PQ-PIM: An Efficient Pruning-Quantization Joint Exploration Framework for Practical ReRAM-Based DNN Accelerator

Yuhao Zhang[†],Xinyu Wang[†],Xikun Jiang[†],Yuhan Yang[†],Zhaoyan Shen[†],Lei Ju[¶],Renhai Chen[⋆§],Zhiping Jia[⋆†]
[†]*School of Computer Science and Technology, Shandong University*
[¶]*School of Cyber Science and Technology, Shandong University*
[§]*College of Intelligence and Computing, Shenzhen Research Institute of Tianjin University, Tianjin University*

***Abstract*—Pruning and quantization are two efficient techniques to achieve performance improvement and energy saving for ReRAM-based DNN accelerators. However, most existing ReRAM-based DNN accelerators using pruning and quantization are based on an overidealized multi-bit ReRAM crossbar while neglecting the practical structure constraints. Due to the restriction of immature process technology, the actual matrix-vector multiplication must be conducted in a smaller operation unit (OU) granularity with single bit ReRAM cells. In this paper, we propose an efficient pruning-quantization joint exploration framework for practical ReRAM-based DNN accelerator, termed as PQ-PIM, which consists of a patch-wise pruning-quantization algorithm based on patch importance analysis to compress DNN models and a configurable mixed OU-based single bit ReRAM DNN engine to enable the algorithm with better performance and energy efficiency. Experimental results show that PQ-PIM achieves up to $1.74\times$ performance improvement, 62% energy saving, and $5.84\times$ compression ratio of occupied crossbars, compared to the state-of-the-art ReRAM-based DNN accelerator.**

***Index Terms*—ReRAM, DNN, pruning, quantization, accelerator**

I. Introduction

Deep Neural Networks (DNNs) have been popularly applied to a wide range of applications (e.g., image classification, speech recognition and object detection) due to their high inference accuracy. However, executing a modern DNN algorithm in conventional hardware platforms incurs massive data movement between the processing units and memory. Processing-in-memory (PIM) architectures, especially memristor-based accelerators, provide a promising solution for above challenge thanks to their integration of computation logic and memory. Among various memristor accelerators, resistive random access memory (ReRAM) supports in-situ matrix-vector multiplications with its crossbar structure, and has been extensively studied to represent synapses in neural computation [1]. However, it is imperative to require too many ReRAM cells to store the hundreds of millions of weights for a typical DNN model. Meanwhile, the intensive computation introduces performance and energy efficiency challenges to a PIM architecture.

Pruning and quantization are two efficient compression techniques for DNN models to reduce the storage requirement and accelerate the computation. These two important techniques have been separately explored in recently proposed ReRAM-based DNN accelerators to adapt the hardware characteristics for achieving performance improvement and energy saving.

Pruning. The irregular sparsity results from non-structured pruning algorithm [2] makes it hard to be utilized in tightly coupled ReRAM crossbar structure. ReCOM [3] and SNrram [4] thereby adopt the structural pruning algorithm to map the weights matrices in a dense manner. Nevertheless, these works have two weaknesses: (1) the structural pruning (e.g., pruning the whole filter or channel) is coarse-grained that incurs higher accuracy loss; (2) the expense of index for all activations and weights brings significant storage overhead. Although Sparse ReRAM Engine (SRE) [5] ameliorates the first weakness by exploiting the fine-grained bit sparsity of activations and weights, the second drawback remains.

Quantization. The coarse-grained quantization applied at network (e.g. binary and ternary weight DNN [6], [7]) or layer (e.g. multi-precision weight DNN [8]) level has been exploited in ReRAM-based DNN architectures. However, the accuracy loss of binary or ternary weight DNN is unacceptable due to the extremely limited weight precision. Although the multi-precision weight DNN alleviates the effect of quantization on accuracy, the design of using multiple crossbars to store multi-bit weights introduces massive intermediate data transfer for aggregation of results and complex control logic to guarantee the correctness of computation, which sacrifices the performance and energy efficiency.

Pruning eliminates the redundancy in the number of weights, whereas quantization removes the redundancy in bit representation of weights. Ideally, these two categories of redundancy can be explored coordinately, thereby leading to a higher degree of DNN model compression [9]. Prior works have proposed memristor-based DNN accelerators with structured weight pruning and quantization [10], [11]. However, these accelerators are based on overidealized ReRAM device characteristics. First, they utilize the multi-bit ReRAM devices, where an ReRAM cell can represent multiple bits of a weight. Nevertheless, it is difficult to fabricate and tape out a multi-bit ReRAM device with considerable storage space to satisfy hundreds of millions of parameters. Meanwhile, the multi-bit ReRAM suffers the problems of resistance non-linear and variation, which have not been addressed well [8]. Second, they assume that an entire ReRAM crossbar array with size of 128×128 or larger can be activated currently in one single cycle. However, the accumulated current deviation of per-cell greatly harms the inference accuracy with so many cells in a crossbar structure [5]. Considering the hardware limitations, the practical ReRAM-based DNN accelerator must perform matrix-vector multiplication in a smaller operation unit (OU) granularity (e.g., 8×8, 16×8) with single bit [5], [8], [12]. Unfortunately, the prior structured pruning and quantization schemes do not directly apply to OU-based single bit ReRAM devices.

