DE-C3: Dynamic Energy-Aware Compression for Computing-In-Memory-Based Convolutional Neural Network Acceleration

Guan-Wei Wu[1], Cheng-Yang Chang[2], and An-Yeu (Andy) Wu[1,2]
[1]*Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan*
[2]*Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan*
b08901019@ntu.edu.tw, kevin@access.ee.ntu.edu.tw, andywu@ntu.edu.tw

***Abstract*—Convolutional neural networks (CNNs) are leveraged in many applications, such as image classification and natural language processing (NLP) tasks. However, the hardware implementation of CNNs not only occupies a considerable amount of memory space but also incurs high computational demands, which can lead to significant energy cost for data transfer. Therefore, an emerging computing-in-memory (CIM)-based architecture has been proposed to alleviate the energy bottleneck of CNNs. Various model compression methods have recently been studied to enhance energy efficiency of the CIM-based accelerator, including quantization and pruning. However, these works are separately discussed with the static inference scenario. In this paper, the proposed DE-C3 can dynamically and jointly design the compression strategy. Besides, it adopts trainable energy-aware thresholds for both quantization and pruning scenarios. Experimental results based on the CIFAR-10 dataset show that our DE-C3 achieves up to 3.4× the energy reduction compared to state-of-the-art works.**

***Index Terms*—Convolutional neural network, computing-in-memory, energy-aware training, dynamic inference**

I. Introduction

Convolutional neural networks (CNNs) [1] are extensively utilized and researched frameworks that enhance the performance in tasks such as image classification, natural language processing (NLP), and video analysis. However, the large number of model weights results in a significant requirement for memory storage and substantial energy consumption due to frequent data transfers between the processing unit and memory in traditional von Neumann architecture. This bottleneck effect in traditional hardware architecture can limit the deployability of CNNs.

Computing-in-memory (CIM) [2] [3] combines the characteristics of memory and the processing unit, reducing the need for frequent data transportation between them and alleviating the memory-wall problem. Moreover, the majority of operations in CNNs involve vector-matrix multiplications (VMMs) between inputs and weights. These VMMs can be efficiently computed in parallel in the analog domain within the CIM architecture. Consequently, CIM offers a practical approach to accelerate CNN inference while reducing energy consumption.

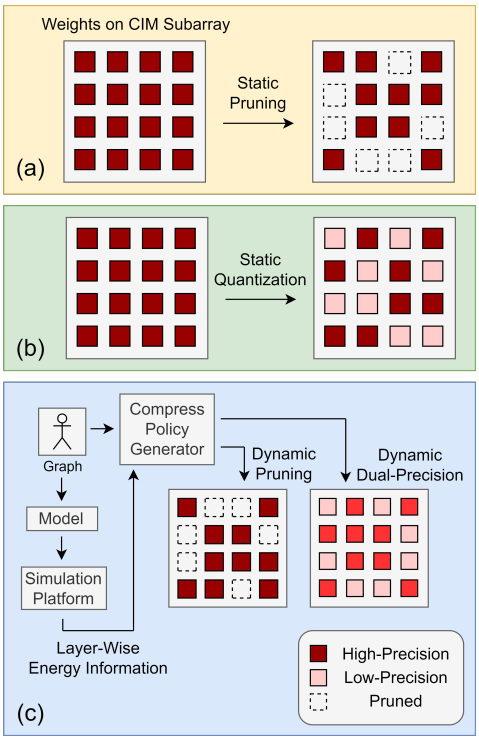

Fig. 1. (a), (b) Previous works vs. (c) proposed DE-C3.

However, the efficiency of CIM architecture is hindered by the significant amount of interface transformation between the analog and digital domains performed by analog-to-digital converters (ADCs). To address this issue, model compression methods tailored for CIM architecture have been extensively studied. These methods aim to reduce the size of parameters and model complexity, and several approaches [4] [5] [6] have been proposed to this end, including quantization and pruning. Some of these compression algorithms are implemented based on operation units (OUs), which are smaller crossbar regions within the memory. The utilization of OU-based methods enables finer-grained computation and provides more precise strategies for compression.

This work was supported by the National Science and Technology Council of Taiwan under Grants of MOST 111-2218-E-002-018-MBK and NSTC 112-2218-E-002-025-MBK. This research was also supported by NOVATEK Fellowship.

