PCM: Precision-Controlled Memory System for Energy Efficient Deep Neural Network Training

Boyeal Kim*, Sang Hyun Lee*, Hyun Kim†, Duy-Thanh Nguyen‡, Minh-Son Le‡, Ik Joon Chang‡,
Dohun Kwon§, Jin Hyeok Yoo§, Jun Won Choi§, and Hyuk-Jae Lee*
*Department of Electrical and Computer Engineering, Seoul National University
†Department of Electrical and Information Engineering, Seoul National University of Science and Technology
‡Department of Electronics Engineering, Kyung Hee University
§Department of Electrical Engineering, Hanyang University
*{bykim, shleemark, hyuk_jae_lee}@capp.snu.ac.kr, †hyunkim@seoultech.ac.kr ‡{dtnguyen,sonlm,ichang}@khu.ac.kr,
§{dhkwon, jhyoo}@spa.hanyang.ac.kr, §junwchoi@hanyang.ac.kr

***Abstract*—Deep neural network (DNN) training suffers from the significant energy consumption in memory system, and most existing energy reduction techniques for memory system have focused on introducing low precision that is compatible with computing unit (e.g., FP16, FP8). These researches have shown that even in learning the networks with FP16 data precision, it is possible to provide training accuracy as good as FP32, de facto standard of the DNN training. However, our extensive experiments show that we can further reduce the data precision while maintaining the training accuracy of DNNs, which can be obtained by truncating some least significant bits (LSBs) of FP16, named as hard approximation. Nevertheless, the existing hardware structures for DNN training cannot efficiently support such low precision. In this work, we propose a novel memory system architecture for GPUs, named as precision-controlled memory system (PCM), which allows for flexible management at the level of hard approximation. PCM provides high DRAM bandwidth by distributing each precision to different channels with as transposed data mapping on DRAM. In addition, PCM supports fine-grained hard approximation in the L1 data cache using software-controlled registers, which can reduce data movement and thereby improve energy saving and system performance. Furthermore, PCM facilitates the reduction of data maintenance energy, which accounts for a considerable portion of memory energy consumption, by controlling refresh period of DRAM. The experimental results show that in training CIFAR-100 dataset on Resnet-20 with precision tuning, PCM achieves energy saving and performance enhancement by 66% and 20%, respectively, without loss of accuracy.**

***Index Terms*—Deep Neural Network, Approximate Computing, Precision Control, Refresh Period Control, General Purpose Graphic Processing Unit, High Bandwidth Memory**

I. Introduction

Deep neural network (DNN) is a widely used concept in various fields. In DNN, appropriate training process is the first step to achieve its unprecedented accuracy. However, the training process consumes a significant amount of energy. In particular, because a large amount of data, such as weight and activation maps, must be moved back and forth between the memory and the core, the memory system is a key bottleneck in the reduction of power consumption [1]. Another process whose power requirements are often underestimated is data retention, which needs to maintain a few terabytes of DRAM per pod on a warehouse-scale server [2]. The power consumption of such a memory system exceeds the energy consumption of the logic unit. For example, Alexnet, a representative convolutional neural network (CNN)-based algorithm known to have computational bounds, consumes 85% of the total energy required for the operation of the first convolution layer for data movement alone [1].

A popular method to reduce the energy consumption of memory systems, and even of logic units, is precision reduction. Proper precision reduction does not incur any overall accuracy loss for the DNN, while simultaneously diminishes energy consumption by reducing computation demands and data movement. For this purpose, data is represented using a small number of floating-point bits, like 16-bit (FP16) or 8-bit (FP8), instead of the commonly used 32-bit data format (FP32). A lot of recent studies have focused on effectively reducing precision [3]–[5]. However, this is limited by hardware constraints such as memory alignments and computation unit designs. Inefficiencies in such hardware components impede the search for the optimal precision that does not cause an accuracy drop. Therefore, the previous studies are not very feasible.

