

SA-ANT: Efficient Low-Bit Group-Wise Quantization for Large Language Models via Sign-Asymmetric Adaptive Numeric Type

Xinkuang Geng*, Siting Liu[†], Hui Wang*, Jie Han[‡], Honglan Jiang*

* School of Integrated Circuits, Shanghai Jiao Tong University, Shanghai, China

[†] School of Information Science and Technology, ShanghaiTech University, Shanghai, China

[‡] Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada

xinkuang@sjtu.edu.cn, liust@shanghaitech.edu.cn, ihuiwang@sjtu.edu.cn, jhan8@ualberta.ca, honglan@sjtu.edu.cn

Abstract—Large language models (LLMs) have demonstrated remarkable potential across diverse domains; meanwhile, their large parameter sizes pose substantial inference costs, motivating the need for efficient low-bit quantization. Group-wise quantization, which adopts finer granularity, has been widely used to improve low-bit quantization performance. Several adaptive numeric types have been proposed to further enhance low-bit group-wise quantization; however, they construct quantization grids based on symmetric numeric types, which limits their ability to model asymmetric distributions. To address this limitation, we propose SA-ANT, a sign-asymmetric adaptive numeric type for efficient low-bit group-wise quantization. SA-ANT constructs quantization grids separately on the positive and negative sides, enabling adaptive support for asymmetric and non-uniform distributions. Furthermore, the carefully designed SA-ANT not only reduces quantization errors but also ensures a unified computing across different sub numeric types, thereby facilitating hardware efficiency. To accelerate LLM inference, we develop (1) a quantization framework that transforms LLM weights into the SA-ANT and adaptively selects the sub numeric type for each group, and (2) an accelerator that maps SA-ANT inference to low-bit INT operations. Experimental results show that SA-ANT delivers 3.92%–5.57% higher accuracy than state-of-the-art adaptive numeric types under 3-bit weight quantization, while also enabling 7.84%–44.65% area savings and 7.80%–43.88% power reductions.

Index Terms—large language model, low-bit quantization, group-wise quantization, adaptive numeric type, accelerator

I. INTRODUCTION

Large language models (LLMs) have achieved remarkable success in various natural language processing tasks [1]–[4]. However, the growing demand for broader world knowledge and stronger reasoning capabilities has driven a continuous increase in model size, leading to substantial memory and computing costs [5]–[7]. As an efficient technique for parameter compression and computing simplification, quantization discretizes data using a grid with fewer possible values, enabling low-bit representations of data in LLMs and reducing the deployment overhead [8]–[10]. Group-wise quantization can effectively reduce quantization errors through finer granularity, and has emerged as an important research direction and practical deployment strategy under low-bit budget [11]–[15].

This work was supported in part by the National Natural Science Foundation of China under grant numbers 62374108 and 62204155, and in part by Alberta Innovates under grant number RES0053965.

Conventional integer (INT)- and floating-point (FP)-based numeric types have achieved usable accuracy in LLM inference under 4-bit quantization, and have therefore been widely adopted in real-world applications [12], [13], [16], [17]. To further reduce quantization errors, emerging studies have explored adaptive numeric types, such as ANT [18], Olive [19], M-ANT [14], and BitMod [15], demonstrating a strong potential. A key challenge for adaptive numeric types is to design sub numeric types that align well with diverse data distributions, while still enabling unified and efficient computing. Existing methods mainly focus on adaptively matching the non-uniform distributions [14], [15], [18], [20], [21] but insufficiently model the asymmetry observed in group-wise quantization.

To address this limitation, we propose SA-ANT, a sign-asymmetric adaptive numeric type for low-bit group-wise quantization of LLMs. The main contributions of this work are summarized as follows.

- We analyze and characterize new distributional properties of group-wise data in LLMs.
- SA-ANT is proposed to accommodate diverse data distributions with asymmetric and non-uniform characteristics.
- We propose an offline adaptive quantization scheme tailored to SA-ANT, resulting in higher zero-shot accuracy than state-of-the-art 3-bit weight quantization methods.
- An accelerator is designed for SA-ANT based on the customized decoders. With a higher accuracy, it consumes a lower area and power than existing low-bit accelerators.