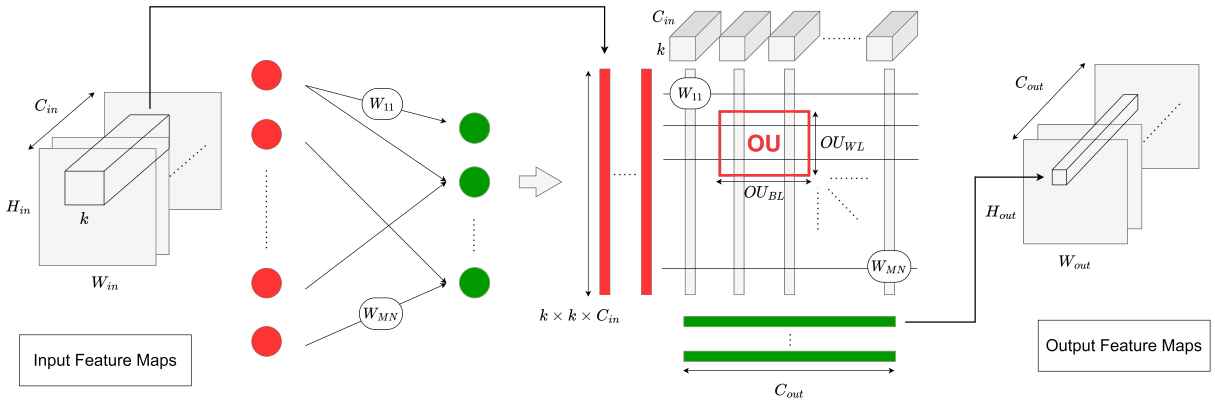

Fig. 2. The typical computing-in-memory (CIM) architecture with the weight mapping configuration. The red rectangle on the subarray indicates a 2×2 OU as an example.

Although previous works have made progress in improving the trade-off between accuracy and energy cost to some extent, there are still several important issues worth investigating:

- **Applying static compression strategies with fixed model weights during the inference phase:** Existing compression methods primarily focus on training strategies that can be efficiently deployed on most training data. These strategies are then applied uniformly to the testing data during the inference phase. However, these models lack the ability to identify input-dependent redundancy and detect critical input components.
- **Considering quantization and pruning separately without considering energy information:** Some prior works have employed either mixed-precision quantization or structured pruning as independent compression techniques and disregard the layer-wise energy cost. However, both the receptive fields and energy consumption can significantly vary across different CNN layers. Therefore, it is possible that these approaches have resulted in sub-optimal solutions.

In this paper, we propose a dynamic energy-aware compression framework for CNN models within a CIM architecture, called DE-C3, to address the mentioned issues. The core concept is illustrated in Fig. 1. Our contributions can be summarized as follows:

- **First dynamic inference framework for CIM architecture:** Unlike previous works that apply statically trained models during the inference phase, DE-C3 dynamically adjusts the compression policy according to the input characteristics. To achieve this, the algorithm is designed to form dual processing paths. One path calculates the most significant bits (MSBs) of weight multiplications, while the other path decides the least significant bits (LSBs) of weights, which are conditionally merged later.
- **Consideration of CIM energy consumption with hybrid compression training strategy:** Our proposed DE-C3 takes CIM energy consumption into account through a hybrid compression training strategy. The compression strategy is determined based on the consequent energy savings. Moreover, the strategy can incorporate both the dual-precision mode and the pruning mode in different layers, thereby enhancing the effectiveness of the reduction policy.

We validate the effectiveness of our proposed DE-C3 using the CIFAR-10 dataset and two CNN models: ResNet-20 [7] and VGG-8 [8]. Among previous works, MARS [4] and T-EAP [5] have proposed similar OU-based compression algorithms. Although both methods demonstrate favorable performance, MARS neglects the energy information, while T-EAP is designed for a static inference scenario. Our proposed DE-C3 exhibits energy efficiency, achieving a reduction of up to 2.3× and 3.4× in total energy cost compared to previous works for ResNet-20 and VGG-8, respectively.

II. Background

A. Computing-in-memory (CIM)

The CIM architecture, shown in Fig. 2, is structured around the memory subarray. In this architecture, digital input signals are received and processed. The input signal is treated as a vector and converted into voltages on the word lines (WLs) of the memory subarray using digital-to-analog converters (DACs). Additionally, the matrix weight components are stored on the memory subarray, represented as the conductance of memory cells. The parallel computation fashion is achieved by measuring the current values on the bit lines (BLs). These currents are the accumulated result of the multiplication between inputs and weights at each intersection node. Subsequently, the output currents are directed to multiplexers (MUXs) and transformed back into digital signals using analog-to-digital converters (ADCs) for further shift-and-add processing.

