CIM-Aware Quantization for Energy Efficient Generative AI Models

ABSTRACT

Large language models (LLMs) with generative capabilities have showcased outstanding performance across a diverse applications. However, deploying these models on resource-constrained edge and mobile devices during inference mode is very challenging due to their massive computations. In this paper, we proposed a compute-in-memory (CIM) aware post-training quantization and sparsity technique to achieve superior energy efficiency compare to the traditional CMOS architecture. In CIM, multiplying to "W" bit requires less energy that "1". Therefore, we quantized the LLMs to the desired numbers where they have less ones in their binary representations that zero. We used a 2bit/cell resistive CIM test chip to evaluate out method. Since in the 2bit/cell RCIM $E_{00} < E_{01} < E_{10} < E_{11}$, we demonstrated that our method achieved 2.6x less energy than the baseline method which eventually enabling running LLMs on edge devices.

1. INTRODUCTION AND BACKGROUND

The extensive integration of mobile devices has brought about a transformative change in how individuals engage with technology. Mobile systems are progressively more proficient in executing advanced artificial intelligence (AI) tasks, encompassing generative AI models. These generative AI models, which comprise substantial language models, have facilitated impressive progress across diverse fields like computer vision [8], natural language processing [10,12], and healthcare [11]. Yet, effectively implementing and operating these demanding generative AI models on mobile platforms give rise to notable difficulties. Mobile and edge devices are resource constrained in terms of memory and power which is very challenging to run AI models on them [5].

Large language models (LLMs) are known for their massive size, often comprising tens of billions of parameters. The number of parameters in LLMs natural language processing (NLP) models are growing exponentially as shown in Fig. 1. These LLMs necessitate substantial computational resources, memory, and energy to operate seamlessly. To illustrate further, consider a scenario where a mobile application aims to provide real-time, contextually relevant suggestions for users' text input. Achieving this using generative AI entails a delicate balance between intricate linguistic patterns and instant response times. The challenge magnifies when dealing with LLMs due to their complex architecture and extensive parameter count. Deploying LLMs on mobile devices requires careful orchestration of computational resources to

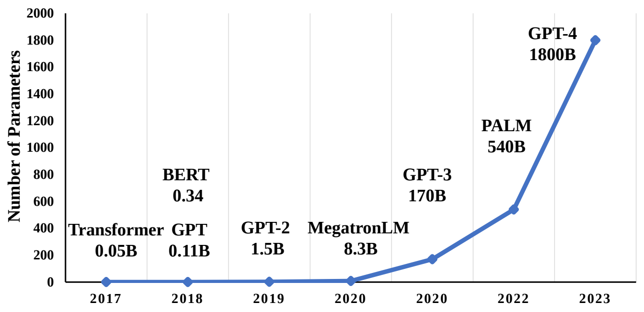

Figure 1: NLP model size is increasing exponentially from Transformers with less than a billion (B) parameters to GPT-4 with estimated 1800B parameters.

ensure smooth functioning without compromising the user experience. In essence, while the potential of generative AI models on mobile platforms is enormous, the challenge of effectively accommodating their complexity within the constraints of mobile hardware is akin to fitting a grand opera onto a pocket-sized stage. Finding innovative ways to optimize these models, streamline their deployment, and manage their resource demands becomes a crucial endeavor to unlock their benefits on the move.

Quantization, in the context of deploying generative AI models on mobile devices, emerges as a pivotal technique to mitigate the challenges posed by their resource-intensive nature. Quantization essentially involves reducing the precision of numerical values in the model, thereby decreasing memory and computation requirements [1]. Two primary quantization approaches are Quantize Aware Training (QAT) and Post-Training Quantization (PTQ) [6]. QAT necessitates adjusting the model's weights during retraining to regain accuracy after quantization. In contrast, PTQ carries out quantization after training. Although QAT often yields improved accuracy, its feasibility for LLMs is limited due to the high retraining costs and potential lack of access to training data and infrastructure [9]. Consequently, PTQ is a better candidate for quantizing LLMs since it involves quantizing an already-trained model without retraining it. Thus, the majority of research on quantizing LLMs has centered around PTQ techniques [1,2,4,9,14,18]. SqueezeLLM [9] showed that memory bandwidth is the main bottleneck for generative inference with LLMs and proposed a PTQ method to quantize LLMs to low-bit precision through sensitivity based non-uniform quantization. PTQ4ViT [17] proposed a twin uniform PTQ to quantize vision transformer to 8 bit precision. Smoothquant [14] developed a smoothing strategy to first smooth the activation outliers and then quantize the LLMs with billions of parameters to enable 8-bit weight, 8-bit activation (W8A8) quantization during inference. Although achieved a negligible negligible loss in accuracy, these quantization methods are not designed to fully take advantage of novel architectures with emerging technologies like compute-in-memory (CIM). In contrast to the Von-Neumann architecture, computation is performed inside memory in CIM architecture. This characteristic avoids huge power dissipation incurred by massive data transfer between the PEs and memory [7]. BitS-Net [7] proposed a CIM friendly bit-level sparsity method which sparsified the traditional deep learning models in the bit-level. However this method cannot be applied to LLMs since BitS-Net quantized and sparsified the networks during training.