Meanwhile, research to reduce the energy consumption of data maintenance processes is being conducted orthogonally with DNN research [6]–[8]. Refresh operations in DRAM are the causes behind the high energy demands of data retention processes. Research to reduce the energy consumption of refresh processes [6], [7] generally involves skipping the refresh operation in the absence of errors. However, it is known that a larger amount of energy can be saved if errors are permitted, and many software and hardware techniques have been proposed based on this principle. However, the DRAM architectures of these studies [6]–[8] solely focus on saving energy used in refresh operations and do not consider data movement.

In this study, we propose a novel memory system architecture to save energy in both aspects, i.e., data movement and data retention, of the training process. In the proposed

This work was supported by Samsung Research Funding Center of Samsung Electronics under project Number SRFC-IT1602-03. H.-J. Lee is the corresponding author.

architecture, the process of transposing data is similar to the previous study [8], but data corresponding to different bit position, i.e., bit-slice, are stored in different rows on the DRAM. Further, a new data mapping is applied to meet the throughput requirements of the de facto deep learning training hardware, GPU with HBM, and the deep learning algorithm. The L1 cache is modified to bridge the format gap between the DRAM and the existing GPU core, while simultaneously eliminating cache underutilization by employing thread throttling and hardware prefetching. We also propose a new refresh scheme, soft-approximation, that can be applied to low precision deep learning training. As a result, our architecture eliminates the hardware inefficiencies that occur in the memory system during training using arbitrary precision by enabling hard-approximation which is the precision truncation mechanism. Therefore, the energy consumed by the proposed memory system is proportional to the degree of precision used. In addition, the architecture is proposed based on the transposed memory technique, which is also used in existing refresh energy saving mechanisms. So, the reduction in power consumption is at least as much as that achieved in previous research.

In summary, our contributions are:

1) **Transposed data layout for HBM** — We propose a memory mapping for the access in bit-slice unit on HBM, which minimizes throughput degradation in training workloads.
2) **L1 cache system design for bit-slice fetching** — We redesign the cache system for supporting the transposed data layout and enabling hard-approximation at the core. PCM reduces energy consumption caused by data movement by 38% and speed up the training process by 20% using hard-approximation.
3) **Exploration on non-power-of-2 precision learning** — Using the proposed system, we propose bit-level precision control and evaluate its potential by measuring training performance altering bit-level precision in the epoch unit.
4) **Refresh energy reduction scheme for low precision deep learning** — We reduce energy consumption of refresh processes in deep learning training by 85% by employing the proposed refresh skipping techniques and conclude that it has no substantial effect on overall accuracy.

II. Background and Motivation

A. Low Precision for Deep Neural Network Training

DNN training is a weight optimization process required before the inference phase. The weights determine the accuracy of the DNN and are thus crucial to the DNN process. In a DNN, the training process is generally a non-convex optimization problem; thus, it consists of an iterative algorithm. To perform the millions of iterations necessary, the training process consumes a significant amount of energy. Especially with the growing popularity of the neural architecture search technique, a technique to determine even network shape by using DNN training process is consuming more energy at once than a person uses during his entire life [9].

Fortunately, DNNs can retain their accuracy even without data of high numerical precision because they comprise a perceptron structure. Furthermore, using less precise data reduces energy consumption and speeds up the system. Therefore, recent complex DNN algorithms designed to produce high accuracy extensively use precision reduction techniques, and relevant research is being conducted actively [3]–[5]. However, as excessive reduction in precision can lead to severe losses in accuracy and hardware that supports such drastic reduction is unavailable, precision reduction remains at an acceptable level in practical industrial applications.

The state-of-the-art low precision technique used in DNN training is called Mixed Precision Training and it uses a mixture of half precision and single precision [3]. During the training process, the feature maps and the weights are converted to half precision via a computation unit, and the partial sum of convolution is accumulated in 32-bit. A training method using even lower precision, such as 1-bit, has also been proposed [10], but it shows unacceptable levels of accuracy for complex DNN tasks.