B. Operation unit (OU)

Most previous works on CIM implementation [9] [10] [11] focused on designing algorithms with coarse granularity, considering the entire memory subarray. Nevertheless, several recent studies [12] [13] [14] [15] have demonstrated that the

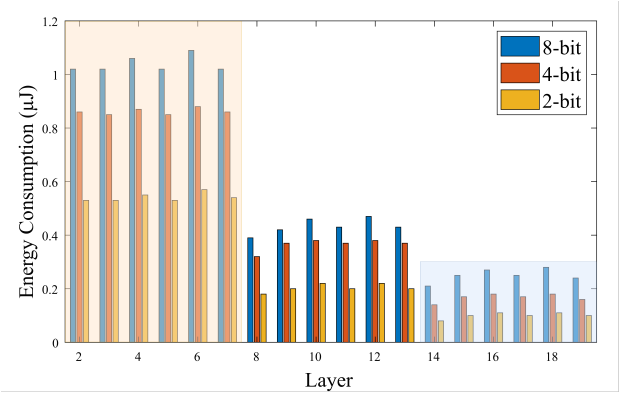

Fig. 3. Layer-wise energy consumption with different bit widths for ResNet-20. The orange part indicates the shallower layer block with a higher energy cost, while the blue part indicates the deeper layer block with a lower energy cost.

OU-based implementation of CIM is more practical due to hardware constraints on OU size and the ability to exploit finer-grained levels of sparsity. The number of WLs is set as $k \times k \times C_{in}$, where k is the kernel size and C_{in} is the input channel length. The weight W is mapped to the conductance of cells on C_{out} BLs, where C_{out} is the output channel length. The entire weight matrix can be further separated into many small crossbar regions called operation units (OUs), as exemplified in Fig. 2. Although this may result in reduced throughput, it helps alleviate the high cost of ADCs and provides a $M \times N$ finer granularity of the compression space. The number of required OUs is also defined as $M \times N$, where:

$$M = \frac{k \times k \times C_{in}}{OU_{WL}}, N = \frac{C_{out}}{OU_{BL}} \tag{1}$$

C. Dynamic inference

Most studies on CNN models are devoted to achieving competitive prediction accuracy compared to state-of-the-art performance. However, the enormous energy consumption associated with these models poses challenges for deploying them in resource-limited scenarios. Therefore, various approaches supporting dynamic inference have been recently proposed. For instance, the BlockDrop [16] technique in residual networks enables the dynamic selection of executed layers during the inference phase.

Nevertheless, there is currently no solution that meets the needs of dynamic OU-based compression within the CIM architecture. Previous works primarily adopted static inference methods and were prone to overlook input-dependent information at the inference level.

Many dynamic inference techniques involve executing only a portion of the network by designing several processing paths between components. Precision Gating (PG) [17] is one such dynamic approach that achieves dual-precision quantization. Our proposed DE-C3 builds upon this core idea and designs a similar computational flow. However, our work distinguishs itself by focusing not only on OU-based dynamic behavior related to the CIM architecture but also on generating a dynamic compression policy while considering the consequent energy savings from a hardware perspective.

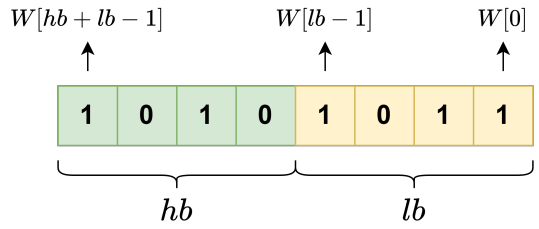

Fig. 4. E_H stands for computing with $W[(hb + lb - 1) : 0]$, and E_L represents computing with $W[(hb + lb - 1) : 0] - W[(lb - 1) : 0]$.

