Quantization and Hardware Architecture Co-Design for Matrix-Vector Multiplications of Large Language Models

Wenjie Li, Aokun Hu, Ningyi Xu, and Guanghui He, *Member, IEEE*

***Abstract*— Large language models (LLMs) have sparked a new revolution in the field of natural language processing (NLP), and have garnered tremendous attention in both academic research and everyday life, thanks to their unprecedented performance in a wide range of applications. However, their deployment remains a significant challenge, primarily due to their intensive computational and memory requirements. Hardware acceleration and efficient quantization are promising solutions to address the two issues. In this paper, a quantization and hardware architecture co-design is presented for matrix-vector multiplications (MVMs) of LLMs. During quantization, we uniformly group weights and activations to ensure workload balance for hardware. To enhance the performance of quantization, we further propose two approaches called *channel sorting* and *channel selection*, which can be applied simultaneously. To support the proposed quantization scheme, we develop two precision-scalable MVM hardware architectures. They are specifically designed for high speed and high energy efficiency, respectively. Experimental results show that our proposed quantization scheme achieves state-of-the-art performance among all the reported post-training schemes that quantize both weights and activations into integers. Compared to MVM architecture of the state-of-the-art LLM accelerator OliVe, our design exhibits significant advantages in terms of area efficiency and energy efficiency.**

***Index Terms*— Large language models, quantization, hardware architecture, precision-scalable, outlier.**

I. Introduction

IN THE field of natural language processing (NLP), language models (LMs) are used to predict the probabilities of future or missing tokens. Large language models (LLMs) whose model sizes are usually over billions of parameters have been developed rapidly in the past few years. As revealed by the scaling law [1], they exhibit improved performance across many downstream tasks. In addition, they surprisingly possess the so-called emergent abilities which are not present in smaller models [2], [3], displaying sharp and unpredictable increases in performance at specific tasks as scale increases. The trend of LLMs has quickly swept through the research community and is driving a new wave of artificial intelligence (AI) technology [4], [5], [6]. Numerous LLM families such as GPT3 [7], GPT4 [8], OPT [9], LLaMA [10] and LLaMA2 [11] have been introduced by both the academic and industrial communities. Thanks to the success of ChatGPT [12], LLMs have caused a tremendous sensation and become ubiquitous in our daily lives. Furthermore, they are considered to hold promise for the realization of artificial general intelligence (AGI) systems in the future [13].

LLMs not only typically possess over billions of parameters, but are also extremely computationally intensive. The most prominent example is GPT3 [7], which boasts a staggering 175 billion model parameters. Since the advent of GPT3, several other LLMs with over hundreds of billions of parameters have been released [9], [14], [15], [16]. Without special optimization and efficient implementation techniques, the inference speed for these models typically falls below one token per second, even with multiple advanced GPUs [17]. Hardware acceleration and efficient quantization are promising solutions to address the issues raised by the intensive computational and memory requirements. Notably, quantization plays a crucial role in enabling the deployment of LLMs on either GPUs or dedicated hardware accelerators.

Recently, a growing body of research has emerged in the field of quantization for LLMs [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. Quantization-aware training (QAT) is a well-known approach to quantizing models while minimizing performance loss by retraining [28]. Unfortunately, it is extremely expensive for the billion-parameter models. In contrast, post-training quantization (PTQ) which requires no retraining is a more appealing solution. To date, the majority of reported works on quantization for LLMs have used PTQ and achieved impressive model performance [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. On the other hand, most of them focus on weight quantization, in order to reduce the memory space required for storing model parameters [18], [19], [20], [21], [22]. Additionally, some works that quantize both weights and activations into integers [23], [24], [25], [26], [27] are preferred for hardware implementations since the area cost and power consumption of fixed-point arithmetic

Manuscript received 22 October 2023; revised 8 December 2023 and 25 December 2023; accepted 3 January 2024. Date of publication 15 January 2024; date of current version 30 May 2024. This work was supported by the National Natural Science Foundation of China under Grant 62074097. This article was recommended by Associate Editor A. Bernardini. *(Corresponding authors: Ningyi Xu; Guanghui He.)*

Wenjie Li, Aokun Hu, and Ningyi Xu are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: wenjielinju@sina.com; huaokun@sjtu.edu.cn; xuningyi@sjtu.edu.cn).

Guanghui He is with the Department of Micro/Nano Electronics and the MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: guanghui.he@sjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2024.3350661.

Digital Object Identifier 10.1109/TCSI.2024.3350661

units are lower than those of their floating-point counterparts. In this paper, we focus on PTQ that quantizes both weights and activations into integers.

As a pioneer in weight-and-activation quantization for LLMs, LLM.int8() [23] isolates the weights and activations of specific channels for floating-point computation to avoid performance degradation caused by quantizing outliers. SmoothQuant [24] successfully circumvents floating-point computation and proposes an 8-bit weight-and-activation quantization approach that achieves negligible accuracy loss by offline migrating the quantization difficulty from activations to weights with a mathematically equivalent transformation. Outlier Suppression+ [26] reduces the bitwidths of weights and activations to 6 bits using similar ideas. Recently, OliVe [25] and RPTQ [27] have pushed the bitwidth limit to 4 bits, i.e., W4A4, where "W" and "A" represent weight and activation, respectively. Since OliVe suffers from an unacceptable performance loss in the W4A4 case, RPTQ has become the state-of-the-art design for low bitwidths. However, RPTQ has a significant issue that hinders its efficient hardware implementation. It divides both weights and activations into groups and quantizes each group using different set of quantization parameters. In general, the number of weights and activations varies among different groups, resulting in severe workload imbalance. Therefore, it remains a challenge to devise a quantization scheme which is not only able to quantize both weights and activations into low-bitwidth integers without significant performance loss, but is also hardware-friendly.

This paper proposes a PTQ scheme that can quantize both weights and activations into low-bitwidth integers while remaining hardware-friendly. Here the term "hardware-friendly" refers to its ease of hardware support and ability to achieve good hardware performance. Unlike RPTQ, in which group size is variable, our proposed scheme uniformly groups weights and activations to ensure workload balance. Due to the existence of outliers [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], it is not surprising that uniform grouping results in significant performance degradation, particularly for low bitwidths. We further present two approaches to enhancing the performance. The first one is *channel sorting*, which sorts the input channels based on a specific metric related to activations. Since grouping is applied after channel sorting, activations within the same group tend to have similar statistics, thereby reducing the quantization error. The second one is *channel selection*, which is employed to deal with possible outliers. A small number of channels within each group are selected and their activations are quantized using more bits. The two approaches are inherently compatible and can be applied simultaneously. To support our proposed quantization scheme, we also develop two precision-scalable hardware architectures for the computationally dominant matrix-vector multiplications (MVMs) of LLMs. They are designed specifically for high speed and high energy efficiency, respectively. Moreover, our developed architectures also support some other integer quantization schemes, such as SmoothQuant [24] and Outlier Suppression+ [26]. The proposed MVM architectures can be integrated into LLM accelerators to fully leverage the advantages of our proposed quantization scheme.

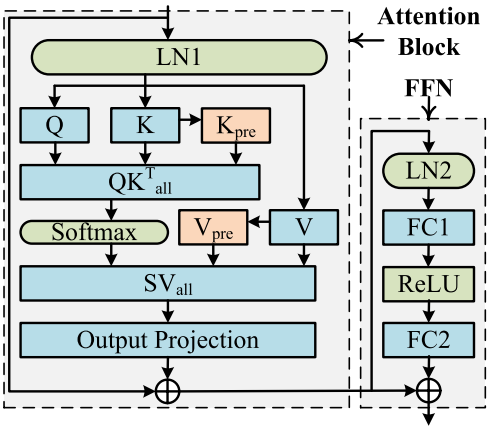


Fig. 1. Architecture of a typical Transformer block. The position of layer normalization and the usage of activations function may vary smong different LLMs.

Revealed by our experiments, the proposed quantization scheme can achieve better performance than RPTQ, which is the state-of-the-art approach for low-bitwidth quantization on both weights and activations. In addition, our proposed scheme is more hardware-friendly, since the quantization and architecture co-design successfully avoids the severe workload imbalance and only introduces limited hardware overhead. Compared to MVM architecture of the state-of-the-art LLM accelerator OliVe [25], our design demonstrates significant advantages in area efficiency and energy efficiency.

The rest of the paper is organized as follows. Section II presents the background and preliminaries. In Section III, we provide a detailed description of our proposed quantization scheme. Subsequently, Section IV develops the MVM architectures. Experimental results and comparisons are given in Section V to demonstrate the superiority of our work. Finally, Section VI draws the conclusion.

II. Background and Preliminaries

This section begins with an introduction to the fundamentals of LLMs, including the model architecture and involved computations. We then briefly review some quantization schemes that are most relevant to our proposed approach.

A. Fundamentals of LLMs

LLMs are models based on the Transformer architecture [29]. The main structure of an LLM is composed of multiple stacked Transformer blocks, each of which is referred to as a layer. As depicted in Fig. 1, a Transformer block can be further divided into two parts: an attention block and a feed-forward network (FFN). The attention block is a unique characteristic of the Transformer architecture that distinguishes it from other conventional deep neural networks (DNNs) [30]. For simplicity, we assume that the attention block has only one head. For more details about multi-head attention (MHA), please refer to [4], [5], and [6].

A layer normalization (LN) block is located in top of each attention block. It outputs a matrix denoted as $\mathbf{X}$ whose size is $n \times D$ where n is the number of tokens and D is the dimension. Note that $\mathbf{X}$ will degenerate into a vector when only one token is processed by the model, i.e., $n = 1$. Given

three weight matrices $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}, \mathbf{W}^{(V)} \in \mathbb{R}^{D\times D}$, the query, key and value matrices are computed as

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(Q)}, \mathbf{K} = \mathbf{X}\mathbf{W}^{(K)}, \mathbf{V} = \mathbf{X}\mathbf{W}^{(V)}. \tag{1}$$

For each query vector $\mathbf{Q}_i$, we compute the inner products between it and all the obtained key vectors, including current $\mathbf{K}_i$ $(0 \leq i < n)$ and the stored ones from previous computations. The inner products of all the query vectors form an $n \times (n + n_{pre})$ attention matrix

$$\mathbf{A} = \mathbf{Q}\mathbf{K}_{all}^T = \mathbf{Q}\left[\mathbf{K}^T, \mathbf{K}_{pre}^T\right]. \tag{2}$$

After applying row-wise softmax to $\mathbf{A}$, a score matrix $\mathbf{S}$ can be obtained. Further, we compute

$$\mathbf{H} = \mathbf{S}\mathbf{V}_{all} = \mathbf{S}\begin{bmatrix}\mathbf{V}\\ \mathbf{V}_{pre}\end{bmatrix}, \tag{3}$$

where $\mathbf{V}_{pre}$ consists of previous value vectors. Before the residual connection, output projection is also required:

$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(O)}, \tag{4}$$

where $\mathbf{W}^{(O)}$ is a $D \times D$ weight matrix.