Nevertheless, as shown in Section IV, when training the CIFAR-100 dataset on Resnet-20, the same accuracy as mixed training can be achieved by using only 9-bit weights and feature maps. However, 9-bit data movement is very inefficient in general GPU architectures, which support only standardized precision such as FP8, FP16, and FP32. This arises from the additional work required to move and map unaligned data to multiple cores and ALUs. As a result, reducing the precision to 9-bit is rendered useless, and so many techniques proposed at the algorithm level cannot be applied in practical scenarios.

B. Approximate DRAM for refresh power reduction

Refresh is an operation to keep data in volatile DRAM. Refresh is expected to account for a large portion of energy consumption as DRAM chips grow in density [6]. This trend on refresh is equally applicable to server systems where DNN training is mainly performed, and the impact of refresh operations on server systems may be more severe than inference phase mainly performed at the edge device because server systems have a lot of acceleration hardware with high-capacity DRAM.

A lot of studies have conducted to solve this refresh problem [6]–[8], to basically skip the unnecessary refresh commands. Especially, the research in [8] varies the refresh period according to the importance of the data. In this method, during the inference phase based on FP32, the DRAM cell containing LSB (i.e., Mantissa bits) is refreshed at a low rate, and the MSB (i.e., Sign and Exponent bits) is protected from error with normal refresh rates. As a result, this technique reduces the power consumption of refresh processes by 70%. However, since the refresh operation is performed in a row unit, data are stored in the DRAM after being transposed to achieve precision-wise refresh control.

In fact, the transposed mapping proposed in this previous study can be utilized to control not only refresh periods by bit-slice but also data access (i.e., data movement) by bit-slice. Therefore, applying this bit transposed structure to control precision during DNN training phase can solve the hardware inefficiency problem presented in Section II-A. However, the naïve adoption of the transposed structure proposed in the previous study [8] results in the significant DRAM bandwidth underutilization due to the degradation of the row locality. This is because each bit-slice is located in different rows in the same bank, which causes excessive row changes that are time-consuming during the data fetching process. Therefore, the inefficiency of the transposed architecture must be addressed to construct a system that can effectively control both dynamic power and refresh power.

III. Proposed Hardware Architecture

We present a precision-controlled memory system, PCM, based on the motivation provided in Section II. PCM is a memory system for DNN training that supports loads and stores data corresponding to any degree of precision without any bandwidth loss. Consequently, both dynamic energy and refresh energy consumption can be reduced.

A. Architecture Overview

Fig. 1 shows the PCM architecture proposed in this study. PCM is based on discrete GPU with HBM, which is often used for server-scale DNN training device. In HBM, data is stored after transposition, and L1 and DRAM controllers of existing GPUs are modified to support the transposed mapping. In L1 cache, the internal SRAM structure is modified, and a write buffer and a prefetcher are added. The DRAM controller also includes a refresh controller.

The sub-unit in the PCM architecture works interactively to reduce energy consumption in memory systems with precision while minimizing performance degradation. To use the transposed data without bandwidth degradation, PCM maps these data by utilizing the parallelism of HBM. In addition, PCM controls the degree of precision in the modified L1, thereby making the dynamic energy consumption in data movement proportional to the degree of precision and minimizing IPC degradation. In addition, to reduce the refresh energy, the DRAM controller in PCM performs refresh skipping at a rate appropriate for the application.