III. PROPOSED DE-C3 FRAMEWORK

A. Off-line energy analysis of DE-C3

Since the DE-C3 model incorporates energy information into the compression policy, it is necessary to analyze the layer-wise energy in the off-line phase. We utilize DNN+NeuroSim [18] to evaluate the energy cost of the CIM-based accelerator for CNN inference. Taking ResNet-20 as an example, Fig. 3 illustrates the layer-wise energy analysis of the model, demonstrating variations in energy cost among different layers and bit solutions. Since the energy consumption is not linearly related to the bit width, we consider energy of two different weight precisions, ${E_H}^{\ell}$ and ${E_L}^{\ell}$, for each layer ℓ. Specifically, the energy savings achieved by compressing one OU in layer ℓ of DE-C3 can be represented as:

$$({E_H}^{\ell} - {E_L}^{\ell}) \times \frac{1}{M^{\ell} \times N^{\ell}} \tag{2}$$

This value tends to become more significant in shallower layers due to the smaller values of ${C_{in}}^{\ell} \times {C_{out}}^{\ell}$ compared to deeper layers. Additionally, the larger size of the input feature map ${H_{in}}^{\ell} \times {W_{in}}^{\ell}$ indicates that ADCs will be activated more frequently in shallower layers, resulting in higher energy consumption, which further justifies the need for compression.

Fig. 4 presents an example where each layer in DE-C3 is assigned a bit width pair (hb^{ℓ}, lb^{ℓ}), manually decided using the layer arranger. This pair determines the compression method for layer ℓ, where hb^{ℓ} stands for the bit width of the MSBs, and lb^{ℓ} represents the bit width of the LSBs.

Within each dual-precision compression layer of DE-C3, ${E_h}^{\ell}$ stands for the energy cost per OU for high precision $(hb^{\ell} + lb^{\ell})$ computation in the ℓ^{th} layer, which means,

$${E_h}^{\ell} = {E_H}^{\ell} \times \frac{1}{M^{\ell} \times N^{\ell}} \tag{3}$$

while ${E_l}^{\ell}$ represents the energy cost for low precision (hb^{ℓ}) computation in the ℓ^{th} layer.

B. DE-C3 processing flow

Fig. 5 shows the overall operation flow of DE-C3. During the training phase, the RGB three-channel input graph is

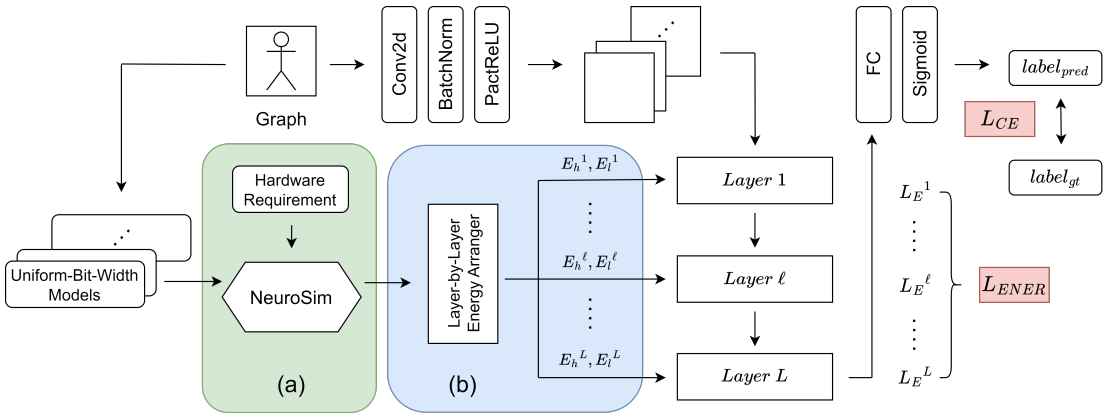

Fig. 5. The overall flow chart of the DE-C3 method, including (a) generating the energy information and (b) arranging layer-wise bit width pairs and corresponding energy.

passed through a regular CNN layer to extract the initial input feature map for subsequent DE-C3 layers. The energy consumption values are sent to an arranger, which manually determines the layer-wise compression method. The arranger provides flexibility to the DE-C3 framework, allowing for a hybrid compression scenario. In previous works, weight quantization and model pruning were considered as separate compression methods. However, within the dual-precision fashion of DE-C3, weight quantization can be seen as bit-wise pruning or dual-precision compressing. Similarly, model pruning can be regarded as a special case where lb^ℓ equals 0 bit. Therefore, the arranger can freely set different combinations, enlarging the search space for the accuracy-energy trade-off.