In this paper, we developed a CIM-aware PTQ method which quntized the LLMs by removing the ones in the bit representation of weight values with no retraining. The motivation is that multiplying to 0 requires 18x less energy than ones. We demonstrated that by using our proposed method can achieve 2.6x energy efficiency compared to the uniform PTG technique.

2. METHOD

Here, we introduce our proposed CIM-aware compression technique for BERT-like model quantization. We quantize the network while sparsify the weight values in bit-level during a PTQ scheme.

2.1 Advantages of Using CIM Architecture

The adoption of CIM architecture holds immense promise for achieving energy-efficient AI computations, particularly in the realm of large language models (LLMs). CIM architecture integrates processing elements directly within the memory units, minimizing data movement between storage and processing units. This characteristic significantly reduces the energy overhead associated with data transfer, which is a major bottleneck in traditional architectures. For LLMs, which are known for their resource-intensive nature, CIM architecture can lead to remarkable energy savings by enabling localized processing of model operations directly within the memory, avoiding the energy-intensive process of fetching large amounts of data back and forth. This approach aligns with the intricacies of LLM computations, such as attention mechanisms and fully connected feed forward layers by capitalizing on parallelism and reducing the overall computational load. Consequently, the synergy between CIM architecture and LL Ms has the potential to revolutionize energy efficiency in AI, enabling more practical deployment of advanced language models on resource-constrained mobile devices while minimizing environmental impact.

In this study, a 2-bit encoding resistive CIM (RCIM) architecture is utilized [15] for evaluation of our proposed quantization method. The high-level architecture is illustrated in Fig. 2. Employing RRAM cells encoded with 2-bits (11, 10, 01, and 00) alongside ADC-based readout circuits, the energy consumed per bit during CIM is measured as 0.83, 0.47, 0.28, and 0.15 pJ/bit, respectively. The actual measurements from the 2bit/cell RCIM test chip shows that multiplying to 00 requires less energy that 11 and this our motivation to develop a bi-level sparsity and PTQ technique for compressing LLMs.

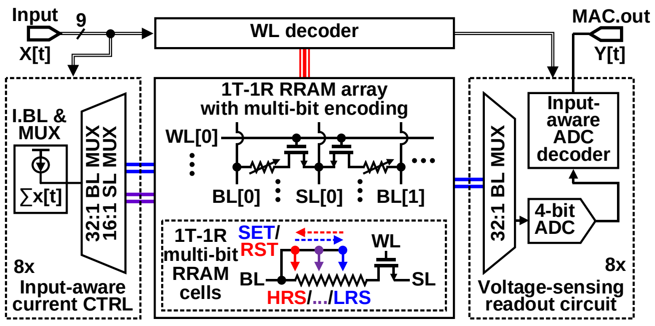

Figure 2: Architecture of voltage-sensing multi-bit RCIM architecture. The figure is adopted from [7].

2.2 CIM-Aware Quantization Scheme

Quantization reduces the precision of numerical values in LLMs, decreasing memory and computational requirements. When coupled with CIM architecture, which integrates processing directly within memory units, the benefits multiply. However, the current quantization methods [] are not designed to take advantage of CIM architecture. Instead of quantizing the network uniformly, the network is quantized to the desired coefficient set which has the characteristics that numbers has more zeros than 1 in their binary representation. This is a simple yet effective method to leverage the advantage of CIM architecture for quantized LLMs.

We define the quantization process as shown in Eq. 1. Δ is the scaling factor, $\Pi_{Q(.)}$ is the method of quantization, b is the bit-precision and $\lfloor . \rceil$ shows the rounding process. Selecting the optimized scaling factor is very important in PTQ methods. Uniform quantization is a process in which the range of continuous values is divided into a fixed number of equally spaced levels, and each input value is mapped to the nearest level. The quantization process for uniform quantization is shown in Eq. 2. However, this method quantize the network to the numbers that is not CIM-friendly for highly energy-efficient computing. Inspired by [7], we proposed a CIM-friendly post-training quantization method that quantize the network non-uniformly to the desired INT8 numbers. The desired numbers are defined as set of numbers with no 11 at their binary representation like 8. During PTQ and in the forward path, the weights are quantized based on Eq. 3 by dividing the weights in each layer (l) by a scaling factor (Δ) to make the weights in the range of [-1, 1]. The scaled weights are quantized by computing the minimum distance between the weights and the coefficient set (c_i) as illustrated in Eq. 3).