The rest of this paper is organized as follows. Section II reviews the background of quantization and adaptive numeric types. Section III discusses the motivation, construction process, and adaptive quantization scheme of SA-ANT. In Section IV, an accelerator tailored for SA-ANT is devised. Section V evaluates the performance of SA-ANT in LLM weight quantization and the hardware overhead of the accelerator. Finally, Section VI concludes the paper.

II. BACKGROUND

A. Quantization Basics

The quantization process based on different numeric types can be formulated as approximating the original data W_f

by searching for the closest element from a given discrete quantization grid G (the value set of a numeric type) as

$$W_q = q(W_f, G) = \arg \min_{W_i \in G} |W_i - \frac{W_f}{\Delta}|, \quad (1)$$

where the scaling factor Δ scales W_f into the same range as G , and W_q is the quantized data consisting of values from G . Subsequently, the dequantization step rescales W_q back to the range of original data as

$$W_{qf} = \Delta \cdot W_q \approx W_f. \quad (2)$$

In general, since Δ is shared across all elements in W_q , dequantization can be deferred until the multiply-accumulate operations for W_q are completed. This allows the core computations to be performed in the domain of the quantization grid G , avoiding high-precision floating-point operations.

In tensor-wise quantization, a single Δ is applied to the entire tensor (matrix), with its value determined by the global extrema. In contrast, channel-wise quantization assigns the same Δ to each channel (row vector) [22]. Group-wise quantization further refines the granularity by partitioning each channel into smaller contiguous groups, each assigned its own Δ [13]. This localizes the impact of outliers to the group they belong to, leaving other groups unaffected.

B. Weight Quantization for Large Language Models

For LLM inference on edge devices with relatively small batch sizes, the memory footprint is typically dominated by the weights [14], [15]. As shown in Fig. 1, we visualize how memory is allocated among weights, activations, and KV cache in Llama-3.1-8B [4] across two representative tasks. It can be observed that with a batch size of 32 (i.e., 32 input sequences processed simultaneously), the weights in Transformer blocks occupy the majority of memory. When the batch size decreases, only activations and KV cache shrink, while weights remain unchanged. Thus, applying low-bit quantization to weights is an effective way to reduce the overall memory in LLM inference.

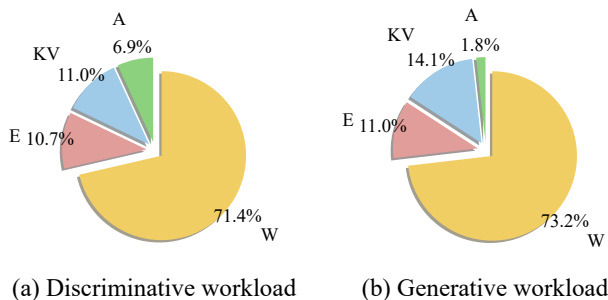


Fig. 1: Memory ratio of activations (A), KV cache (KV), embeddings (E), and weights (W) in Llama-3.2-8B (the batch size is 32) under discriminative (512 input & 1 output tokens) and generative (128 input & 512 output tokens) workloads.

In addition, compared to activation quantization that must be executed online, weight quantization can be performed offline; thus, it offers greater optimization flexibility with minimal impact on inference efficiency. As a result, many recent studies have focused on weight-only quantization [13]–[15], typically compressing weights to 3- or 4-bit while keeping activations in FP16 or INT8.

C. Adaptive Numeric Type

The use of adaptive numeric types as quantization grids has been extensively studied [14], [15], [18], [19]. Adaptive numeric types dynamically select the sub numeric type as per the data distribution characteristics.

ANT [18] employs an adaptive numeric type by assigning appropriate sub numeric types (e.g., INT, FP, power-of-two (PoT)) to the data to be quantized, guided by minimizing the mean squared error (MSE). Olive [19] introduces the concept of outlier-victim pairs and leverages the redundant encoding to identify outliers and then assign them a special sub numeric type with a larger dynamic range. M-ANT [14] derives 16 sub numeric types by mathematically combining INT and PoT quantization grids in different proportions, and selects the most suitable one for each group according to its variance. BitMod [15] modifies FP by exploiting the redundant zero encoding to construct additional adaptive quantization points, enabling dynamic adjustment of the quantization resolution or range. In addition, INT-Asym (asymmetric uniform quantization) can be regarded as a form of adaptive numeric type. Although INT-Asym preserves a uniform quantization resolution, the use of zero points enables the adaptation to asymmetric ranges within each group.