In this paper, we propose an efficient pruning and quantization joint exploration framework for practical ReRAM-based DNN accelerator. Initially, we carefully analyze the characteristics of the weight and activation patches (i.e., several consecutive values) of DNN models. We make the key observation that patches of the same size with different numbers of smaller values have different impacts on the DNN accuracy. Hence, we classify the weight patches into three categories of importance and the activation patches into two categories of importance. We propose a patch-wise pruning-quantization algorithm, which prunes unimportant patches and quantizes weak importance patches. Furthermore, we propose a configurable mixed OU-based single bit ReRAM DNN engine. Since a patch of the weight can just be mapped to an OU within the ReRAM crossbar structure, the invalid OU-based computation can be eliminated in the designed ReRAM engine by pruning and quantizing patch.

To enable the pruning and quantization co-design for practical

⋆Corresponding author: Zhiping Jia and Renhai Chen; E-mail: jzp@sdu.edu.cn; renhai.chen@tju.edu.cn.

OU-based single bit ReRAM accelerators, two challenges must be addressed as follows: (1) how to automatically seek the optimal compression ratio while maximizing the accuracy of DNN models; (2) how to design the OU-based single bit ReRAM DNN engine with minimal overhead to support the fusion algorithm of pruning-quantization to achieve better gains. To address the first challenge, we adopt the particle swarm optimization (PSO)-based search method to automatically find the optimal parameters related to compression ratio for each layer of DNN models. To handle the second challenge, we design weight patch importance table (WPIT) and activation patch importance table (APIT) for the proposed ReRAM engine to enable the patch-wise pruning-quantization algorithm efficient.

In summary, this paper aims to improve performance, reduce energy consumption and storage overhead for practical ReRAM-based DNN accelerator. To this end, we thoroughly analyze the importance of the weight and activation patches of DNN models, and we propose a fine-grained pruning-quantization fusion framework, termed as PQ-PIM. PQ-PIM consists of two parts: (1) a patch-wise pruning-quantization algorithm that significantly compresses the DNN models with negligible accuracy loss; (2) a configurable mixed OU-based single bit ReRAM DNN engine that enables the algorithm with better performance and energy efficiency. We implement a custom cycle-accurate simulator to evaluate the proposed PQ-PIM with several popular DNN models. The evaluation results show that compared with the state-of-the-art ReRAM-based DNN accelerators, PQ-PIM delivers significant performance improvement, energy saving and compression ratio of occupied crossbars. The main contributions of this work are as follows:

- We comprehensively analyze the importance of the weight and activation patches for several popular DNN models and classify them into different categories.
- We propose a fine-grained pruning-quantization fusion framework for practical ReRAM-based DNN accelerator, termed PQ-PIM. PQ-PIM consists of a patch-wise pruning-quantization joint algorithm and a configurable mixed OU-based single bit ReRAM DNN engine.
- We adopt PSO-based search method to automatically seek the optimal compression ratio in algorithm level and we design WPIT and APIT for the proposed ReRAM DNN engine in hardware level. Both to make the framework work efficiently.
- We implement a cycle-accurate simulator and evaluate PQ-PIM with several standard DNN models. Evaluation results show that PQ-PIM achieves up to $1.74\times$ performance improvement, 62% energy reduction and $5.84\times$ compression ratio of occupied crossbars on popular deep learning benchmarks, compared with the state-of-the-art ReRAM-based DNN accelerators.

The rest of the paper is organized as follows: Section II gives the background and motivation of this work. Section III presents the patch-wise pruning-quantization joint algorithm. Section IV describes the design of the OU-based single bit ReRAM DNN engine. Section V evaluates the proposed PQ-PIM. Finally, Section VI concludes the paper.

II. Background and Motivation

A. ReRAM-based DNN Accelerator Architecture

The typical ReRAM-based DNN accelerator architecture implements a hierarchy [1], [5], as shown in Figure 1. The hierarchical architecture is composed of a number of processing engines (PEs). Each PE mainly has an on-chip buffer to store intermediate data, a non-linear unit to support the execution of the non-linear function,

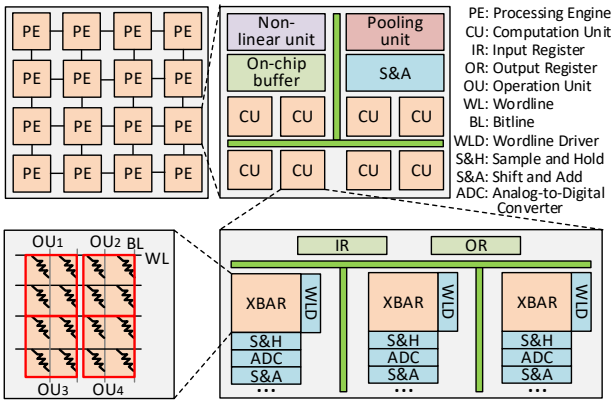

Fig. 1. ReRAM-based DNN accelerator architecture.