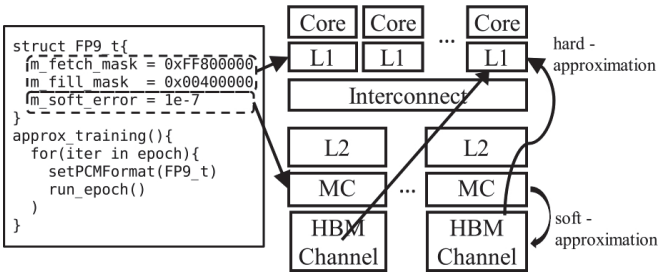

Fig. 1. Overview of the proposed PCM architecture

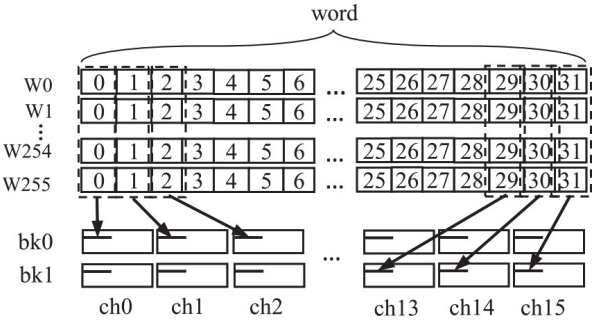

Fig. 2. Address mapping for transposed data

B. Address Mapping for Transposed Data

The proposed architecture provides precision controllability using a transposed data layout. In addition, we propose a data mapping method that uses the parallelism of HBM, which solves the issue of bandwidth degradation caused by the naïve transposed data mapping. The proposed mapping places each bit-slice in an independent DRAM array, such as a channel or a bank, minimizing the opening and closing of rows when multiple bit-slices are being accessed. In particular, since each channel operates independently, the degrees of precision which are less than the number of channels can be accessed with the latency of only one row change. This is faster than the naïve mapping by the number of bit-slices used.

Fig. 2 shows a schematic diagram of the proposed mapping. Each bit-slice is located in a different channel or bank. In the case of adjacent bit-slice, row level dependency is removed by mapping to other channels, and by placing them in different banks in different channels. The parallelism provided by the other HBM dies is excluded from use. This is because locating different bit-slice in different dies could increase latency variations and could stall the GPU core's pipeline until the slowest bit-slice is fetched. An example of the proposed mapping is as follows. When mapping 32-bit data to HBM, 0th—15th bit-slices are assigned to the 0th bank of each channel and 16th—31st bit-slices are placed on the 1st bank of each channel. In this way, each bit-slice is mapped to a different bank, ensuring independent access and refresh. Also, it guarantees that the worst access latency is the same as two accesses in different banks, not 32 accesses in the same bank.

However, since different bit-slices are located on different channels, data can no longer be reconstructed into word from the bit-slices in the DRAM controller as in the conventional method. Since the L2 cache is also channel-private on the GPU, the core-private L1 cache is the lowest memory hierarchy whose bit-slice chunk can be converted into word. In addition, since the DRAM request unit is 32B, the L1 cache line size should be 1KB for 32-bit data, which is very large. Therefore, a new L1 Data subsystem is required.

C. Precision-wise Fetching : Supporting Cache Hierarchy

In PCM, the L1D subsystem interacts with the DRAM data mapping proposed in Section III-B, thereby effectively

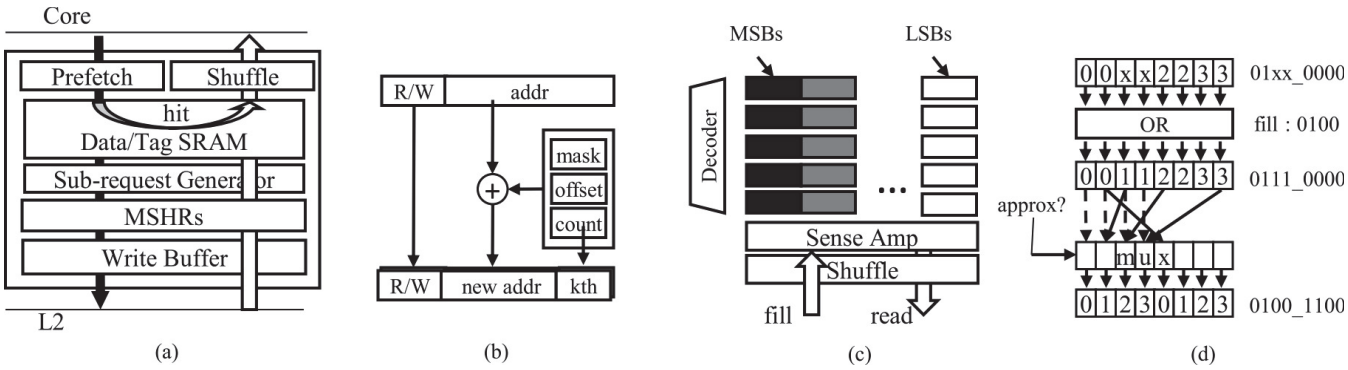

Fig. 3. L1 data cache and internal elements. (a) L1 overview, (b) sub-request generator, (c) modified data SRAM, and (d) shuffle logic