III. SIGN-ASYMMETRIC ADAPTIVE NUMERIC TYPE

A. Motivation

As shown in Fig. 2, we visualize the data distributions of 16 channels and 16 groups from the weights of Llama-3.2-1B [4]. Although the ranges of different channels vary a lot, their distributions are all approximately Gaussian with a mean of zero. Therefore, once the range differences are normalized by channel-wise scaling, it is reasonable to quantize all channels using a single non-uniform numeric type. In contrast, the situation changes under fine-grained group-wise quantization, where smaller groups containing fewer elements lead to reduced statistical stability, i.e., their data distributions deviate from a consistent Gaussian shape. As a result, even scaling different groups to the same range, the obtained data distributions across groups still exhibit significant differences.

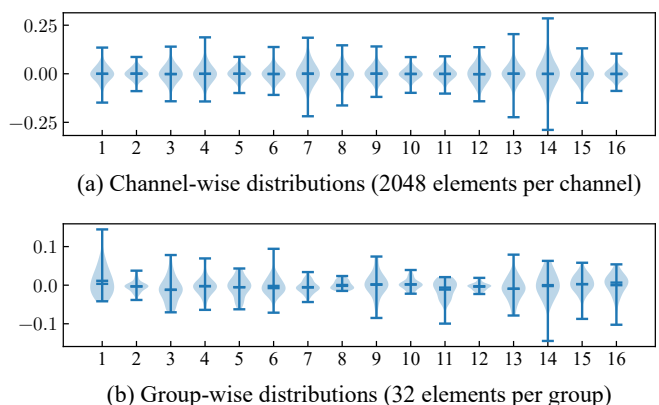


Fig. 2: Data distributions of the sampled 16 channels (a) and 16 groups (b) from the first projection weight matrix in the first Transformer block of Llama-3.2-1B.

We observe that the distribution of group-wise data in LLMs differs significantly from those at the channel or tensor level, characterized by three properties, i.e.,

- (1) asymmetric data ranges,
- (2) asymmetric non-uniformity, and
- (3) the majority of elements concentrated around zero.

We refer to such distributions as skewed normal distributions. For different skewed normal distributions, existing adaptive numeric types fail to effectively align with their properties. As shown in Fig. 3, we highlight this limitation by showing the quantization grids and corresponding quantization errors in MSE for one group from Llama-3.2-1B under various 3-bit numeric types.

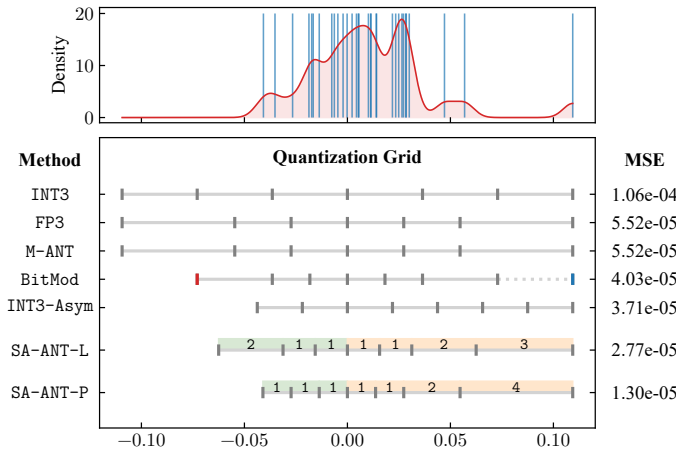


Fig. 3: Data distribution, quantization grids and errors of different numeric types for one group from the first projection weight matrix in the first Transformer block of Llama-3.2-1B. Blue vertical lines denote the elements within this group.

INT3 fails to align with the asymmetry and non-uniformity of the data, resulting in the highest MSE. In contrast, FP3 provides finer resolution for values near zero, thereby introducing appropriate non-uniformity and reducing the MSE. Although M-ANT provides 16 strictly symmetric sub numeric types designed for data with different variances, in this case it selects the same quantization grid as FP3 and fails to match the asymmetric distribution. BitMod preserves the original FP3 grid with possible scaling, while exploiting the redundant zero encoding to introduce an additional quantization point, thereby incorporating a degree of asymmetry. For the specific data shown in Fig. 3, BitMod adaptively adds the blue quantization point to FP3 and achieves lower MSE compared to plain FP3. As an FP3 variant, BitMod keeps mostly symmetric quantization points, leaving some points unused for skewed normal distributions (e.g., the red point in Fig. 3). INT3-Asym employs a zero point to offset the INT3 grid and thus producing asymmetry. However, INT3-Asym provides no adaptation to the non-uniformity of the data.