a pooling unit to achieve the computation of pooling layers, and multiple computation units (CUs). All the units are connected with on-chip interconnects. Each CU mainly consists of numerous crossbar arrays, which are responsible for performing the analog matrix-vector multiplications in convolution and fully-connected layers. ReRAM cells in each crossbar array represent filter weights as conductance. A wordline driver (WLD, e.g., digital-to-analog converter, DAC) is connected to each wordline of the ReRAM crossbar array to convert the activation values to input voltages. By applying external voltages to each wordline, it passes current into bitlines based on Kirchoff's Law. A sample-and-hold (SH) circuit receives the accumulated current at the end of each bitline and feeds it to a shared analog-to-digital converter (ADC). The values converted by ADC are the matrix-vector multiplication results of activation values and filter weights.

B. Practical OU-based Single bit ReRAM Computation

The generic ReRAM crossbar arrays have a large size (e.g., 128×128, 256×256). Each ReRAM cell in a crossbar array represents multi-bit (e.g., 2-bit [1], 4-bit [10], [11]). Assuming an entire multi-bit ReRAM crossbar array is activated concurrently, it will pose serious inference accuracy degradation due to the accumulated effect of per ReRAM cell current degradation. Thus, ReRAM-based matrix-vector multiplication must be performed in a smaller operation unit (OU) granularity (e.g., 8×8, 16×8) with single bit [5], [8], [12].

Figure 2 shows a simple example of OU-based matrix-vector computation with single bit for a convolution layer. The convolution layer has a $4\times4\times1$ input feature map and three $4\times4\times1$ filters. Each activation in input feature map is 4-bit precision, which is separated from the most significant bit (MSB) to the least significant bit (LSB) groups due to the 1-bit WLD resolution. Each weight in filters is 4-bit precision. It requires four ReRAM cells to represent a weight. Each OU crossbar array with 4×4 size can store 4 weights. The weights of the three filters require 12 OUs. For computation, both the MSB data to the LSB data are fed into wordlines sequentially and need to activate the 12 ReRAM OUs separately.

C. Motivation

Previous overidealized ReRAM-based DNN accelerators miss the opportunity of pruning-quantization joint exploration from a practical perspective. Therefore, we are motivated to design a framework for practical ReRAM-based accelerator to jointly prune and quantize the DNN models for further performance improvement and energy saving.

In this paper, we define n consecutive values in both input feature maps and filters as a n-size patch, such as ❶∼❹ with 4-size

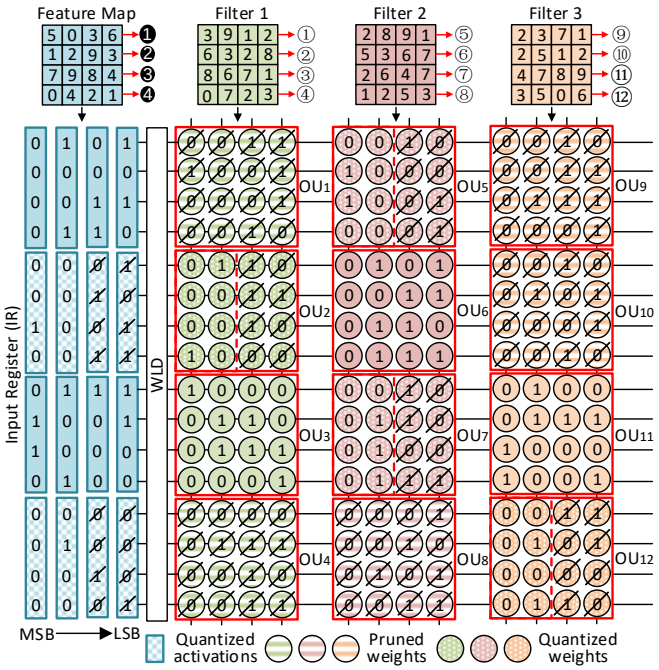

Fig. 2. Pruning-quantization joint exploration in ReRAM crossbars.

activation patches and ①∼⑫ with 4-size weight patches in Figure 2. Initially, we analyze the patch characteristics by conducting some preliminary experiments on several typical DNN models and observe an interesting phenomenon. We observed that the patches of same size with different numbers of small values have different impacts on the DNN model accuracy, which we call *various patch importance*. To be specific, patches with smaller *proportion of small values (PSV)* are more important than other patches to DNN accuracy. As the *PSV* increases, the importance of patches decreases. Moreover, activation patches are more sensitive than weight patches. In Figure 3 and Figure 4, we respectively plot the impact on accuracy for different size patch (n = 4, 8, 16, and 32) of weight and activation with various *PSV*. Take of VGG16 as the example, when we pruned the 32-size weight patches with PSV greater than 0.5, the accuracy did not decline. Meanwhile, when we quantized the 32-size activation patches with PSV greater than 0.7, the accuracy has no loss as well. However, with the decrease of PSV, patch has more impacts on accuracy.