controlling hard-approximation. First of all, the L1 subsystem in the PCM decides which degree of precision to fetch for low precision data movement. It also reconstructs the fetched bit-slice into words to bridge the gap between the core and the memory system. Further, it replaces the values for the bit-slices that would not be fetched. In addition, appropriate techniques are applied to compensate for the lower cache efficiency caused by the larger cache line. The L1 cache with these features is depicted in Fig. 3-(a).

A sub-request generator, which is depicted in Fig. 3-(b), decides which bit-slice needs to be fetched from the memory and creates a memory request for each degree of precision called a sub-request. The sub-request generator contains a software-configurable register called the fetch mask. The fetch mask indicates which bit-slices to be fetched in the form of a bit flag. Sub-requests are created sequentially in the order indicated by the fetch mask. Unlike the original request, the address field in a sub-request is modified to point to the address of the bit-slice, and an additional field is added for bit-slice index. The generated sub-requests are managed in one MSHR entry in the form of a fetch mask. When each bit-slice arrives, the corresponding bit in the stored fetch mask is set. When all fields of the fetch mask in the MSHR entry have the value 1, the valid flag is set in the tag.

The conversion of bit-slices into words is impossible in the DRAM controller, and is therefore performed via the modified SRAM and the shuffle logic, as shown in Fig. 3-(c). First, sub-requests are collected in 1 KB cache line. Within cache lines, data are still stored in the order of their bit position. Word rearrangement is performed in the shuffle logic on the way to the core. Within the shuffle logic, there is a software-configurable register (i.e., fill mask) to replace vacant bit-slices via an OR operation. After the OR operation, data are reorganized from a sequence of bit-slices to that of words using a hard-wired path consisting of 2-input MUX. The example of 2 words with 4 bit-slices is shown in Fig. 3-(d).

Although PCM supports hard-approximation via L1 modification, two challenges are encountered. The first is a large L1 cache line. To maintain the same cache capacity, the number of cache line is reduced, which often results in conflict misses. The second is reduced write-efficiency. GPU writes usually bypass the L1 cache, and request size is generally less than 1 KB. However, in the PCM, a write request must be divided into sub-requests as well as a read-request, resulting in requests of inadequate size. This causes bandwidth degradation because the request size becomes less than that of the DRAM atom.

To solve the issue of conflict miss, our architecture employs two well-known optimization techniques that improve GPU cache utilization. The first is to limit concurrent thread numbers, which reduces the contention for cache lines [11]. In addition, a simple next line hardware prefetcher is included, which compensates for the latency hiding ability caused by the presence of fewer threads [12]. To solve the write request inefficiency, the PCM includes a write buffer, as depicted in Fig. 3-(d). In the write buffer, the write data is collected by a directly mapped 1 KB line, and the line is evicted when the line is hit and has been fully collected or another line is missed and it is selected as a victim line under LRU policy.

D. Deep Learning Training with the Proposed Memory System

Using PCM, DNN training with hard approximation is possible by simply setting a fetch mask and a fill mask. For example, 9-bit training begins by setting the fetch mask and fill mask to 0xFF800000 and 0x00400000, respectively, while storing a full precision copy of the weight and input data in the DRAM. Then, during training kernel execution, the weights and inputs are automatically hard approximated while being fetched into the core. Otherwise, in the DRAM, the refresh period is managed, MSBs (bit 0-–5) are protected, the bits used in the mantissa bits are skipped (bit 6—8), and the refresh of the remaining bits is ignored (bit 9-–32).