Moreover, existing non-uniform numeric types cannot gain support for asymmetry simply by introducing a zero point. This is because their dense regions are deliberately concentrated around zero to match normal or skewed normal distributions.

Offsetting the grid with a zero point to induce asymmetry would inevitably disrupt this non-uniform alignment.

To address the limitation, we propose a sign-asymmetric adaptive numeric type, SA-ANT, which aligns with both the asymmetry and non-uniformity of group-wise data while still enabling unified and efficient computing across different sub numeric types. As shown in Fig. 3, both SA-ANT-L and SA-ANT-P, tailored for different hardware budgets, achieve substantially lower quantization errors compared with existing numeric types.

B. SA-ANT Construction

This section demonstrates how SA-ANT is constructed through combinations of non-decreasing sequences (NDSs) to well align with the properties of skewed normal distributions.

SA-ANT aims to guarantee the performance of 3-bit weight quantization. The 3-bit quantization grid G can contain at most eight quantization points. Since zero always occupies one point, seven points remain to be arranged. We propose to adaptively allocate three points to one side (positive or negative) and four points to the other to constructing sub numeric types. We refer to the side with three points as Side-3 and the side with four points as Side-4. By treating zero as a starting point and using the NDSs in Table I as spacings between adjacent points, we can construct different half grids that vary in range and non-uniformity while always remaining denser grids around zero. These half grids are then combined to form SA-ANT.

TABLE I: Non-decreasing sequences used in SA-ANT.

Method	Side	Non-decreasing sequences
SA-ANT-L	3	$1, 1, 2$; $1, 2, 3$
	4	$1, 1, 1, 1$; $1, 1, 1, 2$; $1, 1, 2, 2$; $1, 1, 2, 3$
SA-ANT-P	3	$1, 1, 1$; $1, 1, 2$; $1, 2, 2$; $1, 2, 4$
	4	$1, 1, 1, 1$; $1, 1, 1, 2$; $1, 1, 2, 2$; $1, 1, 2, 4$; $1, 2, 2, 2$; $1, 2, 2, 4$; $1, 2, 4, 4$; $1, 4, 4, 4$

We take SA-ANT-L that contains 16 sub numeric types as an example to demonstrate the construction process. As shown in Table I, the Side-3 of SA-ANT-L consists of two NDSs, while Side-4 consists of four NDSs. The initial half grids are constructed using these NDSs as spacings, as shown in Fig. 4 (a). For example, the sequence $\{1, 2, 3\}$ in Side-3 produces the half grid $\{0, 1, 3, 6\}$ and the sequence $\{1, 1, 2, 3\}$ in Side-4 produces the half grid $\{0, 1, 2, 4, 7\}$. From this step, it is evident that the half grids constructed based on NDSs naturally align with **property (3)** of skewed normal distributions. We denote these half grids as $T_{1:n_T}$ and $Q_{1:n_Q}$, corresponding to Side-3 and Side-4, respectively. As shown in Fig. 4 (a), SA-ANT-L has two half grids in $T_{1:2}$ and four half grids in $Q_{1:4}$. Next, by combining $-T_{1:n_T}$ with $Q_{1:n_Q}$ pairwise, we obtain the full grids where Side-3 forms the negative part and Side-4 forms the positive part, as shown in Fig. 4 (b). Similarly, by combining $-Q_{1:n_Q}$ with $T_{1:n_T}$ pairwise, we obtain the complementary case, as shown in Fig. 4 (c). This construction process can be expressed as

$$G_{SA} = \bigcup_{i=1}^{n_T} \bigcup_{j=1}^{n_Q} \{-T_i \cup Q_j, -Q_j \cup T_i\}. \quad (3)$$