$C = \pm\{0, 0.3438, 0.3750, 0.4063, 0.5, 0.6250, 0.6563, 1.0\}$ is an example of a desired coefficient set.

$$W_q = \Delta \Pi_{Q(1,b)} \lfloor \frac{W}{\Delta}, 1 \rceil \tag{1}$$

$$\Pi_{Q(1,b)} = clip(\lfloor \frac{W}{\Delta} \rceil;\ 2^b - 1) \tag{2}$$

$$\Pi_{Q(1,b)} = argmin_{c_i \in C}|c_i - |\frac{W_l}{\Delta}|| \quad (3)$$

2.3 Gen AI quantization scheme

In this work, we quantize Bidirectional Encoder Representations from Transformers (BERT) model [3] which consists of 12 layers of encoder. BERT is a type of transformer-based model, and it's designed to understand the context of words in a sentence by considering the words that come before and after them [3]. The illustration of encoder containing multi-head self-atention and feed forward is illustrated in Fig. 3. $\otimes$ is an element-wise addition. We apply our proposed PTQ quantization method on all the weights and activation for INT8 precision.

3. RESULTS

Here, we evaluate the proposed quantization techniques for the BERT model on GLUE RTE task.

3.1 Experimental Setup

We used Nvidia GTX 1080 Ti GPUs. Only one GPU is used during PTQ quantization. we utilized uniform 8-bit quantization with symmetric weights, asymmetric activation and the static activation range [1] as the baseline. We used the General Language Understanding Evaluation (GLUE) [13] which is a collection of resources for analyzing natural language processing (NLP) tasks. Specifically, we used the Recognizing Textual Entailment (RTE) dataset where constructed based on news and Wikipedia text. To have a fully quantized model, we quantized both weights and activation of the BERT model to desired coefficient set that can be implemented in an on-chip system. The got the 2% error compared to the INT8 uniform quantization. This shows that we can quantized the BERST model with a negligible accuracy loss.

In this part, we present the outcomes of our RCIM implementation involving the our proposed technique and the baseline uniform PTQ. The energy estimation encompasses various components like the RRAM array, the ADC, the controller, and peripheral circuits, excluding the voltage reference (VREF) generator. We've omitted the VREF generator

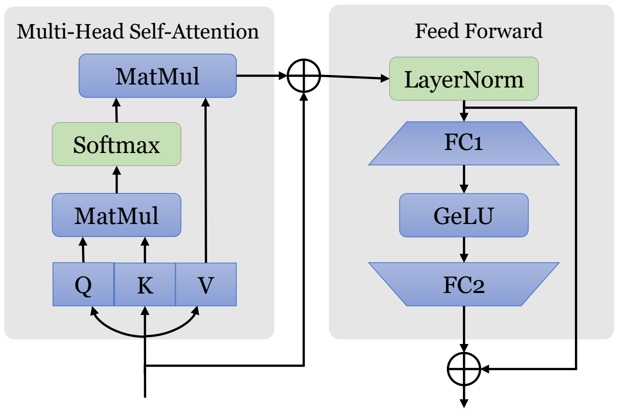

Figure 3: A schematic illustration of the encoder layer in BERT. The blue blocks are quantized to INT8.

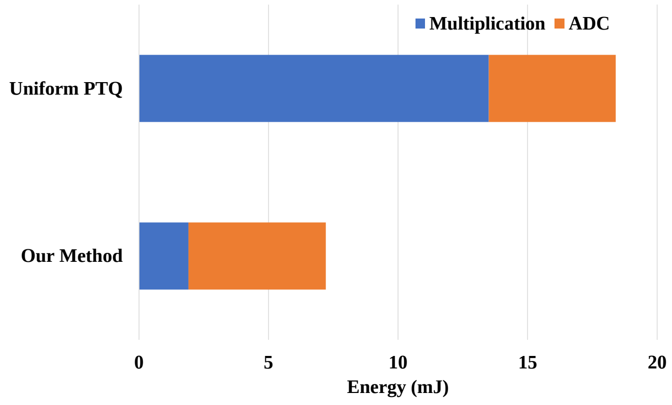

Figure 4: Energy breakdown for multiplication and ADC for BERT model and RCIM test chip for our method and the uniform PTQ during inference.

from this analysis because a single instance of it serves the entire RRAM macro, and its power consumption becomes negligible when scaling up the RRAM macro size. Our energy estimations were conducted using measurements from the RCIM hardware. However, it's worth noting that the RRAM macro's dimensions are insufficient to accommodate the entire model at once. Consequently, we adopted a serial approach where we sequentially input the weights of different model layers into the RRAM array and measured the energy consumption for each layer. The RRAM macro primarily focuses on facilitating CIM operations and, as for array utilization and hardware usage, those will be determined by the overarching system architecture, compiler, and specific design choices, which fall beyond the scope of this work.