Based on the above observation, we classify the widespread weight patches into three categories of importance according to their *PSV*. (1) Weight patches with higher *PSV* have no impact on accuracy, termed as unimportant patches. (2) Weight patches with moderate *PSV* have subtle impact on accuracy, termed as weak important patches; (3) Weight patches with smaller *PSV* have huge impact on accuracy, termed as important patches. Meanwhile, we classify the activation patches into two categories of importance: important patches and weak important patches. For unimportant patches, we prune them. For weak important patches, we quantize them. For important patches, we retrain them to recover the DNN accuracy. In other words, we combine pruning and quantization techniques for DNN models with negligible accuracy loss. Figure 2 shows an example of pruning-quantization joint exploration in OU-based single bit ReRAM crossbars. In this example, values less than or equal to 4 are considered small values. The weight patches ①④⑧⑨⑩ ($PSV \geq 0.75$) are pruned, and the corresponding OU1, OU4, OU8, OU9, OU10 can be removed directly. The weight patches ②⑤⑦⑫ ($0.25 < PSV < 0.75$) are quantized, and the corresponding OU2, OU5,

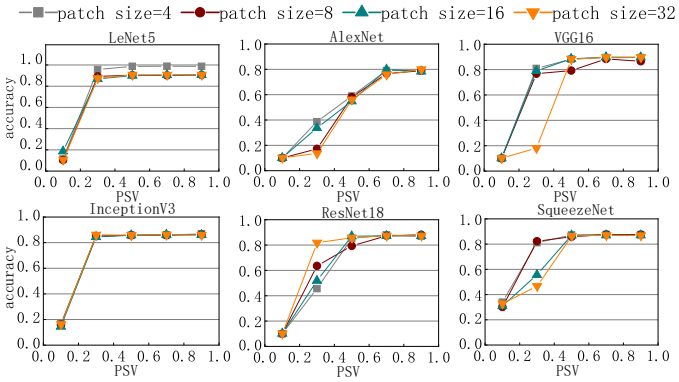

Fig. 3. Weight patch impact on DNN accuracy.

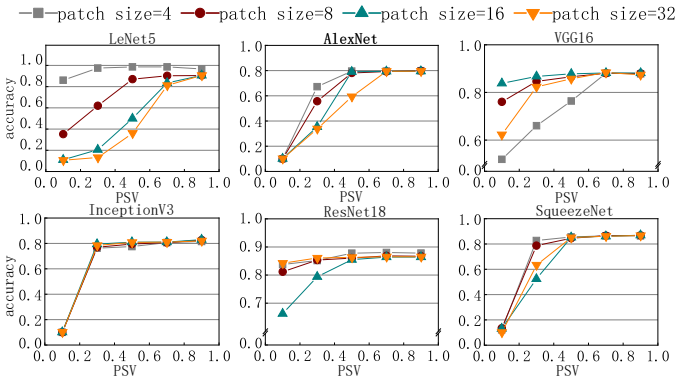

Fig. 4. Activation patch impact on DNN accuracy.

OU7, OU12 can be reduced from 4×4 to 4×2 size. Moreover, the activation patches ❷❹ ($PSV \geq 0.75$) are quantized, and the corresponding bits of inputs can be reduced. All the above methods will reduce storage overhead and improve performance and energy efficiency.

Therefore, in this paper, we propose an efficient pruning-quantization joint exploration framework–a software and hardware co-design. The framework consists of a patch-wise pruning-quantization joint algorithm and a configurable mixed OU-based single bit ReRAM DNN engine design.

III. Patch-wise Pruning-Quantization Joint Algorithm

In this section, we first present an overview of the proposed patch-wise pruning-quantization joint algorithm, then give the design space exploration of the algorithm.

A. Algorithm Overview

Figure 5 depicts the overview of the patch-wise pruning-quantization joint algorithm. The algorithm is consisted of two stages: detection stage and compression stage. The detection stage is used to predict the importance degree for both patches of input feature map and filter. In this stage, we first divide activations and weights into the same size patches. Then we detect the *PSV* for each patch. The *PSV* is compared with the preset threshold to determine the category of importance degree of patch. The compression stage is responsible for eliminating redundancy of DNN. In this stage, we decide to prune, quantify or retain patches according to their importance. Note that it is different for activation patch and weight patch in operation mode. The activation patch is generated dynamically during the DNN inference process, hence we need to compress them online. To this end, we generate the activation patch flags, which can distinguish the category

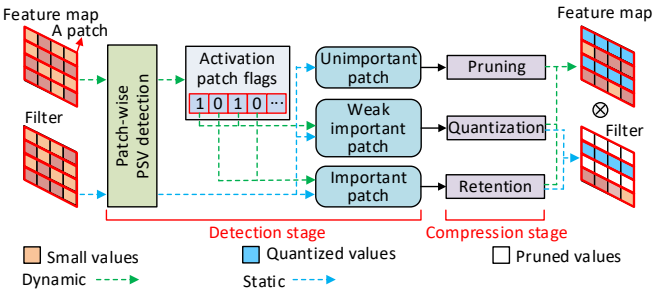

Fig. 5. The overview of patch-wise pruning-quantization algorithm.