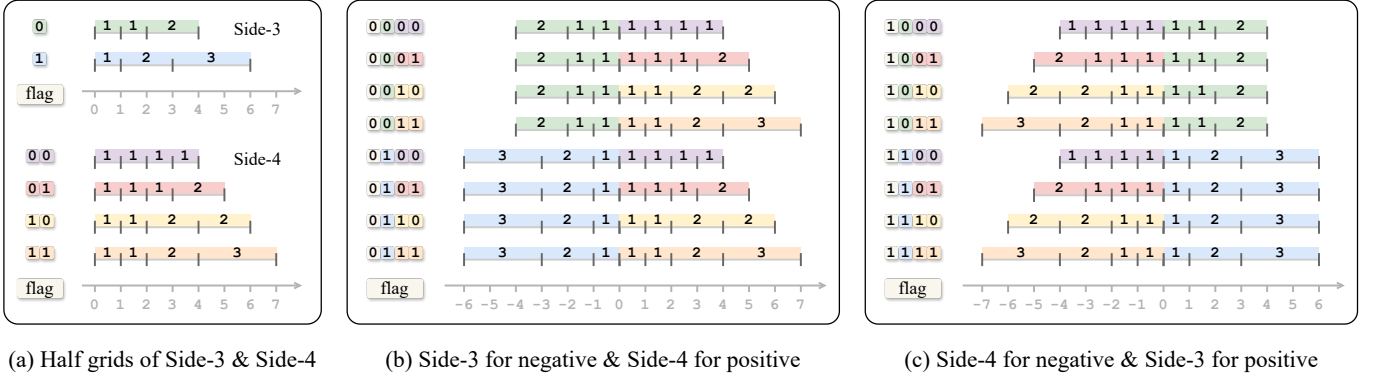


Fig. 4: Overview of the construction of 16 quantization grids (sub numeric types) in SA-ANT-L.

As a result, SA-ANT-L contains 16 different quantization grids, as shown in Fig. 4 (b) and (c). From this step, it becomes evident that quantization grids obtained by combining the half grids constructed from NDSs inherently align with **properties (1) and (2)** of skewed normal distributions.

Similarly, for SA-ANT-P, we first construct half grids of Side-3 ($n_T = 4$) and Side-4 ($n_Q = 8$) from the NDSs listed in Table I, and then combine the half grids as above. Finally, 64 different quantization grids are obtained, which enables lower quantization errors for low-bit group-wise LLM quantization.

It is noteworthy that the NDSs in Table I for defining SA-ANT-L and SA-ANT-P are not unique; they are selected to ensure that the range of decoded values is aligned with a specific integer bit-width, thereby simplifying computation. Other valid NDS choices are also possible and may result in alternative SA-ANT-X numeric types with different trade-offs.

C. Sub Numeric Type Adaptation

To adaptively select the most suitable sub numeric type (i.e., quantization grid), we minimize the norm of the difference between the quantized data and the original data as

$$G^* = \arg \min_{G_i} \|W_f - \Delta \cdot q(W_f, G_i)\|, \quad G_i \in G_{SA}, \quad (4)$$

where G_{SA} denotes the set of all sub numeric types in SA-ANT (see (3)), W_f is the original weight, $q(\cdot)$ is the quantization operation (see (1)), and G^* is the selected sub numeric type. In our quantization framework, the adaptive selection is guided by the ℓ_2 norm criterion.

Notably, weight quantization with adaptive numeric types is commonly performed offline on GPUs. Since there are no data dependencies across groups, the process is highly parallelizable and therefore extremely efficient. For example, on a single NVIDIA 3090 GPU, quantizing all weights of Llama-3.1-8B [4] into SA-ANT-L and SA-ANT-P takes less than one minute and less than three minutes, respectively.

IV. SA-ANT ACCELERATOR

As shown in Fig. 5, a specialized accelerator is designed to support the inference of LLMs whose weights are quantized into 3-bit SA-ANT. The SA-ANT weights are first decoded and then multiplied with INT8 activations by a 128×32 multiplier array. The 32 products in each row are compressed through an adder tree into a single result, which is subsequently

dequantized using the group-wise INT8 scaling factor. Finally, the dequantized result is accumulated into the partial sum stored in the buffer at their corresponding channel locations.