In addition, the fetch mask and refresh periods can vary between each epoch, mini batch, or layer with negligible overhead. Therefore, it is possible to change the precision finely in a specific epoch or layer. In this study, using the fact that the numerical range of weights is small as shown in [4], we begin with aggressively low precision and use 1 additional bit precision from each specific training iteration until the end. Simultaneously with the increase in the degree of precision, the refresh of newly used bits is started. This 1-bit increase scheme does not have any benefit in the existing architecture, but the PCM can execute it efficiently.

IV. Experimental Results

A. Experimental Methodology

This paper evaluates the performance of the PCM when training classification CIFAR-100 [13] dataset on Resnet-20 [14] network. The classification accuracy of various fixed precision and the proposed algorithm is measured through the simulation using a commodity GPU. In the proposed algorithm introduced in Section III-D, a combination of 7-bit (early epoch) and 9-bit (later epoch) is used because the minimum fixed precision without an accuracy loss is 9-bit. The error due to refresh skipping is modeled through bit error injection [15]. When applying refresh skipping, the sign bits and exponent bits are protected and the mantissa bits are injected with error.

To evaluate architectural performance of the PCM, we model the PCM in GPGPU-Sim [11]. This paper keeps the cache capacity constant while enlarging the size of the L1 cache line. Shuffle logic is estimated to be 2 cycles of L1 hit latency. For the baseline, a model of NVIDIA V100 is used. Detailed specifications including the PCM itself are depicted in Table I. Energy consumption in the memory system is calculated by architectural statistics obtained from the simulations using parameters from previous HBM study [16], refresh energy study [7], and models from CACTI [17]. To make the binaries run on GPGPU-Sim, we modify and build PyTorch v1.1 without cuDNN. Then, we obtain Resnet-20 training python code from open source and modify it using half precision data.

TABLE I
Hardware configuration

	Baseline	PCM
Arch.	80 SMs, 1.2 GHz, compute capability sm_60	
Core	2048 max threads 32 max CTAs	1024 max threads 2 max CTAs
L1D Cache	4 set, 32 KB, 32 B sectored, 64 ways, 128 B line 28 cycles hit latency	4 set, 32 KB, 32 B sectored, 8 ways, 1 KB line 30 cycles hit latency write buffer, prefetcher
DRAM	FRFCFS scheduler, BL=2, HBM timing	

B. Accuracy Evaluation

The experimental results for accuracy are depicted in Figures 4 and 5. Fig. 4 depicts the results of training with fixed precision. For the data format compatible with the existing hardware, the accuracy of FP16 is almost the same as that of FP32, but the accuracy is lowered as FP8 is used. However, this loss in accuracy does not occur for the intermediate numerical precision between FP8 and FP16. In the case of Top-1 accuracy, FP9 exhibits accuracy similar to that of FP16, while FP14 exhibits even higher accuracy than that of FP 16. These results show that, in the absence of hardware constraints, it is worth it to search for various precision and finally use the network-optimized precision. Note that existing GPU architectures cannot utilize this precision even if such precision exhibit high accuracy.

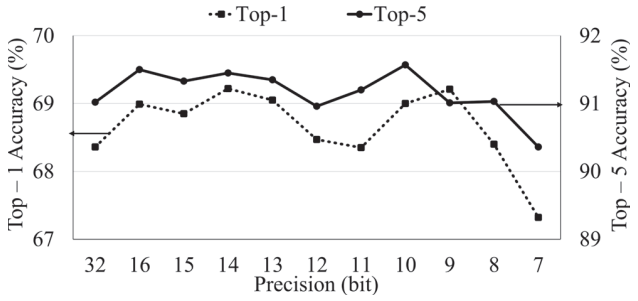

Fig. 4. Top-1 (dotted, left) and Top-5 (line, right) accuracy of Resnet-20 on CIFAR-100 with various numerical precision