An FFN is composed of a residual connection, an LN block, two full-connection (FC) layers, and an activation function block. The usage of activation functions may vary among different LLMs. For example, OPT [9] and LLaMA [10] adopt ReLU [31] and SwiGLU [32], respectively.

LLM inference consists of two stages which are referred to as prompting and generation, respectively. During the prompting stage, prompt sentences are fed into the model and all the tokens are processed simultaneously. However, computation in the generation stage becomes token-wise. At each time step, the model processes only one token, and the output token will be fed back to the model for the next time step. In this case, we have $n = 1$. The computations involved in the blue-colored blocks depicted in Fig. 1 are matrix-vector or matrix-matrix multiplications. Compared to other operations, these matrix multiplications play an overwhelmingly dominant role in terms of computational complexity. All the reported quantization schemes for LLMs [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [58], [59], [60], [61] focus on them, and so does our work.

B. Related Quantization Schemes

Quantization is a technique that maps a large set of values into a smaller set. The primary objective of quantization is to reduce bitwidths and therefore storage requirements. Lower bitwidths can also simplify the hardware complexity of arithmetic units. Uniform quantization is the most attractive scheme due to its better hardware support and efficiency. It is also common to use integers (fixed-point numbers) to represent the quantized numbers. In this paper, we consider uniform quantization that maps floating-point numbers into integers.

Formally, a floating-point number r can be quantized as follows [33]:

$$r = S(q - Z), \tag{5}$$

where q is an integer, S is the scaling factor and Z is the zero point. This indicates that the mapping rule is determined once we have a scaling factor and a zero point:

$$q = round(\frac{r}{S}) + Z. \tag{6}$$

Given b bits to represent the integer, its possible values range in $\left[-2^{b-1}, 2^{b-1} - 1\right]$. Accordingly, scaling factor and zero point are computed as follows:

$$S = \frac{r_{max} - r_{min}}{2^b}, \ Z = -round(\frac{r_{max} + r_{min}}{2S}), \tag{7}$$

where r_{max} and r_{min} are the maximum and minimum values obtained from a calibration dataset, respectively.

SmoothQuant [24] is the pioneer that successfully quantizes both weights and activations into 8-bit integers. Observing that activations in some specific channels exhibit larger absolute values, SmoothQuant proposes to make a mathematical transformation to scale down the activations in the outlier channels. It introduces a diagonal matrix $\mathbf{s}$, the diagonal elements of which are scaling factors for all the channels. Then the multiplication between $\mathbf{X}$ and $\mathbf{W}^{(u)}$ becomes

$$(\mathbf{X}\mathbf{s}^{-1}) \cdot (\mathbf{s}\mathbf{W}^{(u)}), \tag{8}$$

where $u \in \{Q, K, V, O\}$. Outlier Suppression+ [26] reduces bitwidth to 6 bits using similar ideas. It also shifts the activations to align the center of each channel to zero. However, SmoothQuant suffers from sever performance loss when the bitwidth is reduced to 4 bits, which will be shown in Table I of Section V-B. Outlier Suppression+ also fails to provide 4-bit quantization results.

GPTQ [18] is a weight-only quantization scheme that can achieve a bitwidth limit of four bits with negligible performance loss. It aims to find a quantized weight matrix $\hat{\mathbf{W}}^{(u)}$ which minimizes the squared error related to $\mathbf{X}\mathbf{W}^{(u)}$, where $u \in \{Q, K, V, O\}$. Formally, it can be stated as

$$\arg\min_{\hat{\mathbf{W}}^{(u)}} ||\mathbf{X}\mathbf{W}^{(u)} - \mathbf{X}\hat{\mathbf{W}}^{(u)}||_2^2. \tag{9}$$

Obviously, once some weights have been quantized, errors will occur. A feasible approach to compensate for the errors is to update the not-yet-quantized weights. GPTQ performs the weight updating channel by channel (row by row). After quantizing a row of the weight matrix, some of the remaining rows will be updated by subtracting the error values computed based on the second-order information.

RPTQ [27] is built upon GPTQ and further quantizes the activations using 8 or 4 bits. It is the first reported work that uses W4A4 to quantize LLMs while exhibiting acceptable performance loss. Observing that there are notable variations in the activations across channels, RPTQ groups the channels and quantizes each group using different set of parameters. More specifically, RPTQ is performed in the following steps.

1) Perform inference on a calibration dataset and derive the maximum and minimum activation values for each input channel.
2) Employ k-means algorithm to categorize the distinct channels into clusters (groups), based on the points

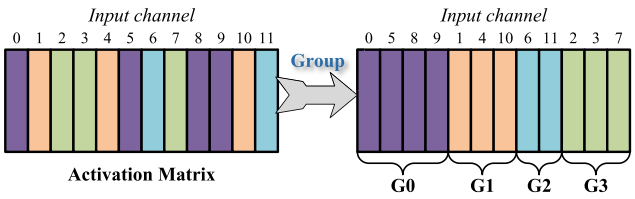


Fig. 2. An illustrative example for the grouping processing in RPTQ.

formed by the obtained maximum and minimum activation values.

3) Update weights using the algorithm of GPTQ [18], and compute quantization parameters S and Z for each group of weights. Quantize the updated weights using the obtained parameters S and Z.
4) With the quantized weights, perform inference on the calibration dataset and compute quantization parameters S and Z for each group of activations.

Fig. 2 uses an illustrative example to visualize the grouping processing for an activation matrix in RPTQ. Take the computation for $\mathbf{Q}$ as an example (see (1)). Note that one column of activation matrix $\mathbf{X}$ and its corresponding row of weight matrix $\mathbf{W}^{(Q)}$ belong to the same channel. The columns of $\mathbf{X}$ are shuffled during grouping. Meanwhile, the rows of $\mathbf{W}^{(Q)}$ are shuffled with the same order. Unlike some other quantization schemes that deliberately propose solutions to deal with outliers, RPTQ handles this issue implicitly by assigning outliers to the same group.

III. Proposed Quantization Scheme

Our quantization scheme is proposed in this section. We first discuss the hardware implementation shortcomings of RPTQ and analyze the necessity of uniform grouping. Then we present two approaches to enhancing the model performance when applying uniform grouping.

A. Uniform Grouping

Consider d-dimension weight and activation vectors

$$\boldsymbol{w} = \left[w_0, w_1, \cdots, w_{d-1}\right]^T, \quad \boldsymbol{a} = \left[a_0, a_1, \cdots, a_{d-1}\right]. \tag{10}$$

Note that $\boldsymbol{w}$ is a column vector of the weight matrix, while $\boldsymbol{a}$ is a row vector of the activation matrix (for example, see (1)). According to (5), their inner product can be expressed as:

$$\begin{aligned}\boldsymbol{aw} &= \sum_{i=0}^{d-1} S_i^{(w)} S_i^{(a)} (q_i^{(w)} - Z_i^{(w)})(q_i^{(a)} - Z_i^{(a)}) \\ &= \sum_{i=0}^{d-1} S_i^{(w)} S_i^{(a)} q_i^{(w)} q_i^{(a)} - \sum_{i=0}^{d-1} S_i^{(w)} S_i^{(a)} (q_i^{(w)} Z_i^{(a)} \\ &\quad + q_i^{(a)} Z_i^{(w)}) + \sum_{i=0}^{d-1} S_i^{(w)} S_i^{(a)} Z_i^{(w)} Z_i^{(a)}. \end{aligned} \tag{11}$$

Without grouping, all the entries of a vector share the same scaling factor and zero point. Namely, we have $S_i^{(w)} = S^{(w)}$, $S_i^{(a)} = S^{(a)}$, $Z_i^{(w)} = Z^{(w)}$, and $Z_i^{(a)} = Z^{(a)}$, for $0 \le i < d$.

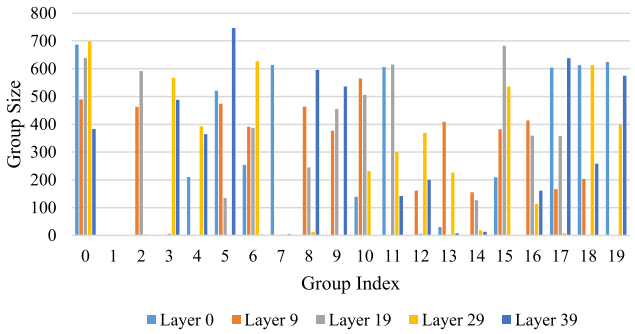


Fig. 3. Comparison of group size for different FC2 layers when applying RPTQ. The model is OPT-13B and the number of groups is $N_{group} = 20$.

Consequently, (11) becomes

$$\begin{aligned}\boldsymbol{aw} &= S^{(w)} S^{(a)} \sum_{i=0}^{d-1} q_i^{(w)} q_i^{(a)} + S^{(w)} S^{(a)} Z^{(a)} \sum_{i=0}^{d-1} q_i^{(w)} \\ &\quad + S^{(w)} S^{(a)} Z^{(w)} \sum_{i=0}^{d-1} q_i^{(a)} + d S^{(w)} S^{(a)} Z^{(w)} Z^{(a)}. \end{aligned} \tag{12}$$

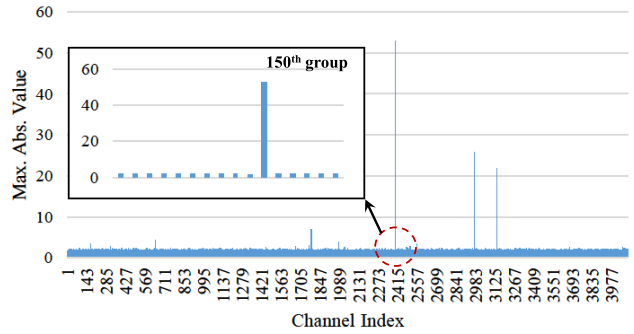
The computational complexity of multiplications for the four terms on the right-hand side of the equation is $O(d)$, $O(1)$, $O(1)$, and $O(1)$, respectively. Therefore, most reported accelerators for DNNs [34], [35], [36], [37], [38] and Transformers [25], [39], [40], [41], [42], [43] focus on accelerating the inner product of the integer vectors: $\boldsymbol{q}^{(w)}\boldsymbol{q}^{(a)} = \sum_{i=0}^{d-1} q_i^{(w)} q_i^{(a)}$. The remaining computations can be left for software or other dedicated computing units. Our design follows this convention and focuses on computing the inner product of integer vectors.