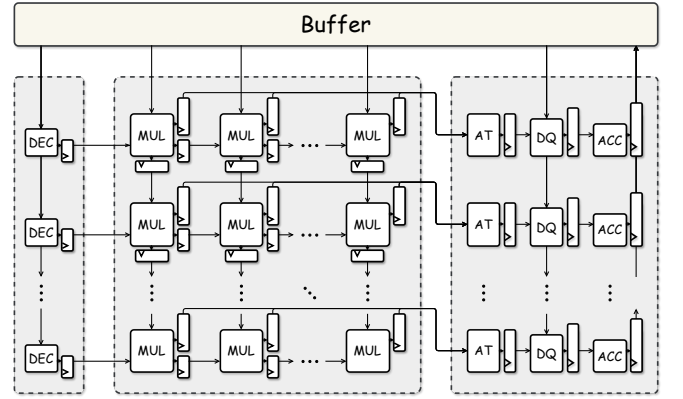


Fig. 5: Microarchitecture and dataflow of the SA-ANT accelerator, consisting of a decoder (DEC) column, a multiplier (MUL) array, an adder tree (AT) column, a dequantizer (DQ) column, and an accumulator (ACC) column.

Decoder Column. Guided by the flag of the group-wise sub numeric type (see Fig.4), the decoder transforms the 3-bit SA-ANT weights into low-bit INT values. Notably, even for SA-ANT-P that contains 64 sub numeric types, a straightforward combinational decoder implemented with an explicit case enumeration results in only limited hardware overhead, so no additional optimizations are performed. As shown in Table I, the largest quantization point in the half grid of SA-ANT-L is 7 (by NDS $\{1, 1, 2, 3\}$), which indicates that data in SA-ANT-L can always be decoded into INT4. Likewise, the maximum possible value for SA-ANT-P is 13 (by NDS $\{1, 4, 4, 4\}$), so it can be decoded into INT5.

Multiplier Array. The multiplier array is organized as a systolic structure. Each multiplier executes $\text{INT}_4 \times \text{INT}_8$ for SA-ANT-L and $\text{INT}_5 \times \text{INT}_8$ for SA-ANT-P. Each row of 32 multipliers collaboratively processes a single weight group. The decoded weights are preloaded into local registers, and the INT8 activations stream vertically through the array, multiplying with weights from different channels in sequence.

TABLE II: Perplexity (**lower is better**) of diverse LLM families for 3-bit weight quantization with a group size of 32, INT8 scaling factors, and INT8 token-wise activations on WikiText-2.

Method	OPT				Qwen-2.5				Falcon-3		Llama-3.1/3.2			Mean
	125M	350M	1.3B	2.7B	0.5B	1.5B	3B	7B	3B	7B	1B	3B	8B	
FP16	27.65	22.00	14.62	12.47	13.07	9.27	8.03	6.85	8.02	6.11	9.75	7.81	6.24	11.68
ANT	38.64	35.42	52.31	143.18	33.80	16.77	427.27	8.94	10.97	7.34	22.55	11.92	9.44	62.97
OliVe	38.46	36.98	28.33	41.73	33.72	15.28	31.87	9.03	11.16	7.34	22.12	11.66	9.11	22.83
M-ANT	38.56	34.92	49.09	139.72	32.07	16.78	349.85	8.99	10.88	7.36	22.77	11.78	9.26	56.31
BitMod	34.29	30.11	18.88	20.95	21.30	12.33	11.53	7.97	9.77	6.85	15.43	9.87	7.80	15.93
SA-ANT-L	32.86	28.20	17.62	17.78	19.15	11.70	10.39	7.70	9.47	6.71	14.40	9.46	7.54	14.84
SA-ANT-P	31.90	27.63	17.28	15.94	17.80	11.44	10.06	7.63	9.32	6.66	13.49	9.15	7.38	14.28

V. EVALUATION

A. Perplexity and Zero-Shot Accuracy

We first compare the quantization performance of SA-ANT with four state-of-the-art adaptive numeric types, ANT [18], OliVe [19], M-ANT [14], and BitMod [15]. ANT and OliVe target tensor-wise or channel-wise quantization in the original designs. M-ANT uses FP16 activations and quantizes weights with a group size of 64, while retaining group-wise scaling factors in FP16. BitMod also employs FP16 activations but quantizes weights with a group size of 128. For fair comparison, we extend ANT and OliVe to group-wise quantization and unify the configurations across all methods. The evaluation covers diverse LLM families, including OPT [2], Qwen-2.5 [3], Falcon-3 [23], and Llama-3.1/3.2 [4], which are quantized with different adaptive numeric types and then assessed in terms of perplexity on WikiText-2 [24]. Note that lower perplexity indicates stronger language modeling ability. The results are summarized in Table II.