Algorithm 1 PSO-based Parameters Search Method

1: Initialize parameter vector $V_i \leftarrow [TSV, PS, PSV](i \leftarrow 1 \ldots m)$, update speedup S_i, accuracy loss threshold L_t and iterations N;
2: **for** each iteration $j \leftarrow 1$ to N **do**
3: **for** each parameter vector V_i **do**
4: Computing the compression ratio C_i and the accuracy loss L_i;
5: **if** C_i>individual_max_compression_ratio C_{im} and $L_i < L_t$ **then**
6: individual_max_compression_ratio $C_{im} \leftarrow C_i$;
7: individual_optimal_parameters_vector $V_{io} \leftarrow V_i$;
8: **end if**
9: **end for**
10: Update S_i according V_{io};
11: swarm_max_compression_ratio $C_{sm} \leftarrow max(C_i)$;
12: swarm_optimal_parameters_vector $V_{so} \leftarrow V_i$;
13: **if** C_i>swarm_historical_max_compression_ratio C_{sm} and $L_i < L_t$ **then**
14: swarm_max_compression_ratio $C_{sm} \leftarrow C_i$;
15: swarm_optimal_parameters_vector $V_{so} \leftarrow V_i$;
16: **end if**
17: **end for**
18: **return** Parameter vector V_i

of importance for activation patch in online process. For the flags, "1" indicates that the patch is weak important, while "0" indicates that the patch is important. On the contrary, the weight patch can be conveniently compressed offline.

B. Design Space Exploration

Obviously, the compression ratio and accuracy of DNN algorithm are determined by three metrics: the threshold of small values (*TSV*), the patch size (*PS*), and the threshold of *PSV* (*TPSV*). Meanwhile, the higher of the compression ratio leads to the lower of ReRAM storage overhead, which can improve the performance and energy efficiency. It is extremely complicated to find the optimal parameters of these metrics to achieve the maximum model compression ratio while ensuring the accuracy. Fortunately, particle swarm optimization (PSO) algorithm, which is inspired by the foraging behavior of birds [13], is proposed to solve the optimization problem. Therefore, we adopt the PSO algorithm to search the optimized parameters of three metrics.

Algorithm 1 shows the complete search process. As shown, a set of parameters *TSV*, *PS*, and *TPSV* are combined to form a parameter vector V_i. First, we initialize the following parameters: (1) parameter vectors, where *TSV* starts from values derived from the prior works [14], *PS* and *PSV* are empirically chosen; (2) update speedup S_i, which is how fast a parameter vector changes to a better parameter vector; (3) accuracy loss threshold L_t of DNN; (4) global iterations N (Line 1). Then we compute the compression ratio and the accuracy loss for each parameter vector V_i and select the maximum compression ratio in the individual history that satisfies the accuracy constraint. The corresponding V_i is used as the individual optimal parameter vector (Line 3-9). After that, we update S_i for each parameter vector according to the individual optimal parameter vector (Line 10). Finally, the parameter vector of the individual with the largest compression ratio in the current swarm is temporarily selected as the optimal (Line 11-16). This procedure is executed iteratively until it reaches the maximum iterations N. The optimal parameter vector of swarm is returned, which the corresponding *TSV*, *PS*, and *TPSV* are obtained.

The PSO-based parameters search algorithm is executed in the DNN training process. The searched parameters are set to joint prune and quantize DNN models, which can speed up the inference process.

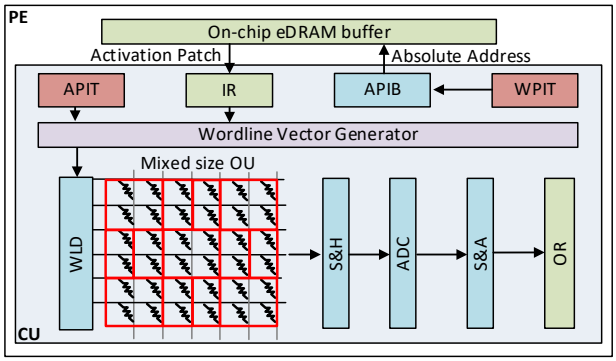

Fig. 6. Architecture Overview.