In RPTQ, $\boldsymbol{w}$ and $\boldsymbol{a}$ are grouped using the same grouping rule. Denote the group number by N_{group}. Then the set of entry indices involved in the ith group can be denoted as G_i, for $0 \le i < N_{group}$. For example, after the grouping processing shown in Fig. 2, we have $N_{group} = 4$, $G_0 = \{0, 5, 8, 9\}$, $G_1 = \{1, 4, 10\}$, $G_2 = \{6, 11\}$ and $G_3 = \{2, 3, 7\}$. Since different groups have their own quantization parameters, the first term on the right side of (12) can be written as

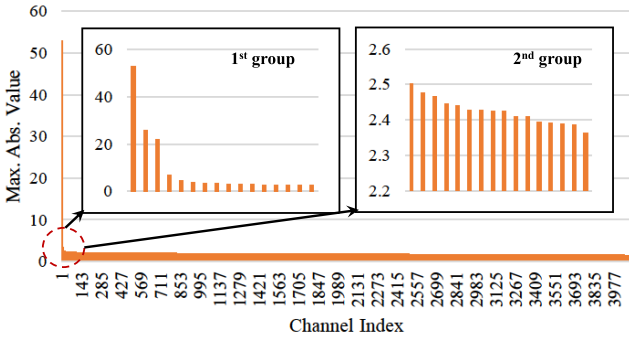
$$\sum_{i=0}^{N_{group}-1} S_i^{(w)} S_i^{(a)} \sum_{j \in G_i} q_j^{(w)} q_j^{(a)}. \tag{13}$$

It shows that each group requires the computation of an inner product, i.e., $\boldsymbol{q}_i^{(w)}\boldsymbol{q}_i^{(a)} = \sum_{j \in G_i} q_j^{(w)} q_j^{(a)}$. Revealed by our experiments, there is a significant difference in the sizes of different groups. Fig. 3 provides an illustrative example that compares the group sizes for different FC2 layers. Obviously, the group size significantly varies within a layer. Besides, there is a clear distinction in the grouping results among different layers. A more interesting finding is that some groups consist of only one or two channels. These channels often exhibit activations with larger absolute values and are referred to as outliers, which have been reported in many existing studies [18], [19], [20], [21], [22], [23], [24], [25], [26], [27].

The number of channels contained in each group shows poor regularity, resulting in workload imbalance in hardware implementation. Consider P_{group} computing units each of



(a) Grouping result without channel sorting.



(b) Grouping result with channel sorting.

Fig. 4. Grouping results without and with channel sorting. The uniform grouping is applied to **X** of the 10th layer of OPT-6.7B. In this example, the group size is set to 16 and the maximum absolute value is selected as the metric.

which is capable of handling a group independently. When $P_{group} = 1$, all the groups can be sequentially processed without workload imbalance. However, in practice, it is common to have $P_{group} > 1$, in order to achieve a higher degree of parallelism. If we synchronize all the P_{group} computing units when they are processing their own groups, a problem of severe workload imbalance arises. A possible solution is to allow each computing unit to sequentially process groups, instead of being synchronized with others. In this case, there still exist some challenges. First, it requires an scheduling algorithm to assign groups to different computing units such that the workload is balanced as much as possible. Designing such an algorithm is not a trivial task. Moreover, it can not always perfectly balance the workload. Consider a toy example in which $P_{group} = 2$, $N_{group} = 4$, and the group sizes are 1, 2, 10, and 100, respectively. Clearly, the best arrangement is assigning the first three groups to a computing unit while the fourth group is assigned to the other one. The workload imbalance is still serious, as the sum of 1, 2, and 10 is much smaller than 100. The above discussion assumes that a computing unit processes the input channels sequentially, i.e., $P_{ic} = 1$ (here P_{ic} is equal to P_{entry} of Fig. 6). With a larger P_{ic}, the workload problem may become worse as some groups contain only one or two channels. One can also map the computations involved in a single group to all the P_{group} computing units at a time. In this case, the workload imbalance will lead to low hardware utilization, especially for the groups with extremely small sizes.

In order to make the quantization scheme more hardware-friendly, we propose to uniformly group the input channels. Instead of using k-means algorithm to categorize the distinct channels, we enforce groups to contain an equal number of channels, ensuring workload balance. Note that the matrices **Q**, **K**, **V**, and the input matrix in output projection exhibit characteristics that are strongly associated with the attention heads. Thus grouping of these matrices needs to be strictly constrained within each attention head. In other words, the group sizes for these matrices should not exceed the head dimension. However, there is no doubt that uniform grouping will bring a significant performance loss, to deal with which we next propose two optimization approaches.

B. Two Approaches to Enhancing Model Performance

The activations of outlier channels typically have larger absolute values and are more sensitive to quantization. The most common approach to addressing the issue is to use more bits or keep them in floating-point format [23]. In fact, the activations of outlier channels are quantized using the same quantization parameters as other channels within each group. RPTQ alleviates the impact of outlier quantization on performance by isolating them into individual groups. However, uniform grouping fails to achieve this. In this context, we propose two approaches to enhancing the model performance.

The first one is *channel sorting* which sorts the input channels before grouping. The sorting is based on a metric that characterizes the sensitivity of activations to quantization. For example, the maximum absolute value is a feasible metric. It appears preferable to use the same quantization parameters to quantize activations with similar maximum absolute values. Fig. 4 uses an example to compare the grouping results without and with the channel sorting. The results are consistent with the observation that a few outlier channels have activations with larger absolute values. Without channel sorting, some groups suffer from significant quantization error. For example, the scaling factor of the 150th group will have an absolute value exceeding $55/2^4 \approx 3.44$, if we use four bits for its quantization. Since the activations of other channels in this group have absolute values smaller than 3.44, they will be quantized into the same integer, according to (6). In Fig. 4(a), there are four groups containing significant outlier channels. When grouping after channel sorting, all the outliers are grouped together and the remaining groups exhibit a more uniform distribution of activation values.

The second approach to enhancing the performance is *channel selection*. We identify the outlier channels using the adopted metric, such as the maximum absolute value. More specifically, we perform top-k algorithm to identify the outlier channels with the K largest maximum absolute values. Since the outliers constitute only a small proportion of all channels, we propose assigning double the number of bits for quantizing their activations, resulting in a negligible increase on overall computational complexity. Note that the number of bits for quantizing weights remains unchanged. As shown in (13), each group has a pair of scaling factors and zero points, for quantizing activations and weights respectively. Recall the

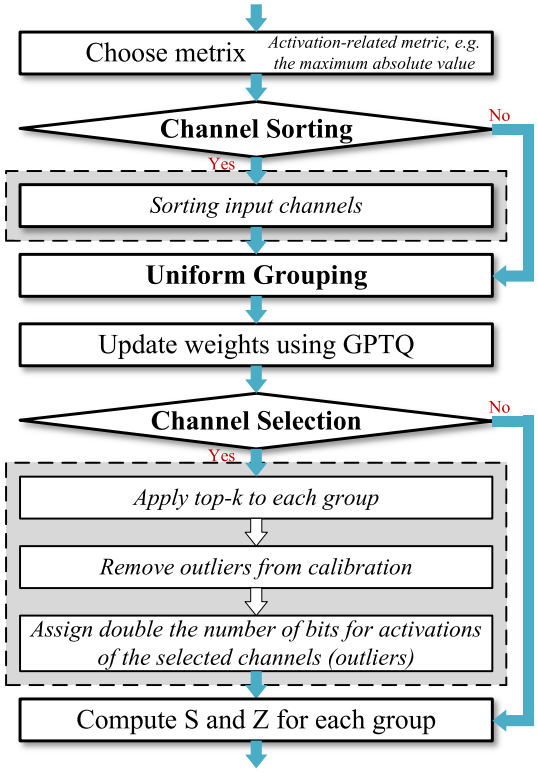


Fig. 5. Flowchart of our proposed quantization scheme.

example given when we discuss channel sorting (150th group of Fig. 4(a)). Outlier channels usually cause scaling factors with larger absolute values, leading to a significant increase in quantization error for other channels. To address this issue, we exclude the selected outlier channels when we compute quantization parameters during calibration. After performing the top-k algorithm, the number of selected outlier channels may vary among the groups. It implies workload imbalance, similar to what we have discussed for RPTQ. To balance the workload, we apply top-k algorithm to each group individually. K outlier channels will be selected from each group. The number of all selected channels is KN_{group}.

In fact, the two proposed approaches are inherently compatible and can be applied simultaneously. As depicted in Fig. 4(b), the influence of outliers on quantization error may still be significant in the first few groups after channel sorting. Additionally introducing channel selection is expected to further alleviate this issue. Flowchart of our proposed quantization scheme is shown in Fig. 5. We first choose a reasonable metric to characterize the sensitivity of activations to quantization. Then, sort input channels in a descending order according to the metric, if channel sorting is invoked. After that, uniformly group the channels and perform the weight updating algorithm of GPTQ. Subsequently, we can adopt channel selection, in which we identify the outlier channels by performing top-k algorithm for each group. The outlier channels will be removed from calibration. Meanwhile, they are assigned double the number of bits for activation quantization. Finally, quantization parameters for each group will be computed, similar to Step 3) and 4) of RPTQ. Once again, we emphasize that the number of bits for quantizing weights remains unchanged when applying channel selection. All quantization parameters are computed offline for a certain model. In channel sorting, the channels are sorted in advance rather than on-the-fly, introducing no extra computational overhead.