The input information for each layer includes the input feature map I^ℓ, the energy pair $(E_h{}^\ell, E_l{}^\ell)$, and the number of required OUs $M^\ell \times N^\ell$ for each DE-C3 layer ℓ. Notably, the size of Δ^ℓ corresponds to $M^\ell \times N^\ell$, signifying that DE-C3 is characterized as an OU-based compression approach. In each layer ℓ, the energy consumption is estimated by a linear combination of $E_h{}^\ell$ and $E_l{}^\ell$, representing the level of compression. These terms are added together and form part of the objective function L_{ENER}, aiming to minimize the energy cost. The training process also targets the cross-entropy loss L_{CE} between predicted and ground-truth labels to achieve high accuracy. The joint training scenario is formulated with the loss function $L_{CE}+\alpha\times L_{ENER}$, where α is a hyperparameter controlling the inclination of trade-off between accuracy and energy. Specifically, setting the value of α higher means to prioritize the energy reduction, and thus may potentially impact the accuracy.

Referring to the dynamic executions shown in Fig. 6(a) and Algorithm 1, the weight matrix W is separated into two parts: $W[(hb+lb-1):0]-W[(lb-1):0]$ stands for the MSB part, and $W[(lb-1):0]$ represents the LSB part. The MSB part of weights is activated first to perform convolutional operations, and the resulting partial sums of each OU serve as an indicator of importance. A trainable OU-wise threshold set, forming unit step functions shifted by Δ^ℓ, is leveraged to filter the essential partial sums. An OU-wise mask, storing values of

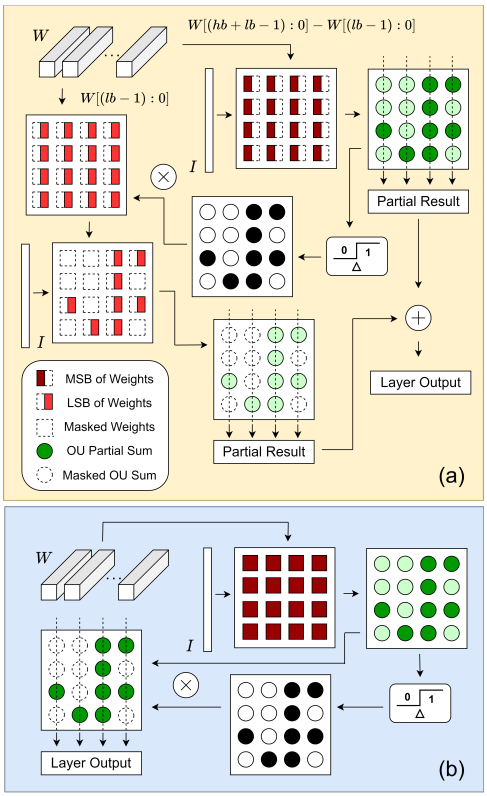

Fig. 6. Detailed DE-C3 operations for (a) dynamic dual-precision mode and (b) dynamic pruning mode.

0 and 1, records the results of partial sum importance and is applied to the LSB part. Compression occurs in the LSB part, as only those positions corresponding to the mask value of 1 are computed. After accumulating partial sums, the results of both parts are shifted and combined to obtain the output feature map of the layer.

DE-C3 serves as a unified view of quantization and pruning. When lb^ℓ is set to 0, as illustrated in Fig. 6(b), DE-C3 adopts the pruning mode. If lb^ℓ is 0 due to the 0-bit pruning condition, the LSB part is all-0 regardless of the mask value.

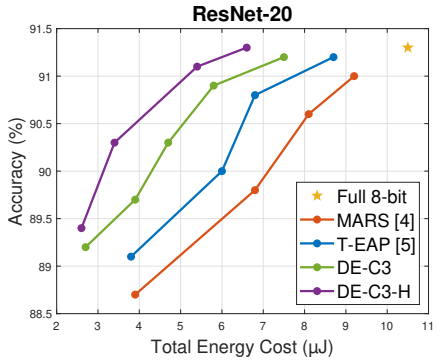

Fig. 7. Pareto fronts of accuracy-energy trade-off for MARS, T-EAP, and DE-C3 methods on ResNet-20 with CIFAR-10.

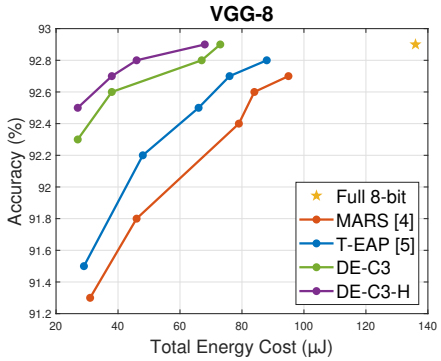

Fig. 8. Pareto fronts of accuracy-energy trade-off for MARS, T-EAP, and DE-C3 methods on VGG-8 with CIFAR-10.