IV. OU-based single bit ReRAM Engine Design

In this section, we first present an architecture overview of the proposed ReRAM DNN engine, and then show the dataflow.

A. Architecture Overview

Figure 6 illustrates the overview of the proposed ReRAM DNN engine. The engine mainly consists of weight patch importance table (WPIT), activation patch importance table (APIT), activation patch index buffer (APIB), on-chip eDRAM buffer, wordline vector generator, mixed size OU, and other peripheral circuits. WPIT records the information of OUs pruning. It consists of a sequence of "0" and "1" flag bits. The flag bit "0" implies that the corresponding OU is pruned. APIT records the information of activation quantization. It also consists of a sequence of "0" and "1" flag bits. The flag bit "0" implies that the corresponding activation patch is quantized. APIB is used to store the activations index information. It stores an index for an activation patch to reduce the storage overhead. The on-chip eDRAM buffer is used to store activation patch. The wordline vector generator produces the wordline activation vector at each cycle to support quantized activation patch. The mixed size OUs are used to execute matrix-vector multiplications.

B. Workflow

The workflow of the ReRAM engine consists of two fundamental stages: offline stage and online stage. In the offline stage, the weight matrices are compressed using patch-wise pruning-quantization joint algorithm and programmed into the OUs. The corresponding information in WPIT and APIB is also generated. In the online stage, as shown in Figure 7, the engine first reads the flag bits of WPIT in order. If the flag bit is "1" (marked by yellow), it fetches the

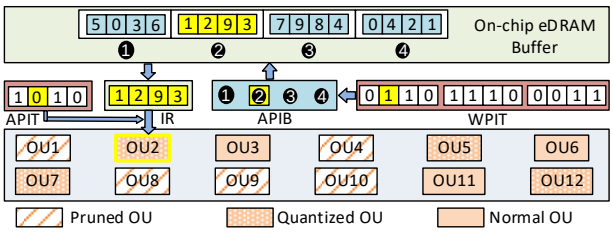

Fig. 7. Workflow of ReRAM engine.

TABLE I
HARDWARE CONFIGURATION.

PE configuration (1.2GHz, 32nm process, 352 PE in total)		
Component	**Spec**	**Power**
On-chip Buffer	size: 64KB; number banks: 4	20.7 mW
Pooling unit	number: 1	0.4 mW
Non-linear unit	number: 2	0.52 mW
CU configuration (12 CUs per PE)		
S&H	number: 8×128	10 uW
S&A	number: 4	0.2 mW
ADC	number: 8; resolution: 6 bits; frequency: 1.2GSps	5.14 mW
DAC	number: 8×128; resolution: 1 bit	4 mW
Memristor Array	number: 8; size: 128 × 128; bits-per-cell: 1;	2.4 mW
IR	size: 2KB	1.24mW
OR	size: 256B	0.23mW

corresponding index of activation patch (❷) in APIB. Then, the engine fetches the activation patch $(1, 2, 9, 3)$ in eDRAM buffer into input buffer (IR) according to the index. Finally, the engine reads the flag bit of above activation patch ("0", marked by yellow) in APIT to get the quantized information. The quantized activation patch activates the corresponding OU (OU2) as input.

V. EXPERIMENTS

In this section, we first present the experimental setup, and then show the experimental results and analysis of the proposed PQ-PIM framework.

A. Experimental Setup

We implement a custom cycle-accurate simulator based on a behavior level simulation platform MNSIM [15] to evaluate the performance, energy consumption, and ReRAM crossbar compression ratio of the proposed PQ-PIM. For the simulation, we make a reference to hardware configuration from ISAAC [1]. Table I summarizes the hardware configuration of each PE. Each PE has 12 CUs, and each CU has 8 crossbar arrays. The crossbar array size is 128×128 and each ReRAM cell only stores 1 bit. The OU size is set according to the patch size and per weight precision. The energy consumption of all memories, including the on-chip buffer, IR, and OR, is modeled using the CACTI at 32nm process.

The benchmarks that we used comprise five typical DNN models (i.e., LeNet5, AlexNet, VGG16, ResNet18, SqueezeNet). They are trained based on MNIST and CIFAR-10 datasets. For a fair comparison, we use two OU-based state-of-the-art ReRAM-based DNN accelerators as baselines. One of the accelerators (called SRE) [5] exploits pruning method, while the other (represented as CRMP) [8] studies quantization method. We evaluated our design with 4-size, 8-size, 16-size, and 32-size patch. The patch sizes are locally optimal parameters searched by PSO-based method. The precision of weight patch and activation patch is 8-bit or 4-bit. For the comparison results, we first show the accuracy and compression ratio of DNN models after using patch-wise pruning-quantization algorithm. We then present the performance, energy consumption and ReRAM resource utilization for PQ-PIM. Last, we analyze the WPIT and APIT overhead of the framework.