IV. Proposed Hardware Architectures

During generation stage, an LLM processes only one token at a time, which is a distinct difference from the prompting stage where multiple tokens are processed simultaneously. Some of the existing Transformer accelerators such as those proposed in [39], [40], and [43] are targeted at processing multiple tokens and are not suitable for LLMs due to the low hardware utilization during generation stage. On the other hand, the computational complexity of LLMs is dominated by the linear layers and attention blocks. As reported in [22], the GPU runtime for the nonlinear operations (e.g. softmax and LN) is typically less than one-tenth that of the linear layers and attention blocks. In this context, matrix multiplications dominate the computation of LLMs, making their hardware acceleration crucial. We next develop hardware architectures for MVMs of LLMs. Cooperating with our proposed quantization scheme, the MVM architectures can be integrated into LLM accelerators. Note that the matrix-matrix multiplications required by prompting stage can also be decomposed into MVMs.

In general, fixed-point (integer) arithmetic units have lower area cost and power consumption than their floating-point counterparts [48], [49]. Therefore, quantization schemes that quantize both weights and activations into integers are preferred for hardware implementation. The existing quantization schemes for LLMs [24], [25], [26], [27] offer several different bitwidth configurations, among which eight and four bits are the most commonly used. Within a Transformer block, the bitwidths for different operations may vary. For example, in the W4A4 configuration of RPTQ, the activations output by LN blocks are quantized using eight bits to improve the model performance. These facts indicate that precision scalability is crucial in hardware. Furthermore, recent findings in [47] demonstrate that 4-bit bitwidth is nearly universally optimal in terms of total model bits and zero-shot accuracy. Accordingly, the bit granularity of our proposed architectures is set to four bits.

Fig. 6 depicts the proposed MVM architecture. It is composed of $P_{oc} \times P_{group}$ processing elements (PEs), each of which is in charge of computing the inner product of one group. Every cycle, a PE computes the products of P_{entry} pairs of weights and activations from the same group, and then adds them together. Some shifters are deployed to support precision scalability. Note that the lowest supported precision is W4A4. P_{oc} columns of PEs correspond to P_{oc} different output channels, while P_{group} rows of PEs correspond to P_{group} different groups. The PEs in each row share the same activation bits from one group. As we have discussed in Section III-A, the proposed architecture also works for RPTQ when $P_{group} = 1$ and $P_{entry} = 1$. Fig. 7 illustrates dataflow for the proposed MVM architecture with $P_{oc} = P_{group} = P_{entry} = 2$. In order to clarify the fundamental computing process, we consider

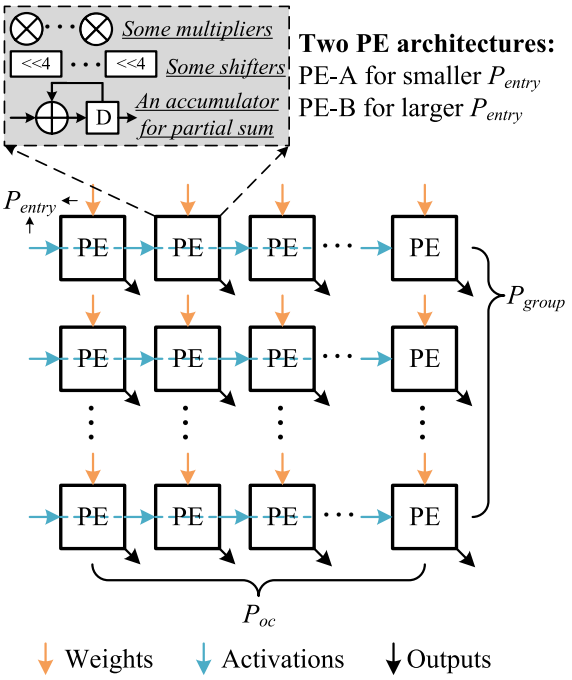


Fig. 6. The proposed MVM architecture.

the lowest supported precision W4A4 here. Hence no shifting is invoked in this example. The first 32 cycles are used for the computations of the first two output channels $OC0$ and $OC1$. More specifically, cycles 0∼15 are used to compute the inner products of groups G_0 and G_1, while cycles 16∼31 are used to compute the inner products of groups G_2 and G_3. From cycle 32, the computation proceeds to the next two output channels $OC2$ and $OC3$. In this example, we assume that channel sorting and channel selection are excluded for the sake of brevity. When we discuss the architectures of PE later, we will present the dataflows that takes into account both channel sorting and channel selection.

While the top-level architecture given in Fig. 6 seems to be common, it helps to make the dataflow easier to understand. The architecture innovation primarily lies at the PE level. We propose two architectures for PE, as depicted in Fig. 8. The two architectures, denoted as PE-A and PE-B, are intended for smaller and larger P_{entry}, respectively. Both architectures have P_{entry} multiplier units (MUs) to deal with the input P_{entry} pairs of weights and activations from the same group. These weights and activations will be multiplied and added together within one clock cycle. To support precision scalability, shifters are employed in both of the two architectures. Note that these shifters are implemented by hard wiring which introduces no hardware cost. Current partial sum is fed into the accumulate unit (ACU) to be added to the previous partial sum. When configured into the W4A4 mode, no shifting is required. If higher-precision mode is invoked, current partial sum will be shifted accordingly. Now take the W8A8 mode as an example. To compute the multiplication of two signed numbers A and B, we can decompose the computation into $A[0:8] \times B[0:8] = A[0:3] \times B[0:3] + (A[0:3] \times B[4:7]) \ll 4 + (A[4:7] \times B[0:3]) \ll 4 + (A[4:7] \times B[4:7]) \ll 8$. It should be noted that $A[0:3]$ and $B[0:3]$ are unsigned numbers while $A[4:7]$ and $B[4:7]$ are signed numbers. To support the multiplications of both signed and unsigned 4-bit numbers, MU adopts 5-bit signed multipliers. The computation of inner product is accordingly divided into four stages, during each of which the partial sums corresponding to the higher-order bits are shifted before being summed. It is worth noting that although using arithmetic units with high bitwidths can support low-precision computations by setting the high-order bits to zero, it can lead to lower hardware utilization. In contrast, decomposing high-precision multiplications and supporting them using arithmetic units with lower bitwidths have been widely adopted in existing hardware designs [34], [35], [36], [37], [38].

Since PE-A is developed for smaller P_{entry}, it enjoys a shorter critical path and is thus a preferable choice for high-speed implementations. Our proposed channel selection approach can be supported by PE-A without requiring additional multipliers. Higher-order bits of activations in the selected channels can be processed by PE-A through additional cycles. Recalling that the number of selected channels in each group is K, the computation for each group can be accomplished by working for additional $\lceil K/P_{entry} \rceil$ clock cycles. An example of the dataflow for PE-A in W4A4 mode is shown in Fig. 9(a), where $K = 2$, $P_{entry} = 1$ and the group size is 8. Two extra cycles are used to process the four higher-order bits of the activations in the selected channels. When P_{entry} is sufficiently large, the negative impact on throughput may be prominent. For instance, assume the group size is 8 and $P_{entry} = 8$. Even when K is equal to 2 or 1, it is still necessary to work for an additional cycle. In this case, the channel selection reduces the throughput by 50%. However, the reduction in throughput is only 12.5% in the example shown in Fig. 9(a) because its P_{entry} is fairly small.

As we have analyzed, when P_{entry} is large, it is better to employ more multipliers to deal with the higher-order bits, instead of introducing additional cycles. Fig. 8(b) depicts the architecture of PE-B, in which k multiply-shift units (MSUs) are deployed to handle the high-order activation bits in the selected channels. Denoting the group size by $|G|$, the computations of a group are completed within $\lceil |G|/P_{entry} \rceil$ cycles. Consequently, at most $\lceil |G|/P_{entry} \rceil k$ selected channels can be supported. Fig. 9(b) illustrates how PE-B handles the computations that are handled by PE-A in Fig. 9(a). In this example, PE-B has $P_{entry} = 8$ MUs and additionally employs $k = 2$ MSUs for the $K = 2$ selected channels.

With PE-A and PE-B, we actually have two MVM architectures. Obviously, there are three adjustable degrees of parallelism: P_{oc}, P_{group} and P_{entry}. One can improve the throughput by increasing the degrees of parallelism. Note that there exist some practical limitations on the maximum achievable values for P_{oc}, P_{group} and P_{entry}. Since each attention head is processed individually, the value of P_{oc} cannot exceed the head dimension, which is typically 64 or 128 for LLMs. P_{group} and P_{entry} are limited by the group number and group size, respectively. The group sized is also restricted by the head dimension, which means that the value of P_{entry} cannot exceed the head dimension either.

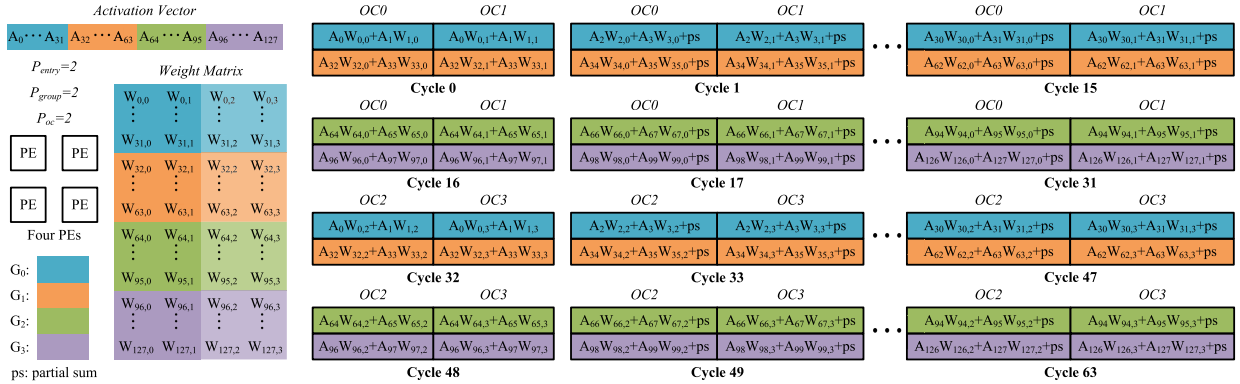
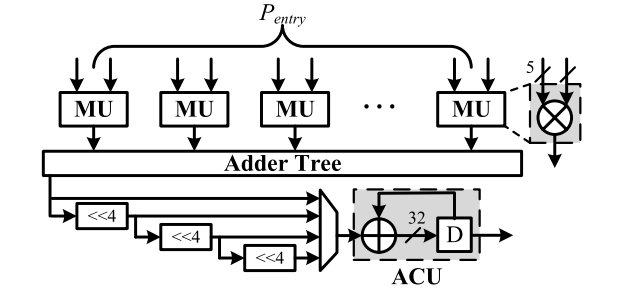