Thus, the mask is applied back to the partial sum result of the MSB part to achieve the pruning fashion. Although the functionality in reducing energy consumption may not be as effective, the pruning mode helps eliminate irrelevant features more completely than the dual-precision mode.

IV. EXPERIMENT AND RESULTS

A. Experimental setting

This paper applies the CIFAR-10 image classification dataset to two CNN models, ResNet-20 and VGG-8. For ResNet-20, the hyperparameter set in the software part includes a learning rate of 0.1, a batch size of 128, and α values searched within the range of [1e-7, 1e-8, 1e-9]. The models are trained for 100 epochs using the SGD optimizer, with the learning rate decreasing by 0.1 at the 50^{th} and 75^{th} epochs.

Furthermore, the DNN+NeuroSim is utilized to simulate and estimate the hardware behavior of the experiments, providing layer-wise energy consumption data. The corresponding hardware conditions include a subarray OU crossbar size of 16, ADC resolution being adjusted from 4 to 6 bits, and a technology node of 22nm ReRAM. This paper considers bit lengths of 8-bit, 6-bit, 4-bit, 2-bit, 1-bit, and 0-bit (pruned).

B. Accuracy-energy trade-off

We compare the performance of our proposed DE-C3 with previous works, MARS [4] and T-EAP [5], as well as the uncompressed 8-bit model, which serves as the lower bound. We evaluate the performance based on the Pareto front results of these methods as we are interesting in the trade-off between accuracy and energy consumption. Fig. 7 presents the experimental results for ResNet-20, obtained from models trained using the MARS, T-EAP, and DE-C3 methods.

To provide more details, each data point on the MARS, T-EAP, and DE-C3 lines stands for models trained with different bit width pairs denoted by (hb^ℓ, lb^ℓ). The combination pairs include (6, 2), (2, 6), (4, 4), (4, 2), (2, 4), (2, 2) and (1, 1). Additionally, the DE-C3-H line represents for the implementation of hybrid compression in each layer of the models using various combinations of bit width pairs. Taking our experiment for ResNet-20 as an example, it consists of three combination pairs since the model has 18 layers, excluding the first and last layers. Every six layers share the same architecture, forming a layer block targeted at different bit widths to adopt hybrid compression. Our experiments for DE-C3-H include combinations of the mentioned (hb^ℓ, lb^ℓ) pairs

Algorithm 1 Proposed DE-C3

Input: Input Feature Map I^ℓ; Energy $({E_h}^\ell, {E_l}^\ell)$; Number of OUs $M^\ell \times N^\ell$, where $M = \frac{k \times k \times C_{in}}{OU_{WL}}$ and $N = \frac{C_{out}}{OU_{BL}}$; for $\ell = 1, 2, ..., L$

for $\ell = 1$ to L **do**
  ${O_{hb}}^\ell = Conv2d(I^\ell, {W_{hb}}^\ell)$
  for $i = 1$ to M_ℓ **do**
    for $j = 1$ to N_ℓ **do**
      if $Conv2d((I^\ell)_{ij}, ({W_{hb}}^\ell)_{ij}) \geq ({\Delta^\ell}_{ij})$ **then**
        $(Mask^\ell)_{ij} = 1$
      else
        $(Mask^\ell)_{ij} = 0$
      end if
    end for
  end for
  ${O_{lb}}^\ell = Conv2d(I^\ell, Mask^\ell \times {W_{lb}}^\ell)$
  $O^\ell = {O_{hb}}^\ell + {O_{lb}}^\ell$
  $spar^\ell = \frac{Mask^\ell[Mask^\ell < 1].numel()}{Mask^\ell.numel()}$
  ${L_E}^\ell = {E_l}^\ell + ({E_h}^\ell - {E_l}^\ell) \times (1 - spar^\ell)$
  $L_{NENR} = L_{NENR} + {L_E}^\ell$
end for

$label_{pred} = Sigmoid(FC(O^L))$
$L_{CE} = CrossEntropy(label_{pred}, label_{gt})$
$L_{total} = L_{CE} + \alpha \times L_{ENER}$
$L_{total}.backward()$

along with pairs where lb equals 0 for the pruning scenario, such as (4, 0) and (2, 0).