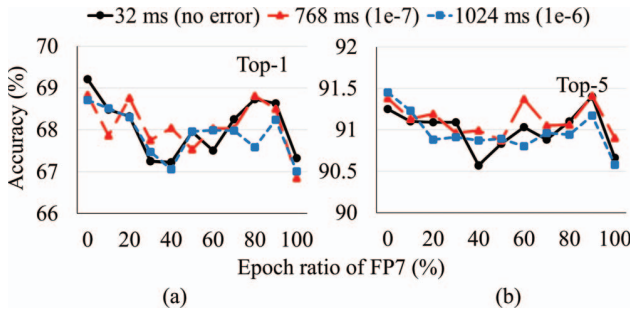

Fig. 5. Accuracy of proposed training algorithm according to epoch ratio of FP7 and refresh skipping rate. (a) Top-1 Accuracy (b) Top-5 Accuracy

The solid lines in Fig. 5 depicts the accuracy of the proposed algorithm in Section III-D, which uses FP7 for the early epoch and FP9 for the late epoch. In Fig. 5-(a), as the ratio of FP7 increases, top-1 accuracy tends to decrease; however, top-5 exhibits the same performance when running at FP7 at 10% epoch, while it exhibits a higher accuracy at 80% of FP7 as depicted in Fig. 5-(b). These results also effectively demonstrate the need for efficient hardware that can utilize the results of a fine-grained search for degrees of precision searching.

The dotted lines in Fig. 5 depicts the results of adjusting refresh skipping according to the proposed algorithm. The results show that there is no substantial change in accuracy compared to the result without error. Even increasing the refresh period of the mantissa bit to 1024 ms incurs no noticeable drop in the accuracy, and in some cases, the top-5 accuracy even improves. This improvement is caused by the fact that error injection has the effect of preventing overfitting. Therefore, with PCM, additional power reduction is possible through soft approximation without accuracy loss even after hard approximation, and the complete evaluation is presented in the next subsection.

C. Evaluation of the Proposed Hardware Architecture

The energy consumption of the PCM is depicted as a bar diagram in Fig. 6. In Fig. 6, the energy consumption for

fixed precision with a refresh period of 768 ms in mantissa bits is presented. There is a large improvement by 66% in energy consumption for the PCM, compared to the baseline architecture. This is because the refresh energy dominates the total energy consumption in the baseline and refresh energy is reduced by 85%. It should be noted that a large proportion of the reduction in the refresh energy is obtained from refresh skipping, not due to stopping refresh which is proportion to the degree of precision used. However, the refresh energy improvement has become saturated, since the sign and exponent bits are protected.

Fig. 7 depicts the dynamic energy consumption breakdown of the PCM. It is clear that the PCM uses less energy when using lower degrees of precision. Compared to FP9 and FP7, there are 35% and 40% dynamic energy improvements, respectively. For the proposed scheme, dynamic energy is reduced by 38% compared to the FP16 baseline. The breakdown results show that the energy consumption of the DRAM is significantly reduced compared to the baseline because of the higher L2 cache hit rate caused by thread limitations. Also, energy overhead from the write buffer, which is only added part of the PCM datapath, is negligible. Within the PCM results, as the degree of precision is lowered, the energy consumed in the L2 cache is proportionally reduced, resulting in precision reduction gain.

The effect of the PCM is not only apparent in energy consumption but also in speeding up the system. The training time is normalized to baseline and depicted via a line graph in Fig. 6. PCM using FP9 is 19% faster than the baseline and FP7 is 20% faster. The proposed algorithm is 20% faster. The reason why the proposed system can achieve such a fast speed compared to the baseline is that the number of DRAM accesses is reduced because of thread limitation. On the other hand, the difference between FP7 and FP9 is not so large, because Resnet-20 training is computationally bound. We can therefore confirm that faster DNN training is possible using the proposed system with significantly less power consumption.