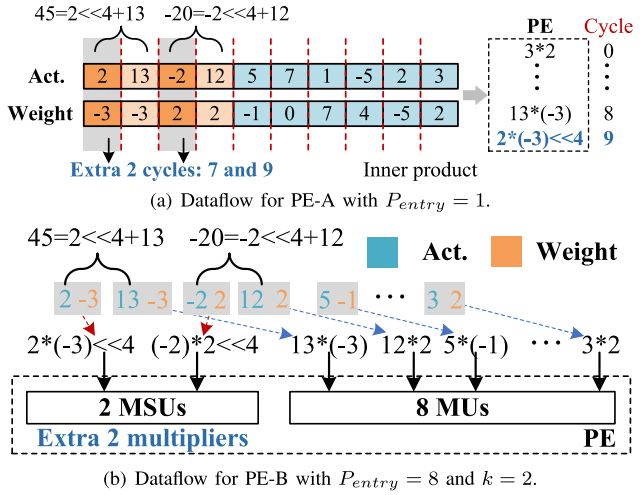
Fig. 7. Dataflow example for the MVM architecture with $P_{oc} = P_{group} = P_{entry} = 2$. The lowest supported precision W4A4 is considered here. In this example, vector dimension (number of input channels) is 128, number of output channels is 4, number of groups is 4, and the group size is 32. Uniform grouping is applied. For simplicity, channel sorting and channel selection are excluded in this example.



(a) Architecture PE-A for smaller P_{entry}.

(b) Architecture PE-B for larger P_{entry}.

Fig. 8. Two architectures of PE, which are denoted as PE-A and PE-B, respectively.



(a) Dataflow for PE-A with $P_{entry} = 1$.

(b) Dataflow for PE-B with $P_{entry} = 8$ and $k = 2$.

Fig. 9. Dataflow examples for PE-A and PE-B in the W4A4 mode. In these examples, group size is 8 and $K = 2$.

A larger P_{entry} leads to a higher area cost of the adder tree and a longer critical path. Another noteworthy fact is that the area cost of an ACU is higher than that of an MU. In our experiments, we have observed that increasing P_{entry} results in smaller hardware overhead than increasing P_{oc} or P_{group}. As we will show in Section V-C, architecture deploying PE-B offers better energy efficiency than that deploying PE-A. In summary, the architectures based on PE-A and PE-B are designed to achieve high speed and better energy efficiency, respectively. By setting $P_{group} = P_{entry} = 1$, our proposed MVM architectures also work for RPTQ whose group size is not a constant. In addition, the two architectures are compatible with some of the existing W8A8 quantization schemes which quantize both weights and activations into integers, such as SmoothQuant [24] and Outlier Suppression+ [26].

V. Experimental Results and Comparisons

In this section, we present experimental results and comparisons with the state-of-the-arts designs to demonstrate the effectiveness and superiority of our work.

A. Experimental Settings

In the experiments of this section, the metric employed by our quantization scheme is $|x_{max}| + |x_{min}|$, where x_{max} and x_{min} are the maximum and minimum activations in a channel, respectively. Other feasible metrics, such as $|x|_{max}$, can also be used. In our experiments, we have observed that adopting $|x_{max}| + |x_{min}|$ or $|x|_{max}$ did not have a significant difference in performance. We finally chose $|x_{max}| + |x_{min}|$ since we found that it performed slightly better in some cases.

With batch size 1 and practical sequence length (frequently smaller than 2048), weights of LLMs dominate the memory requirement, which is the reason why weight compression has recently received tremendous attention [18], [19], [20], [21], [22]. It has been demonstrated that applying 4-bit quantization to the weights of LLMs can maintain good model performance while significantly reducing memory usage [18], [19], [20], [47]. Consequently, RPTQ quantizes weights using 4 bits and so does our proposed quantization scheme. For quantization scheme using 8-bit weights, please refer to SmoothQuant [24] and Outlier Suppression+ [26], which are also supported by our MVM architectures as mentioned earlier. Following RPTQ [27], OPT family [9] is evaluated in our experiments. We use 256 samples randomly selected from WikiText2

TABLE I

PERPLEXITY (↓) COMPARISON FOR THE PROPOSED QUANTIZATION SCHEME, RPTQ [27] AND SMOOTHQUANT [24] IN LANGUAGE MODELING TASKS

Bit Configuration		W4A8				W4A4			
Task		WIKI	PT	C4	Average	WIKI	PT	C4	Average
OPT-1.3b ($D_{head}=64$, $N_{head}=32$)	SmoothQuant [24]	16.89	19.35	16.26	17.5	68.25	68.44	53.77	63.49
	RPTQ [27]	**15.22**	17.77	**15.47**	**16.15**	18.62	21.38	17.80	19.27
	Ours ($\|G\|=64$, $K=8$)	15.39	**17.75**	**15.47**	16.20	**18.31**	**21.07**	**17.65**	**19.01**
OPT-6.7b ($D_{head}=128$, $N_{head}=32$)	SmoothQuant [24]	11.62	14.04	12.47	12.71	51.19	67.14	73.44	63.92
	RPTQ [27]	**11.15**	13.75	**12.21**	12.37	12.27	15.62	13.19	13.69
	Ours ($\|G\|=128$, $K=8$)	11.34	**13.51**	**12.21**	**12.35**	**12.09**	**14.64**	**13.04**	**13.26**
OPT-13b ($D_{head}=128$, $N_{head}=40$)	SmoothQuant [24]	12.55	14.73	12.20	13.16	235.38	285.85	187.96	236.40
	RPTQ [27]	**10.78**	13.91	**11.60**	12.10	14.06	16.39	16.03	15.49
	Ours ($\|G\|=128$, $K=8$)	10.97	**12.83**	11.67	**11.82**	**12.62**	**14.78**	**13.91**	**13.77**

* The data for SmoothQuant is sourced from [27]. For each layer, our quantization scheme and RPTQ use the same number of groups. In the W4A4 mode, activations output by the two LN blocks are quantized using 8 bits for better model performance, which is suggested by RPTQ [27].
** When computing $\mathbf{QK}^T$, our channel selection is only performed on $\mathbf{Q}$. For our quantization scheme, channel sorting is not applied on OPT-13b.
*** Since we double the number of bits for the activations of outlier channels, the bitwidths of these activations become 16 and 8 for W4A8 and W4A4, respectively.

(WIKI) [50], Pen Treebank (PT) [51], and C4 [52] to form a calibration dataset.

We implement the quantization scheme in PyTorch, based on the open-source code provided in [27]. We implement the hardware architectures in Verilog RTL and synthesize them using Synopsys design compiler with a TSMC 65nm CMOS technology library, in order to estimate area and power. The voltage is set to 1.08 V. In MVMs, the number of additions is almost equal to that of multiplications. Without loss of generality, an operation refers to a single multiplication when we evaluate throughput. The value of throughput is obtained by dividing the number of multiplications processed within a cycle by the clock period. The area efficiency and power efficiency are computed as

$$AE = \frac{Throughput}{Area} \text{ and } PE = \frac{Throughput}{Power}, \tag{14}$$

respectively.

B. Model Performance After Quantization

Table I presents a comparison of perplexity scores (lower is better) for our proposed quantization scheme, RPTQ [27] and SmoothQuant [24] across three language modeling tasks. We also present the perplexity scores for floating-point baselines and GPTQ [18] in Table II. Although SmoothQuant has demonstrated that it performs well in the W8A8 configuration, it suffers from severe performance degradation with lower bitwidths, especially for W4A4. In contrast, RPTQ achieves state-of-the-art performance in the W4A8 and W4A4 modes. In most cases, our method can further outperform RPTQ, especially for larger models and lower bitwidths. We also compare our quantization scheme to RPTQ in six zero-shot tasks, as shown in Table III. The results indicate that zero-shot tasks exhibit better robustness to quantization. With the exception of the W4A4 case for OPT-13b, our quantization scheme outperforms RPTQ in terms of average accuracy. Overall, it achieves better model performance than RPTQ.

Table IV demonstrates the effectiveness of the proposed channel sorting and selection. Both of them can effectively enhance the performance, except that channel sorting fails for OPT-13b. After channel sorting, the first several groups contain many outlier channels. However, the quantization parameters are determined by the first channel of each group. Quantizing the other outliers using these parameters may lead to severe quantization error. This is a possible reason why channel sorting does not work well for the OPT-13b model. For all the evaluated models, combining channel sorting and channel selection sometimes can further improve the performance.

TABLE II

PERPLEXITY (↓) FOR BASELINES (FLOATING-POINT) AND GPTQ [18]

Task		WIKI	PT	C4	Average
OPT-1.3b	Baseline	14.62	16.96	14.72	15.43
	GPTQ*[18]	14.82	17.30	15.10	15.74
OPT-6.7b	Baseline	10.86	13.09	11.74	11.90
	GPTQ [18]	11.07	13.31	11.97	12.12
OPT-13b	Baseline	10.13	12.34	11.20	11.22
	GPTQ [18]	10.20	12.46	11.31	11.32

* GPTQ is a weight-only quantization scheme. In this table, its weight bitwidth is set to 4. For OPT-1.3b, OPT-6.7b, and OPT-13b, uniform grouping is applied and the grouping sizes are the same as our proposed quantization scheme given in Table I, i.e., 64, 128, and 128, respectively.

The number of selected channels in the top-k algorithm may have an impact on the performance. In Table V, we compare the average perplexity achieved by our proposed quantization scheme with different K. As K increases, there is a decreasing trend in perplexity. For W4A4, the decreasing trend is more pronounced. Since K is much smaller than the group size which we typically set to 64 or 128, the number of additional high-order activation bits is insignificant. For example, only 6.25% additional activation bits are introduced when $K = 8$ and $|G| = 128$. Our proposed quantization scheme is able to make a good tradeoff between model performance and the number of extra activation bits by adjusting K.

