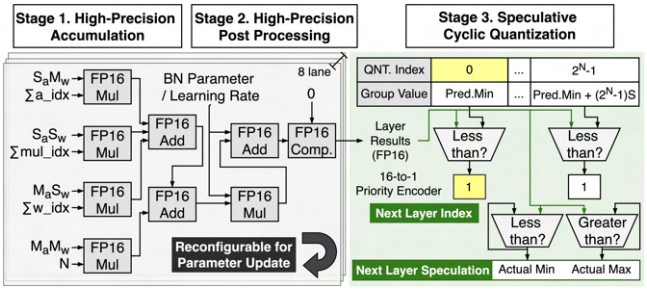


Fig. 8. Little-FP16 Vector Processor in THTC

operation are passed to the vector processor, regardless of dataflow.

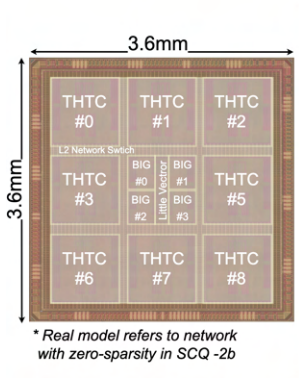
Figure 8 illustrates the operation flow of a little-FP16 vector processor. It has three fully pipelined operational stages to maximize the processing throughput and seamlessly serve the four array processors by sharing FP16 arithmetic units. The first stage generates the final results in FP16 by multiplying and summating the separate indexes with pre-calculated scale products. The second stage performs post-layer processing such as batch normalization and activation function. These two front stages can be reconfigured for parameter update in full precision. The last SCQ unit converts the FP16 results into INT8 indexes for the next layer. Based on the predicted min and scale value, it compares the result from previous high-precision stage with speculated group values to determine which index group the result belongs to. It also dynamically monitors the actual min/max value of the layer to speculate the next layer more precisely. Having integer-dominant operations with triple heterogeneity, JNPU saves the cluster's computing area and power by 56.3% and 26.9% compared to FP16-arrays only, respectively.

IV. Evaluation

Figure 9 shows the chip photograph of JNPU fabricated in 28nm CMOS technology, occupying 12.96mm^2 die area. It achieves the peak performance of 1.04TFLOPS at 200MHz and 1.1V, and the maximum efficiency of 1.70TFLOPS/W at 20MHZ and 0.79V. JNPU achieves 435.8FPS inference and 1.38TFLOPS/W training efficiency for MobileNet, and 245.4FPS inference and 1.26TFLOPS/W training efficiency for the joint-DNN workload (MobileNet+GoogLeNet). More-

TABLE I
PERFORMANCE COMPARISON WITH PREVIOUS TRAINING PROCESSOR
* RN REFERS TO THE RECONFIGURABLE NETWORK. ** REAL MODEL REFERS TO THE DENSE MODEL WITH ZERO SPARSITY.

	ISSCC'20 [4]	ISSCC'21 [5]	JSSC'21 [6]	S.VLSI'21 [7]	S.VLSI'22 [8]	**This Work**
Multi DNN Support	RN*	X	RN*	X	X	- Computational Network - Heterogeneous Dataflow
Technology	65nm	40nm	28nm	28nm	40nm	28nm
Die Area	32.4 mm^2	6.25 mm^2	12.96 mm^2	20.96 mm^2	7.73 mm^2	12.96 mm^2
Supply Voltage	0.7 - 1.1 V	0.75 - 1.1 V	0.58 - 1.04 V	0.58 - 1.0 V	0.62 - 0.9 V	0.79 - 1.1 V
Max Frequency	200 MHz	180 MHz	250 MHz	440 MHz	200 MHz	200 MHz
Precision	FP 8/16	FP8-SE8	SDFXP 4/8/12/16	FXP 4/8, FP 8/16	BFP 8	FP 16
Peak Performance [T(FL)OPS]	0.54 - 1.08	0.567	0.77 - 12.3	0.45 - 0.9	0.57	1.04
Real Model** Inference Efficiency [FPS/W]	17.91 (FP8, CycleGAN)	163.53 (FP8, ResNet-18)	375.19 (SDFXP16, YOLOv3)	N/A	N/A	550.6 (MobileNet) 57.1 (PoseNet) 310.0 (Joint-DNN)
Real Model** Training Efficiency [T(FL)OPS/W]	0.57 (FP8, CycleGAN)	1.64 (FP8, ResNet-18)	1.4 (SDFXP16, YOLOv3)	N/A	N/A	1.38 (MobileNet/PoseNet) 1.26 (Joint-DNN)