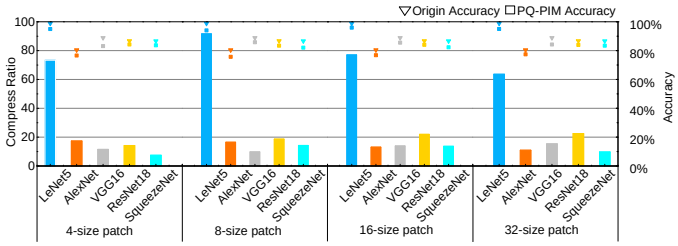

Fig. 8. The accuracy and compression ratio of DNN models.

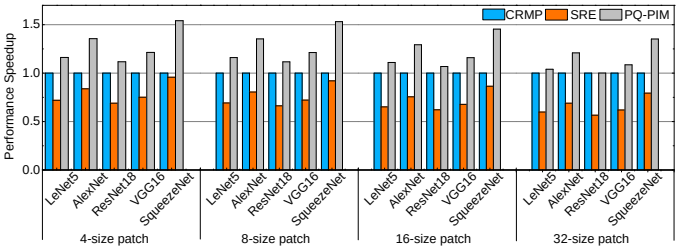

Fig. 9. Performance comparison.

B. Experimental Results and Analysis

1) DNN model accuracy and compression ratio: Figure 8 shows the top-1 accuracy and compression ratio of five DNN models that use the patch-wise pruning-quantization algorithm based on different patch sizes. The results are compared with original DNN models. It can be seen that when the accuracy loss is within 3%, the effect of compression is significant. Specifically, LeNet5 obtains the highest compression ratio, which is 76.4× on average. Especially for 8-size patch, the compression ratio of LeNet5 is up to 91×. SqueezeNet gets the lowest compression ratio, which is still up to 11.3×.

2) Performance: Figure 9 presents the performance comparison of three different designs with different benchmarks in terms of the execution time. For all patch sizes given, PQ-PIM achieves performance speedup compared with CRMP and SRE. The performance improvement mainly comes from the pruning and quantization of weight patches, and the quantization of activation patches. Meanwhile, our architecture design perfectly supports patch-wise pruning-quantization algorithm, which contributes to the improvement. Compared with CRMP, for the 4-size patch, PQ-PIM achieves the highest performance improvement on average among different patch sizes, which is 1.28× speedup. Compared with SRE, PQ-PIM obtains the highest performance improvement for 32-size and the lowest performance improvement for 4-size, which are 1.74× and 1.62×, respectively. The reason for the different performance speedup roots in different compression ratios.

3) Energy: Figure 10 shows the energy consumption of the three different designs normalized to the baselines. It can be seen that for most patch sizes and DNN models given, PQ-PIM saves more energy compared to CRMP and SRE. The energy saving mainly comes from the access reduction of peripheral circuits, such as DAC and ADC. The access reduction results from the elimination of unimportant patches and the quantization of weakly important patches. Taking CRMP as a baseline, for the 4-size patch, PQ-PIM saves 62% energy on average. Compared with SRE, for the 32-size patch, PQ-PIM saves 30% energy on average. Note that since the OU-based single bit

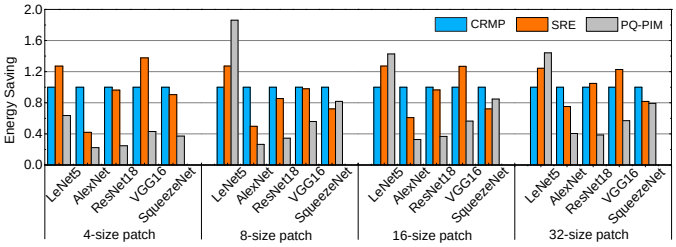

Fig. 10. Energy saving.

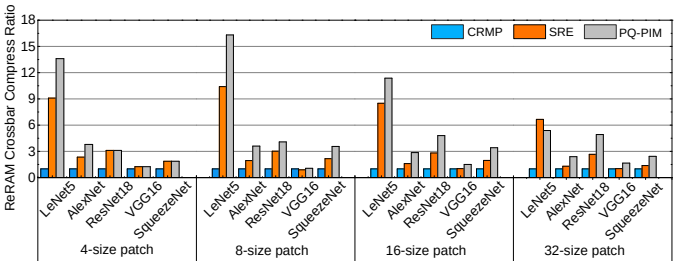

Fig. 11. ReRAM crossbar compression ratio.

ReRAM engine we proposed is more friendly for the deployment of large size DNN models, the energy consumption of small size models (e.g. LeNet5) is more than the baselines' for several patches.

4) ReRAM Resources: Figure 11 shows the comparison of the ReRAM crossbar compression ratio of these three ReRAM-based architectures normalized to CRMP and SRE. As shown, for most patch sizes and DNN models given, PQ-PIM achieves a higher compression ratio over CRMP and SRE. The ReRAM resources saving directly comes from the removal of unimportant weight patches and the quantization of weakly important weight patches. Among all patch sizes, PQ-PIM with 8-size patch achieves the highest compression ratio, which is $5.84\times$ on average compared to CRMP. Compared with SRE, PQ-PIM with 16-size patch obtains the highest compression ratio ($1.77\times$ on average).