The group size can also affect model performance, as shown in Fig. 10. In general, when the group size is relatively large, perplexity decreases as the group size decreases. However, excessively small group sizes can lead to a decline in performance. On the other hand, excessively small group sizes may also undermine their benefits in terms of storage and computational efficiency. Assume scaling factors and zero points are represented using 16 bits. When the group size is 128, the average number of bits used to represent scaling factors and zero points is $16 \times 2/128 = 0.25$. If we decrease the group size to 16, it will increase to 2. Moreover, as we can see in (13), the number of multiplications between scaling factors and the integer inner products will also increase with

TABLE III

ACCURACY (↑) COMPARISON BETWEEN THE PROPOSED QUANTIZATION SCHEME AND RPTQ [27] IN ZERO-SHOT TASKS

	Task	LAMBADA [53]	PIQA [54]	ARC(easy) [55]	ARC(challenge) [55]	OpenBookQA [56]	BoolQ [57]	Average
OPT-1.3b	Baseline	58.01	72.42	50.97	29.52	33.00	57.83	50.29
	GPTQ (W4)	59.11	71.16	50.17	29.10	32.60	56.91	49.84
	RPTQ(W4A8)	56.12	**71.16**	**50.51**	28.92	**33.00**	56.64	49.39
	Ours(W4A8)	**58.84**	70.57	49.75	**29.35**	32.40	**58.20**	**49.85**
	RPTQ(W4A4)	50.69	69.31	47.90	**27.30**	**33.00**	55.11	47.22
	Ours(W4A4)	**51.19**	**69.37**	**48.11**	26.71	31.20	**59.30**	**47.65**
OPT-6.7b	Baseline	67.69	76.55	60.06	34.64	37.20	66.06	57.03
	GPTQ (W4)	68.87	77.09	60.19	34.64	37.20	66.48	57.41
	RPTQ(W4A8)	68.15	**76.66**	**60.19**	34.04	**38.80**	66.09	57.32
	Ours(W4A8)	**69.01**	**76.66**	**60.19**	**34.13**	37.40	**66.82**	**57.37**
	RPTQ(W4A4)	63.26	75.52	57.15	33.87	**38.40**	63.27	55.25
	Ours(W4A4)	**65.94**	**76.39**	**58.96**	**34.47**	37.40	**64.40**	**56.26**
OPT-13b	Baseline	68.66	76.82	61.83	35.67	39.00	65.84	57.97
	GPTQ (W4)	69.98	76.99	61.32	34.90	39.40	66.88	58.25
	RPTQ(W4A8)	68.74	76.82	**62.63**	**35.49**	38.00	65.63	57.89
	Ours(W4A8)	**69.75**	**77.09**	61.53	35.41	**38.4**	**65.93**	**58.02**
	RPTQ(W4A4)	64.93	**75.46**	**59.89**	**35.67**	37.40	**66.12**	**56.58**
	Ours(W4A4)	**66.54**	74.76	57.45	34.64	**37.80**	63.15	55.72

* The quantization settings for RPTQ and ours are the same as those used in Table I. he quantization settings for GPTQ are the same as those used in Table II.

TABLE IV

AVERAGE PERPLEXITY (↓) ACHIEVED BY OUR PROPOSED QUANTIZATION SCHEME WITH CHANNEL SORTING OR/AND CHANNEL SELECTION

Approach		Grouping	Grouping + Sorting	Grouping + Selection	Grouping + Both
OPT-1.3b	W4A8	16.54	16.23	16.35	16.21
	W4A4	19.57	19.49	19.00	19.01
OPT-6.7b	W4A8	12.55	12.40	12.38	12.35
	W4A4	13.96	13.66	13.31	13.26
OPT-13b	W4A8	12.09	11.81	11.82	11.82
	W4A4	15.07	15.47	13.77	13.93

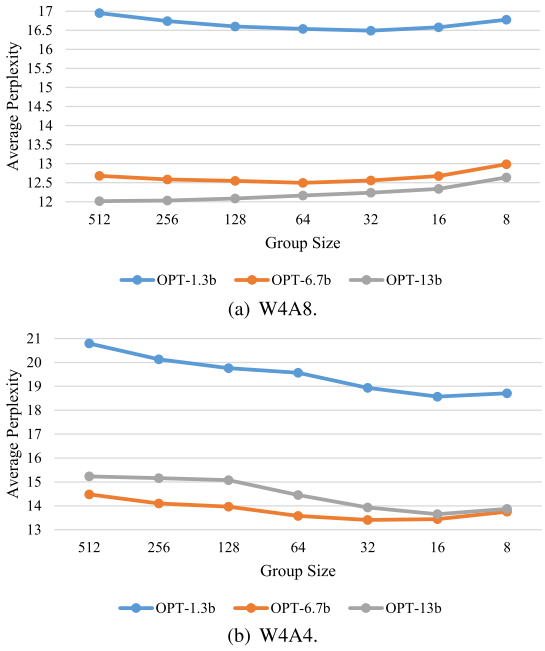
* Except for the cases in which channel selection is applied to OPT-13b, the other quantization settings are the same as those used in Table I.

TABLE V

AVERAGE PERPLEXITY (↓) ACHIEVED BY OUR PROPOSED QUANTIZATION SCHEME WHEN DIFFERENT NUMBERS OF SELECTED CHANNELS ARE USED IN THE TOP-K ALGORITHM

K		1	2	4	8
OPT-1.3b	W4A8	16.49	16.21	16.20	16.21
	W4A4	19.29	19.27	19.16	19.01
OPT-6.7b	W4A8	12.38	12.37	12.35	12.35
	W4A4	13.39	13.37	13.32	13.26
OPT-13b	W4A8	11.82	11.82	11.85	11.82
	W4A4	13.92	13.86	14.02	13.77

* Except for K, the other quantization settings are the same as those used in Table I.



(a) W4A8.

(b) W4A4.

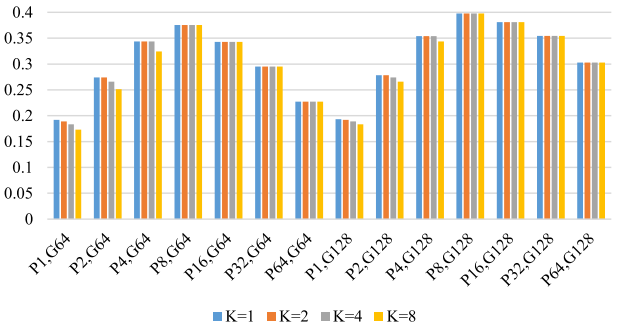
Fig. 10. Average perplexity for different group sizes.

smaller group sizes. In conclusion, we recommend using group sizes of at least 64.

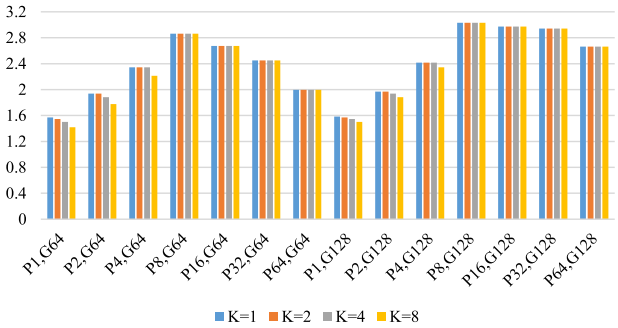
C. Hardware Implementation

As we have discussed in Section IV, a larger P_{entry} leads to a larger adder tree, which results in a higher area cost and a longer critical path. On the other hand, it also increases the throughput of PE if the clock frequency is fixed. Even if we set the clock frequency to the maximum achievable value, the reduction on clock frequency is less significant than the increase on the degree of parallelism, leading to a higher throughput. Area efficiency is a reasonable metric for simultaneously evaluating both area cost and throughput. Meanwhile, energy efficiency is a key metric that simultaneously measures both power consumption and throughput. We evaluated the area efficiency and energy efficiency of PE-A for different values of P_{entry}, K and $|G|$. The results are illustrated in Fig. 11. It shows that $P_{entry} = 8$ is the best option in terms of both area efficiency and energy efficiency. For PE-B, we also give similar evaluation. Unlike PE-A, area efficiency and energy efficiency of PE-B will not change with the group size $|G|$ nor the number of selected channels K, as long as the architecture is fixed (i.e., with fixed P_{entry} and k). Table VI lists the results for PE-B with four different parameter settings. We can see that $(P_{entry}, k) = (64, 8)$ is the best parameter setting. Consider two MVM architectures dubbed Arc-A and Arc-B that employ PE-A and PE-B as their PE, respectively. Parameter settings for the two architectures are given in Table VII. As such, the two architectures have the same number of MUs, i.e., the same overall degree of parallelism.

OliVe [25] is a recently proposed quantization and architecture co-design for LLM accelerators. It prunes normal values



(a) Area efficiency.



(b) Energy efficiency.

Fig. 11. Area efficiency ($TOPS/mm^2$) and energy efficiency ($TOPS/W$) of PE-A with different P_{entry}, K and $|G|$. The clock period is set to 5 ns.

TABLE VI

AREA EFFICIENCY ($TOPS/mm^2$) AND ENERGY EFFICIENCY ($TOPS/W$) OF PE-B WITH DIFFERENT P_{entry} AND k

(P_{entry}, k)	(8,1)	(16,2)	(32,4)	(64,8)
Area Eff.	0.39	0.40	0.39	0.40
Energy Eff.	4.21	4.51	4.44	4.62

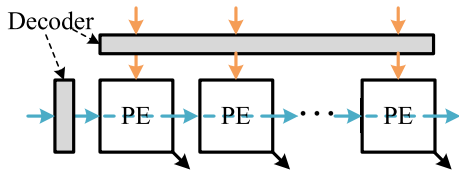
* The clock period is set to 5 ns.