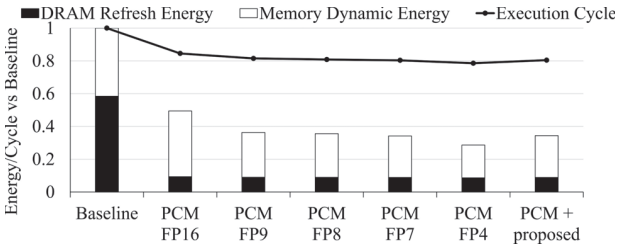

Fig. 6. Energy and cycle reduction of PCM normalized to baseline

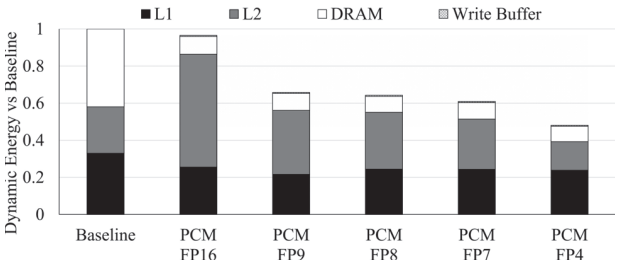

Fig. 7. Dynamic energy breakdown normalized to baseline

V. Conclusion

A PCM architecture proposed in this paper can reduce energy consumption scalably by enabling the control of degrees of precision in DNN training on GPU memory systems. In addition, it is possible to simultaneously reduce refresh energy, which takes up a significant proportion of the total power consumption. Therefore, PCM supports DNN by achieving a significant dynamic/refresh power reduction and speed-up of the system compared to existing GPUs. Experimental results show that while training CIFAR-100 on Resnet-20, the PCM system consumed only 34% energy and achieved a 20% speed-up compared to conventional GPU architectures.

References

[1] T. Yang, Y. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in the IEEE Conf. Computer Vision and Pattern Recognition, 2017, pp. 5687–5695.

[2] C. Chao and B. Saeta, "TPU V3 in Google Cloud: architecture and infrastructure," in IEEE Hot Chips 31 Symposium, August 2019.

[3] S. Narang et al., "Mixed precision training," in Int. Conf. on Learning Representation, 2017.

[4] U. Köster et al., "Flexpoint: an adaptive numerical format for efficient training of deep neural networks," in Adv. in Neural Information Processing Systems, 2017, pp. 1742–1752.

[5] D. Das et al., "Mixed precision training of convolutional neural networks using integer operations," in Int. Conf. on Learning Representation, 2018.

[6] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in Proc. Int. Symp. Computer Architecture, Jun. 2012, pp. 1-12.

[7] I. Bhati, Z. Chishti, S. Lu, and B. Jacob, "Flexible auto-refresh: Enabling scalable and energy-efficient DRAM refresh reductions," in Proc. Int. Symp. Comput. Archit., Jun. 2015, pp. 235-246.

[8] D. Nguyen, H. Kim, H. Lee, and I. Chang, "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in Int. Conf. on Circuits and Systems, 2018.

[9] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy consideration for deep learning in NLP," 2019. [online]. Available: https://arxiv.org/abs/1906.02243

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural network," in European Conf. on Computer Vision, 2016, pp. 525–542.

[11] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in 2009 IEEE Int. Symp. on Performance Analysis of Systems and Software, April 2009.

[12] J. Lee, N. Lakshminarayana, H. Kim, and R. Vuduc, "Many-thread aware prefetching mechanisms for GPGPU applications," in Proc. 43rd Annual Int. Symp. Microarchitecture, 2010, pp. 213–224.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images." Distributed by Toronto University, https://www.cs.toronto.edu/ kriz/cifar.html.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in the IEEE Conf. Computer Vision and Pattern Recognition, 2016.

[15] D. Mathew et al., "An analysis on retention error behavior and power consumption of recent DDR4 DRAMs," in Proc. IEEE Conf. on Design, Automation & Test in Europe, 2018, pp. 293–296.

[16] M. O'Connor et al., "Fine-Grained DRAM: Energy-Efficient DRAM for Extreme Bandwidth Systems," in Proc. 50th Annual Int. Symp. Microarchitecture, 2017.

[17] R. Balasubramonian, A. Kahng, N. Muralimanobar, A. Shafiee, and V. Srinivas, "CACTI 7: New tools for interconnect exploration in innovative off-chip memories," ACM Trans. Archit. Code Optim., vol. 14, no. 2, pp. 1–25, June 2017.