5) Overhead: In PQ-PIM, the overhead mainly comes from the storage overhead related to the WPIT and APIT. Given a DNN model, assuming it has M weights and N activations, the size of weight patch and activation patch is n, each weight patch and activation patch need 1-bit binary flag. The storage overhead of WPIT and APIT are $(\frac{M}{n})$ and $(\frac{N}{n})$ bits. Figure 12 shows the storage overhead of WPIT and APIT. As shown, for a VGG16 model, when the patch size is 32, the storage overhead of WPIT is 0.44 MB, and the storage overhead of APIT is only 0.0055 MB. In addition, for the same DNN model, the larger the patch size is, the smaller the storage overhead of WPIT and APIT will be. Considering the performance, energy consumption and ReRAM resource utilization significant gain brought by PQ-PIM, the storage overhead of the tables is acceptable.

VI. Conclusion

In this paper, we propose an efficient pruning-quantization joint exploration framework for practical ReRAM-based DNN accelerator, named PQ-PIM, which consists of a patch-wise pruning-quantization algorithm and a configurable mixed OU-based single bit ReRAM DNN engine. The algorithm significantly compresses DNN models based on comprehensive patch importance analysis. The ReRAM DNN engine mainly consists of WPIT and APIT, which is designed to support the patch-wise pruning-quantization algorithm. It enables the algorithm with better performance improvement and energy saving. Experimental results show that PQ-PIM achieves up to $1.74\times$ speedup, 62% energy saving, and $5.84\times$ compression ratio of occupied ReRAM crossbars, compared with the state-of-the-art ReRAM-based DNN accelerators.

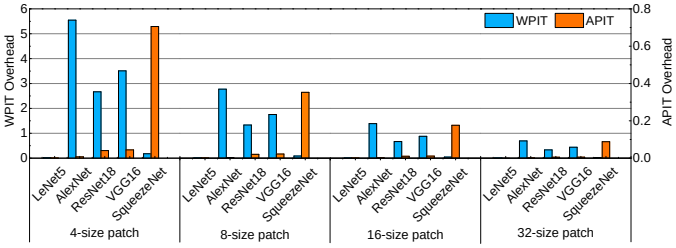

Fig. 12. Table overhead.

VII. Acknowledgements

The work described in this paper is partially supported by the grants from the National Science Foundation for Young Scientists of China (Grant No. 61902218), Shenzhen Science and Technology Foundation (JCYJ20170816093943197), and the National Natural Science Foundation of China (Grant No.92064008).

References

[1] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ISAC*, 2016.

[2] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015.

[3] H. Ji, L. Song, L. Jiang, H. H. Li, and Y. Chen, "Recom: An efficient resistive accelerator for compressed deep neural networks," in *DATE*, 2018.

[4] P. Wang, Y. Ji, C. Hong, Y. Lyu, D. Wang, and Y. Xie, "Snrram: an efficient sparse neural network computation architecture based on resistive random-access memory," in *DAC*, 2018.

[5] T.-H. Yang, H.-Y. Cheng, C.-L. Yang, I. Tseng, H.-W. Hu, H.-S. Chang, H.-P. Li *et al.*, "Sparse reram engine: joint exploration of activation and weight sparsity in compressed neural networks," in *ISCA*, 2019.

[6] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," in *DATE*, 2018.

[7] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[8] Z. Zhu, H. Sun, Y. Lin, G. Dai, L. Xia, S. Han, Y. Wang, and H. Yang, "A configurable multi-precision cnn computing framework based on single bit rram," in *DAC*, 2019.

[9] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *ASPLOS*, 2019.

[10] X. Ma, G. Yuan, S. Lin, C. Ding, F. Yu, T. Liu, W. Wen, X. Chen, and Y. Wang, "Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient dnn implementation," in *ASP-DAC*, 2020.

[11] G. Yuan, X. Ma, C. Ding, S. Lin, T. Zhang, Z. S. Jalali, Y. Zhao, L. Jiang, S. Soundarajan, and Y. Wang, "An ultra-efficient memristor-based dnn framework with structured weight pruning and quantization using admm," in *ISLPED*.

[12] M.-Y. Lin, H.-Y. Cheng, W.-T. Lin, T.-H. Yang, I.-C. Tseng, C.-L. Yang, H.-W. Hu, H.-S. Chang, H.-P. Li, and M.-F. Chang, "Dl-rsim: A simulation framework to enable reliable reram-based accelerators for deep learning," in *ICCAD*, 2018.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, 1995.

[14] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[15] L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "Mnsim: Simulation platform for memristor-based neuromorphic computing system," *TCAD*, vol. 37, no. 5, pp. 1009–1022, 2017.