The energy measured from our 2bits/cell RCIM test chip [16] are 1.46 pJ/2bits, 0.73 pJ/2bits, 0.36 pJ/2bits, and 79 fJ/2bits for multiplying to 11, 10, 01 and 00, respectively. Additionally, the energy consumption attributed to the ADC amounted to 0.208 pJ/2bits. It's worth noting that these measurements were conducted with a cycle time of 20 nanoseconds. For a detailed breakdown of energy consumption during multiplication and ADC operations, please refer to Figure 4. Notably, the results reveal that our method exhibits superior energy efficiency (2.6x less total energy) compared to the baseline method.

4. CONCLUSION

LLMs are showing amazing performance in different application such as vision classification, vision-text, text-text and etc. However, these models are growing exponentially in size which makes it hard to deploy LLMs for generative tasks on resource-constrained edge devices such as cell phones. In addition, traditional CMOS technologies suffers from massive energy dissipation as a result of data transfer from PE and memory. Therefore, CIM architectures plays an important role for implementing energy-efficient LLMs during inference. In CIM architectures, the computations are performed in the memory itself. However, the already proposed sparsity and quantization techniques are designed for traditional CMOS technologies and cannot fully take advantage of CIM.

In this work, we proposed a CIM-aware post-training quantization technique where quantize the LLMs to 8-bit precision while sparsifying the network in the bit-level. We evaluated our method on a 2bit/cell RCIM test chip for inference mode. In 2bit/cell RCIM, the energy levels are different for various 2-bits (i.e. 00, 01, 10 and 11). Energo of multiplication to 00 is 18x less than 11 and therefore we quantized the model to the numbers that does not have 11 in their binary representations. As a result, we achieved 2.6x energy reduction compared to the uniform PTQ method. Our method gain superior energy efficiency using CIM architecture which eventually leads to enabling LLMs and generative models to be run on mobile and edge devices for accurate and energy-efficient processing. As a future works, we are planing to test our proposed method on more datasets and use different generative models such as vision to text. In addition, ADC is the bottleneck in CIM architecture which is an active area of research.

ACKNOWLEDGMENTS

This work is supported by CocoSys as a part of Semiconductor Research Corporation (SRC) JUMP 2.0 program.

REFERENCES

[1] Y. Bondarenko, M. Nagel, and T. Blankevoort, "Understanding and overcoming the challenges of efficient transformer quantization," *arXiv preprint arXiv:2109.12948*, 2021.

[2] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[5] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "A hardware-friendly approach towards sparse neural networks based on lfsr-generated pseudo-random sequences," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 751–764, 2020.

[6] F. Karimzadeh and A. Raychowdhury, "Towards cim-friendly and energy-efficient dnn accelerator via bit-level sparsity," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2022, pp. 1–2.

[7] F. Karimzadeh, J.-H. Yoon, and A. Raychowdhury, "Bits-net: Bit-sparse deep neural network for energy-efficient rram-based compute-in-memory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[8] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[9] S. Kim, C. Hooper, A. Gholami, Z. Dong, X. Li, S. Shen, M. W. Mahoney, and K. Keutzer, "Squeezellm: Dense-and-sparse quantization," *arXiv preprint arXiv:2306.07629*, 2023.

[10] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, 2021.

[11] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *Nature*, pp. 1–9, 2023.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[14] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.

[15] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40-nm 118.44-tops/w voltage-sensing compute-in-memory rram macro with write verification and multi-bit encoding," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 3, pp. 845–857, 2022.

[16] J. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "29.1 a 40nm 64kb 56.67 tops/w read-disturb-tolerant compute-in-memory/digital rram macro with active-feedback-based read and in-situ write verification," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 404–406.

[17] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization," in *European Conference on Computer Vision*. Springer, 2022, pp. 191–207.

[18] C. Zhao, T. Hua, Y. Shen, Q. Lou, and H. Jin, "Automatic mixed-precision quantization search of bert," *arXiv preprint arXiv:2112.14938*, 2021.