TABLE VII

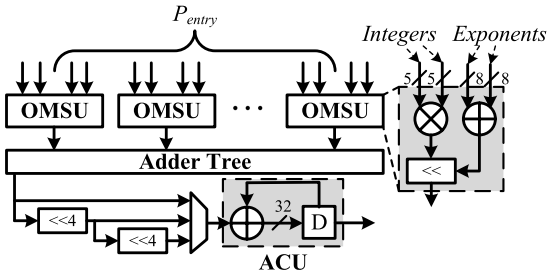
PARAMETER SETTINGS FOR ARC-A AND ARC-B

Parameter	$P_{channel}$	P_{group}	P_{entry}	k
Arc-A	64	8	8	-
Arc-B	64	1	64	8

that are adjacent to the outliers. The pruned normal values provide extra space for embedding the outliers into a low-precision matrix. To cover the broad data range of outliers, a data type called adaptive biased float (abfloat) is devised in OliVe. Activations and weights will be quantized into the abfloat format in software and the quantized values (in the abfloat format) are processed by the hardware accelerator. The original design of OliVe adopts a systolic array to accelerate matrix-matrix multiplications. Nevertheless, as we analyzed earlier, a systolic array for matrix-matrix multiplications suffers from extremely low hardware utilization in the token generation stage. We hence adapt the systolic array of OliVe into a $P_{group} \times P_{channel}$ PE array that focuses on accelerating MVMs, which is the same as the architecture shown in Fig. 6. As shown in Fig. 12(a), each row of PEs is assigned with two decoders which aim to decode the data in the abfloat format into exponent-integer pairs. Each PE of the adapted OliVe consists of P_{entry} OliVe multiply-shift units (OMSU), as shown in Fig. 12(b). The inputs of an OMSU are two exponent-integer pairs, corresponding to an activation-weight pair. Namely, both the input activation and weight are represented by an exponent



(a) Deployment of decoders in each row of PEs.



(b) Architecture of OliVe PE.

Fig. 12. Details of the MVM architecture adapted from OliVe [25]. The top-level architecture is also a $P_{group} \times P_{channel}$ PE array, which is the same as that of Fig. 6.

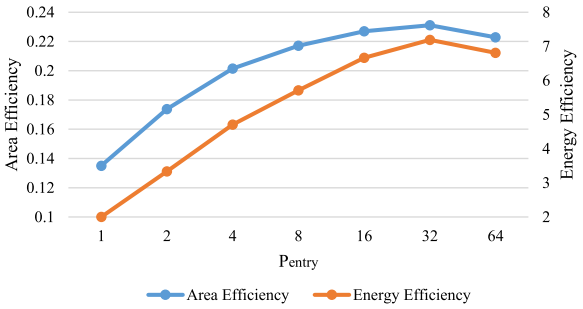


Fig. 13. Area efficiency ($TOPS/mm^2$) and energy efficiency ($TOPS/W$) of OliVe [25] PE with different values of P_{entry}. The clock period is set to 5 ns.

and an integer. Product of the two 5-bit integers will be left-shifted by the number of bits specified by the sum of the exponents. Since the experiments of [25] show that the magnitudes of outlier values are much smaller than 2^{15}, we set the bitwidth of the outputs of OMSUs to 15 bits. For more details about OliVe, please refer to [25]. We attempt to make a fair comparison of hardware performance with OliVe, rather than a point-to-point comparison of its technical details.

We evaluated the area efficiency and energy efficiency of OliVe PE with different values of P_{entry}. The results are depicted in Fig. 13, indicating that $P_{entry} = 32$ is the most area- and energy-efficient option. Consequently, we consider the MVM architecture which is adapted from OliVe and has $P_{channel} = 64$, $P_{group} = 2$ and $P_{entry} = 32$ as the reference for comparison. Table VIII compares it with our proposed architectures. As reported in [25], OliVe successfully quantizes LLMs with negligible performance loss when the bit configuration is W8A8. However, it results in unacceptable performance degradation in the case of W4A4. Since our proposed quantization scheme focuses on low-bitwidth quantization, here we consider the W4A4 case. Clearly, our proposed quantization scheme achieves better model performance than that of OliVe. When the clock frequency is set to 100 MHz, Arc-A and Arc-B both surpass the MVM architecture of OliVe in terms of area cost, area efficiency and energy efficiency. While Arc-B has the same throughput as OliVe, Arc-A is

TABLE VIII

COMPARISON OF THE PROPOSED MVM ARCHITECTURES AND THAT OF THE STATE-OF-THE-ART DESIGN OLIVE

Design		OliVe [25]		Arc-A (proposed)		Arc-B (proposed)	
Clock Frequency (MHz)		100	172*	100	**370***	100	263*
Throughput** ($TOPS$, 4b×4b)		0.41	0.71	0.39	**1.43**	0.41	1.08
Area (mm^2)		4.51	4.88	1.92	2.91	**1.84**	2.72
Number of Gates (M)***		2.82	3.05	1.2	1.82	**1.15**	1.70
Power (mW)		157.50	280.82	87.04	471.04	**78.08**	295.68
Area Efficiency ($TOPS/mm^2$)		0.09	0.15	0.20	**0.49**	0.22	0.40
Energy Efficiency ($TOPS/W$)		2.60	2.53	4.23	3.03	**5.25**	3.65
Performance Loss (W4A4) ****	WIKI	33.30		**1.23**			
	C4	21.78		**1.30**			

* Maximum achievable clock frequency.
** Efficient throughput, which takes into account the number of the additional multiplications introduced by our proposed channel selection approach. For Arc-A, extra clock cycles are required to handle the high-order activation bits in the selected channel, leading to a decrease in throughput. The throughput of Arc-A is evaluated for OPT-6.7b, the parameters of which are the same as those in Table I.
*** The number of gates is obtained by dividing the overall area by that of a NAND gate.
**** To be consistent with [25], W4A4 for OPT-6.7b is considered. Performance loss refers to the increased perplexity when compared with the floating-point baselines. The results for our design is derived from Table I while those for OliVe are from [25].

TABLE IX

PARAMETER SETTINGS FOR ARC-RPTQ, ARC-A′ AND ARC-B′

Parameter	$P_{channel}$	P_{group}	P_{entry}	k
Arc-RPTQ	64	1	1	-
Arc-A′	1	8	8	-
Arc-B′	1	1	64	8

inferior due to the additional cycles introduced to process the high-order bits of activations. As Arc-A enjoys the shortest critical path, its clock frequency can achieve up to 370 MHz, leading to a throughput of 1.35 $TOPS$. Generally, with the same overall degree of parallelism, higher clock frequency brings better throughput and area efficiency but the resulting energy efficiency becomes worse. Overall, Arc-B can achieve the same throughput, 2.44× area efficiency, and 2.02× energy efficiency of the MVM architecture of OliVe, with the same clock frequency of 100 MHz. When operating at the maximum allowed clock frequency, Arc-A achieves 2.01× throughput, 3.27× area efficiency, and 1.20× energy efficiency of the MVM architecture of OliVe.

We also compare with RPTQ from the hardware perspective. The top-level architecture for RPTQ is the same as that shown in Fig. 6. We employ PE-A for RPTQ except that the bottom shifter is removed since there is no activation with more than eight bits in RPTQ. To avoid the workload imbalance we have analyzed before, we set $P_{group} = 1$ and $P_{entry} = 1$ for RPTQ. As for our proposed architectures, we still consider $P_{entry} = 8$ for PE-A and $P_{entry} = 64$ for PE-B. Denote the three MVM architectures by Arc-RPTQ, Arc-A′ and Arc-B′, whose parameter settings are listed in Table IX. Their implementation results are given in Table X. As expected, our proposed architectures perform better in terms of area efficiency and energy efficiency, when cooperating with the proposed quantization scheme. Arc-RPTQ can achieve higher clock frequency, due to its small P_{entry}. Accordingly, it has better throughput than ours. Unfortunately, this comes at the expense of higher area cost and power consumption.

Without support for the proposed quantization scheme, we can employ the basic PE architecture, which is the same as PE-A except that the bottom shifter of Fig. 8(a) is removed.

TABLE X

COMPARISON WITH RPTQ FROM THE HARDWARE PERSPECTIVE

Design	Arc-RPTQ		Arc-A′ (proposed)		Arc-B′ (proposed)	
Clock Frequency (MHz)	100	**588***	100	370*	100	263*
Throughput** ($GOPS$, 4b×4b)	6.40	**37.63**	6.02	22.29	6.40	16.83
Area (um^2)	63360.0	108288.0	30297.6	45424.0	**28806.0**	42442.8
Number of Gates (K)	39.6	67.7	18.9	28.4	**18.0**	26.5
Power (mW)	3.28	24.56	1.36	7.36	**1.22**	4.62
Area Efficiency ($TOPS/mm^2$)	0.10	0.35	0.20	**0.49**	0.22	0.40
Energy Efficiency ($TOPS/W$)	1.95	1.53	4.23	3.03	**5.25**	3.65

* Maximum achievable clock frequency.
** To evaluate throughput of Arc-A, we still consider the OPT-6.7b used in Table I.

TABLE XI

HARDWARE COMPARISON OF THE PE-BASIC, PE-A AND PE-B

PE Type	PE-basic		PE-A	PE-B
(P_{entry}, k)	(8,0)	(64,0)	(8,0)	(64,8)
Area (um^2)	3707.6	28161.6	3787.2	32065.6
Power (mW)	0.34	2.45	0.35	2.77

* The clock period is set to 5ns.

We denote the basic PE architecture by PE-basic. Table XI presents hardware comparison of PE-basic, PE-A and PE-B. The configurations for PE-A and PE-B are the same as those adopted in previous experiments, i.e., $(P_{entry}, k) = (8, 0)$ and $(P_{entry}, k) = (64, 8)$, respectively. As the counterpart, PE-basic is configured with the same values of P_{entry}. As demonstrated by Table XI, PE-A has similar hardware performance to PE-basic, since it only introduces an extra shifter. Although PE-A supports our proposed quantization scheme at the expense of extra clock cycles, the degradation on throughput is still limited (see Table. VIII and Table X) because the number of selected channels is much smaller than the group size. On the other hand, PE-B employs additional MSUs to handle higher-order bits for activations in the outlier channels, resulting in 13.9% additional area overhead when compared with PE-basic.