We observe that T-EAP exhibits a more favorable accuracy-energy trade-off compared to MARS, as it has modified the pruning approaches and focused on energy-intensive elements. Furthermore, our DE-C3 can shift the Pareto front to achieve an even better accuracy-energy trade-off level by considering not only energy information but also adopting dynamic inferences. Besides, the DE-C3-H results demonstrate that different bit width pairs are chosen for different CNN layers in the DE-C3 implementation. The arranger enhances the flexibility of compression and allows for a finer compression decision, pushing the Pareto front to a more upper-left position. The overall energy consumption reduction achieved by DE-C3 compared to the other two methods is up to 2.3× for ResNet-20. A similar trend can also be observed for models trained on VGG-8, showing a 3.4× reduction in energy consumption while achieving similar accuracy, as demonstrated in Fig. 8.

V. CONCLUSION

This work proposes DE-C3, a dynamic and energy-aware compression approach for CIM-based CNN models. DE-C3 leverages layer-wise energy consumption data generated by the DNN+NeuroSim hardware simulator as part of the training objective. It employs dual processing paths and weight masks to enable dynamic inference, thereby enhancing the overall compression effectiveness of the model systems. As a result, DE-C3 improves the accuracy-energy trade-off and achieves the energy cost reduction of up to 2.3× for ResNet-20 and 3.4× for VGG-8.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE circuits and systems magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[3] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*. Springer, 2022, pp. 171–243.

[4] S.-H. Sie, J.-L. Lee, Y.-R. Chen, Z.-W. Yeh, Z. Li, C.-C. Lu, C.-C. Hsieh, M.-F. Chang, and K.-T. Tang, "Mars: Multimacro architecture sram cim-based accelerator with co-designed compressed neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 5, pp. 1550–1562, 2021.

[5] C.-Y. Chang, Y.-C. Chuang, K.-C. Chou, and A.-Y. Wu, "T-eap: Trainable energy-aware pruning for nvm-based computing-in-memory architecture," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 78–81.

[6] C.-Y. Chang, K.-C. Chou, Y.-C. Chuang, and A.-Y. Wu, "E-upq: Energy-aware unified pruning-quantization framework for cim architecture," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 21–32, 2023.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] J. Lin, Z. Zhu, Y. Wang, and Y. Xie, "Learning the sparsity for reram: Mapping and pruning sparse neural network for reram based accelerator," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 639–644.

[10] C. Chu, Y. Wang, Y. Zhao, X. Ma, S. Ye, Y. Hong, X. Liang, Y. Han, and L. Jiang, "Pim-prune: Fine-grain dcnn pruning for crossbar-based process-in-memory architecture," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[11] B. Li, S. Qu, and Y. Wang, "An automated quantization framework for high-utilization rram-based pim," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 3, pp. 583–596, 2021.

[12] T.-H. Yang, H.-Y. Cheng, C.-L. Yang, I.-C. Tseng, H.-W. Hu, H.-S. Chang, and H.-P. Li, "Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 236–249.

[13] S. Yang, W. Chen, X. Zhang, S. He, Y. Yin, and X.-H. Sun, "Auto-prune: Automated dnn pruning and mapping for reram-based accelerator," in *Proceedings of the ACM International Conference on Supercomputing*, 2021, pp. 304–315.

[14] S. Yu, L. Zhang, J. Wang, J. Yue, Z. Yuan, X. Li, H. Yang, and Y. Liu, "High area/energy efficiency rram cnn accelerator with pattern-pruning-based weight mapping scheme," in *2021 IEEE 10th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2021, pp. 1–6.

[15] F. Liu, Z. Wang, Y. Chen, Z. He, T. Yang, X. Liang, and L. Jiang, "Sobs-x: Squeeze-out bit sparsity for reram-crossbar-based neural network accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 204–217, 2022.

[16] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8817–8826.

[17] Y. Zhang, R. Zhao, W. Hua, N. Xu, G. E. Suh, and Z. Zhang, "Precision gating: Improving neural network efficiency with dynamic dual-precision activations," *arXiv preprint arXiv:2002.07136*, 2020.

[18] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "Dnn+ neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *2019 IEEE international electron devices meeting (IEDM)*. IEEE, 2019, pp. 32–5.