Technology	Samsung 28nm CMOS
Die Area	12.96mm^2
On-chip Memory	351 KB
Supply Voltage	0.79 - 1.1V
Frequency	20 - 200 MHz
Peak Performance	1036.8 GFLOPS
Max Energy Efficiency	1.70 TFLOPS/W
Real Model* Efficiency	1.26 - 1.38 TFLOPS/W
Real Model* Throughput	Maximum 435.8 FPS 45.2 FPS @ PoseNet 245.4 FPS @ Joint-DNN
Target Application	Edge Inference/Training on Joint Computer Vision Tasks

Fig. 9. JNPU Chip Photograph and Specification

over, it shows higher throughput than a real-time edge processing requirement (>30FPS, [9]) for all target applications. Table 1 shows the comparison results with previous training processors [4]–[8]. We evaluate the effectiveness of each inference and training phase of neural network, while taking FLOPS and power into account for fair comparison. Considering the precision of activation and weight, JNPU's joint-DNN processing is 13.5-125.0× and 1.97-9.68× more efficient for inference/training than state-of-the-art works.

V. CONCLUSION

In this paper, we propose JNPU, an energy-efficient multi-DNN training processor, to realize user-specific metaverse applications on mobile devices without any privacy concerns. JNPU supports joint-DNN training with high energy efficiency through speculative cyclic quantization. It quantizes the entire training parameter to less than 8-bit integer-type data, reducing external memory access while saving the computation power and area with the integer-dominant operation. Moreover, the triple heterogeneity training cluster supports all the data stationary types with a tangram dataflow mapper even in the training operation. Thanks to the algorithm/architecture co-design approach, JNPU successfully demonstrates 245.4FPS inference and 1.38TFLOPS/W training for the joint-DNN workload (MobileNet+GoogLeNet). As a result, JNPU achieves state-of-the-art performance both in real model inference speed (13.5-125×) and training efficiency (1.97-9.68×), enabling efficient processing of metaverse applications at the edge.

ACKNOWLEDGMENT

This research was supported by Institute for Information Communication Technology Planning and Evaluation (IITP) grants (No.2021-0-00871 and No.2022-0-01037) under the Ministry of Science and ICT (MSIT), Korea. The EDA Tool was supported by the IC Design Education Center (IDEC).

REFERENCES

[1] Kwon, Hyoukjun, et al. "XRBench: An Extended Reality (XR) Machine Learning Benchmark Suite for the Metaverse." arXiv preprint arXiv:2211.08675 (2022).
[2] Barham, Paul, et al."Pathways: Asynchronous distributed dataflow for ml." Proceedings of Machine Learning and Systems 4 (2022): 430-44.
[3] Fu, Yonggan, et al. "CPT: Efficient Deep Neural Network Training via Cyclic Precision." The 9th International Conference on Learning Representations 2021 (ICLR 2021). 2021.
[4] Kang, Sanghoon, et al. "7.4 GANPU: A 135TFLOPS/W multi-DNN training processor for GANs with speculative dual-sparsity exploitation." 2020 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2020.
[5] Park, Jeongwoo, Sunwoo Lee, and Dongsuk Jeon. "9.3 A 40nm 4.81 TFLOPS/W 8b floating-point training processor for non-sparse neural networks using shared exponent bias and 24-way fused multiply-add tree." 2021 IEEE International Solid-State Circuits Conference (ISSCC). Vol. 64. IEEE, 2021.
[6] Han, Donghyeon, et al. "HNPU: An adaptive DNN training processor utilizing stochastic dynamic fixed-point and active bit-precision searching." IEEE Journal of Solid-State Circuits 56.9 (2021): 2858-2869.
[7] Wang, Yang, et al. "A 28nm 276.55 TFLOPS/W sparse deep-neural-network training processor with implicit redundancy speculation and batch normalization reformulation." 2021 Symposium on VLSI Circuits. IEEE, 2021.
[8] Fu, Zih-Sing, et al. "A 40-nm 646.6 TOPS/W Sparsity-Scaling DNN Processor for On-Device Training." 2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits). IEEE, 2022.
[9] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh,"Real-time video inference on edge devices via adaptive model streaming," Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4572–4582