Now we briefly analyze the storage overhead. Consider a practical setting in which $K = 8$ and $|G| = 128$ (see Table I). The storage overhead introduced by the higher-order bits of activations is only $K/|G| = 6.25\%$. Recall the fact that with small batch size and practical sequence length, weights of LLMs dominate the memory requirement. Compared to the overall storage overhead, the increase becomes less significant. The fundamental purpose of channel sorting is to assign appropriate quantization parameters to different channels. In fact, channels are sorted only when computing the quantization parameters. During inference, we only need to store group index for each channel, indicating which group it belongs to. Then the weights and activations of each channel can be quantized using the quantization parameters of corresponding group. For each channel, the number of 4-bit weights is at least D (see Section II-A). The number of bits used to store group index is $\lceil log_2 N_{group} \rceil$. Thus, the storage overhead introduced by channel sorting is roughly $\frac{\lceil log_2 N_{group} \rceil}{4D}$ which is negligible (e.g. 0.1% for OPT-6.7b of Table I). In conclusion, the additional storage overhead brought by our proposed design is limited.

D. Other Related Works on LLM Quantization

While our paper was under review, several new works on LLM quantization emerged. In this subsection, we present a brief qualitative comparison with these works. QLLM [58] introduces an adaptive channel reassembly technique to mitigate the impact caused by outliers. It leads to some additional computations that require to be performed on the fly, such as decomposing the input outlier channels into several sub-channels and merging some similar channels by computing certain channel distances. Moreover, training is invoked to compensate for the performance loss brought by quantization. QUIK [59] is a quantization scheme similar to LLM.int8(), as it quantizes the outliers using 16-bit floating-point numbers. The quantization scheme proposed in SmoothQuant is improved in [60], by means of so-called activation-quantization-aware scaling and sequence-length-aware calibration. However, this work only considers the bitwidth configuration of W4A8 and it introduces specifically designed data format for weights. Unlike integer quantization, such an unconventional data format requires support from specialized arithmetic units which are not compatible with general fixed-point operations. OmniQuant [61] tackles activation outliers by shifting the challenge of quantization from activations to weights through a learnable equivalent transformation. It also operates within a differentiable framework using block-wise error minimization. OmniQuant can achieve better quantization performance than RPTQ at the cost of a certain degree of training.

In summary, among existing PTQ schemes that quantize both weights and activations into integers with low bitwidths, our proposed one achieves the best performance. It can serve as an excellent plug-and-play quantization solution and is also hardware-friendly. Certainly, it is viable to compensate for quantization error through training. However, it will not be discussed here as it is beyond the scope of this paper.

VI. Conclusion

This paper presents a quantization and hardware architecture co-design for the MVMs of LLMs. To address the workload imbalance issue caused by the state-of-the-art quantization scheme RPTQ, we propose to uniformly group the channels. Furthermore, two approaches named channel sorting and channel selection are devised to enhance the quantization performance. To support the proposed quantization scheme, two MVM architectures are developed, which are targeted at high speed and high energy efficiency, respectively. We use experimental results to demonstrate that the proposed quantization scheme can achieve better performance than RPTQ. Moreover, it is shown that the proposed MVM architectures outperform that of OliVe in area efficiency and energy efficiency.

References

[1] J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.

[2] J. Wei et al., "Emergent abilities of large language models," 2022, *arXiv:2206.07682*.

[3] R. Schaeffer, B. Miranda, and S. Koyejo, "Are emergent abilities of large language models a mirage?" 2023, *arXiv:2304.15004*.

[4] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.

[5] Y. Chang et al., "A survey on evaluation of large language models," 2023, *arXiv:2307.03109*.

[6] J. Yang et al., "Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond," 2023, *arXiv:2304.13712*.

[7] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[8] OpenAI, "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[9] S. Zhang et al., "OPT: Open pre-trained transformer language models," 2022, *arXiv:2205.01068*.

[10] H. Touvron et al., "LLaMA: Open and efficient foundation language models," 2023, *arXiv:2302.13971*.

[11] H. Touvron et al., "LLaMA 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[12] *OpenAI*. Accessed: Nov. 2022. [Online]. Available: https://openai.com/blog/chatgpt/

[13] S. Bubeck et al., "Sparks of artificial general intelligence: Early experiments with GPT-4," 2023, *arXiv:2303.12712*.

[14] B. Workshop et al., "BLOOM: A 176B-parameter open-access multilingual language model," 2022, *arXiv:2211.05100*.

[15] A. Zeng et al., "GLM-130B: An open bilingual pre-trained model," 2022, *arXiv:2210.02414*.

[16] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.

[17] Y. Sheng et al., "FlexGen: High-throughput generative inference of large language models with a single GPU," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 31094–31116.

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

[19] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, "OWQ: Lessons learned from activation outliers for weight quantization in large language models," 2023, *arXiv:2306.02272*.

[20] J. Lin et al., "AWQ: Activation-aware weight quantization for LLM compression and acceleration," 2023, *arXiv:2306.00978*.

[21] T. Dettmers et al., "SpQR: A sparse-quantized representation for near-lossless LLM weight compression," 2023, *arXiv:2306.03078*.

[22] S. Kim et al., "SqueezeLLM: Dense-and-sparse quantization," 2023, *arXiv:2306.07629*.

[23] T. Dettmers et al., "LLM.int8(): 8-bit matrix multiplication for transformers at scale," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 30318–30332.

[24] G. Xiao et al., "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 38087–38099.

[25] C. Guo et al., "OliVe: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, Jun. 2023, pp. 1–15.

[26] X. Wei et al., "Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling," 2023, *arXiv:2304.09145*.

[27] Z. Yuan et al., "RPTQ: Reorder-based post-training quantization for large language models," 2023, *arXiv:2304.01089*.

[28] A. Gholami et al., "A survey of quantization methods for efficient neural network inference," *Low-Power Computer Vision*. Boca Raton, FL, USA: CRC Press, 2022, pp. 291–326.

[29] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. and Stat.*, 2011, pp. 315–323.

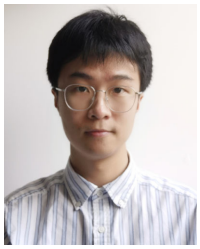
[32] N. Shazeer, "GLU variants improve transformer," 2020, *arXiv:2002.05202*.

[33] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[34] H. Sharma et al., "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 764–775.

[35] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[36] S. Ryu et al., "BitBlade: Energy-efficient variable bit-precision hardware accelerator for quantized neural networks," *IEEE J. Solid-State Circuits*, vol. 57, no. 6, pp. 1924–1935, Jun. 2022.
[37] J. Yang et al., "GQNA: Generic quantized DNN accelerator with weight-repetition-aware activation aggregating," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 10, pp. 4069–4082, Oct. 2022.
[38] W. Li, A. Hu, N. Xu, and G. He, "A precision-scalable deep neural network accelerator with activation sparsity exploitation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 1, pp. 263–276, Jan. 2024.
[39] L. Lu et al., "Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 977–991.
[40] T. Yang et al., "DTATrans: Leveraging dynamic token-based quantization with accuracy compensation mechanism for efficient transformer architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 2, pp. 509–520, Feb. 2023.
[41] Z. Zhou, J. Liu, Z. Gu, and G. Sun, "Energon: Toward efficient acceleration of transformers using dynamic sparse attention," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 1, pp. 136–149, Jan. 2023.
[42] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 97–110.
[43] J. Dass et al., "ViTALiTy: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear Taylor attention," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2023, pp. 415–428.
[44] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Jan. 2022.
[45] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–28, Dec. 2022.
[46] K. Han et al., "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, Jan. 2023.
[47] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: K-bit inference scaling laws," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 7750–7774.
[48] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[49] Y. Kim et al., "Winning both the accuracy of floating point activation and the simplicity of integer arithmetic," in *Proc. Int. Conf. Learn. Represent.*, 2023.
[50] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv:1609.07843*.
[51] M. Marcus et al., "The Penn treebank: Annotating predicate argument structure," in *Proc. Workshop Hum. Lang. Technol.*, 1994, pp. 114–119.
[52] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
[53] D. Paperno et al., "The LAMBADA dataset: Word prediction requiring a broad discourse context," 2016, *arXiv:1606.06031*.
[54] S. Tata and J. M. Patel, "PiQA: An algebra for querying protein data sets," in *Proc. 15th Int. Conf. Sci. Stat. Database Manage.*, 2003, pp. 141–150.
[55] P. Clark et al., "Think you have solved question answering? Try ARC, the AI2 reasoning challenge," 2018, *arXiv:1803.05457*.
[56] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," 2018, *arXiv:1809.02789*.
[57] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "BoolQ: Exploring the surprising difficulty of natural yes/no questions," 2019, *arXiv:1905.10044*.
[58] J. Liu, R. Gong, X. Wei, Z. Dong, J. Cai, and B. Zhuang, "QLLM: Accurate and efficient low-bitwidth quantization for large language models," 2023, *arXiv:2310.08041*.
[59] S. Ashkboos et al., "QUIK: Towards end-to-end 4-bit inference on generative large language models," 2023, *arXiv:2310.09259*.
[60] J. Lee, M. Kim, S. Baek, S. J. Hwang, W. Sung, and J. Choi, "Enhancing computation efficiency in large language models through weight and activation quantization," 2023, *arXiv:2311.05161*.
[61] W. Shao et al., "OmniQuant: Omnidirectionally calibrated quantization for large language models," 2023, *arXiv:2308.13137*.



Wenjie Li received the B.E. and M.S. degrees from Nanjing University, Nanjing, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China.

He has published more than ten peer-reviewed articles. His research interests include error correction codes, stochastic computing, and VLSI design for deep learning.

Aokun Hu received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2021, where he is currently pursuing the M.S. degree with the School of Electronic, Information and Electrical Engineering.

His current research interests include stochastic computing and VLSI design for deep learning.

Ningyi Xu received the B.S. and Ph.D. degrees from Tsinghua University in 2001 and 2006, respectively.

He was a Principle Architect with Baidu and a Lead Researcher with the Hardware Computing Group, Microsoft Research Asia, Beijing. He is currently a Professor with the Qingyuan Research Institute, Shanghai Jiao Tong University, Shanghai, China. His current research interests include domain specific computing, computer architecture, parallel computing, and machine learning systems.

Guanghui He (Member, IEEE) received the B.S. degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2002, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2007.

He is currently a Professor with the Department of Micro/Nano Electronics and the MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University. His research interests include energy efficient algorithms and circuits design for wireless communication, image processing, emerging memory devices, and artificial intelligent systems.