It shows that in most models, OliVe achieves lower perplexity than ANT and M-ANT that consist of sub numeric types constrained to strictly symmetric quantization grids. BitMod achieves relatively low perplexity by adaptively inserting a quantization point into the FP format, which increases its flexibility in modeling asymmetric and non-uniform distributions. Both variants of SA-ANT deliver the lowest perplexity, reaching performance closest to the original FP16 model while compressing activations by 50% and weights by about 80%. Specifically, compared to the best existing adaptive numeric type (BitMod), SA-ANT-L and SA-ANT-P reduce the average perplexity by 1.09 and 1.65, respectively, establishing a new state of the art for 3-bit weight quantization.

To evaluate the performance of SA-ANT under different group sizes, we keep all other experimental settings unchanged and increase the group size to 64 and 128. The perplexity results of three LLMs are shown in Table III. As the group size grows, the quantization granularity becomes coarser, leading to degraded language modeling ability of the quantized LLMs, as shown by the higher perplexity. For each group size, both variants of SA-ANT consistently achieve the lowest perplexity, demonstrating the robustness of our method in low-bit group-wise quantization.

SA-ANT can be seamlessly integrated into software-only quantization frameworks by substituting SA-ANT quantizer for the original one. To illustrate the integrability of SA-ANT,

TABLE III: Perplexity (**lower is better**) for 3-bit weight quantization with group sizes of 64 and 128, INT8 scaling factors, and INT8 token-wise activations on WikiText-2.

Method	OPT-125M		Qwen-2.5-0.5B		Llama-3.2-1B	
	g64	g128	g64	g128	g64	g128
ANT	44.30	61.49	51.46	74.26	30.93	52.74
OliVe	40.93	44.60	39.03	52.26	27.96	35.38
M-ANT	44.03	62.50	49.58	81.88	31.63	50.35
BitMod	39.81	49.76	26.13	32.55	18.07	20.31
SA-ANT-L	34.98	38.08	22.96	28.19	16.42	19.47
SA-ANT-P	34.18	35.45	21.27	25.34	15.31	17.50

we combine SA-ANT with AWQ [13]. AWQ is an emerging software-only weight quantization framework that typically employs asymmetric uniform quantization, e.g., INT3-ASYM. Different methods are evaluated on the quantized Llama-3.2-1B by measuring perplexity on C4 [25], PTB [26], and Pile [27]. For fair comparison, we adopt the same quantization settings as AWQ. As shown in Table IV, incorporating SA-ANT consistently reduces perplexity, confirming that SA-ANT is orthogonal to AWQ. Furthermore, both variants of SA-ANT surpass INT3-ASYM in this evaluation.

TABLE IV: Perplexity (**lower is better**) of Llama-3.2-1B for 3-bit weight quantization, evaluated with AWQ and without it (Naive). All experiments are conducted with a group size of 128, FP16 scaling factors, and FP16 activations.

Method	C4		PTB		Pile	
	naive	awq	naive	awq	naive	awq
FP16	14.01	-	17.59	-	8.92	-
INT3-ASYM	56.47	24.44	62.98	30.10	28.90	14.42
SA-ANT-L	28.05	22.15	36.19	27.42	16.05	13.37
SA-ANT-P	25.63	21.78	32.70	26.49	14.94	13.05

To further validate the effectiveness of SA-ANT, Different adaptive numeric types are evaluated on the quantized Qwen-2.5-0.5B (all integrated with AWQ) on five tasks, LAMBADA [28], PIQA [29], ARC-Challenge [30], ARC-Easy [30], and HellaSwag [31]. The zero-shot accuracy results are presented in Table V. Across these downstream tasks, SA-ANT consistently achieves higher average accuracy than existing adaptive numeric types, highlighting its superior adaptability. Specifically, compared to prior methods, SA-ANT-L and SA-ANT-P improve the average accuracy by 1.08%–2.73% and 3.92%–

5.57%, respectively. Compared to the original FP16 model, our quantization framework with SA-ANT-P compresses activations by 50% and weights by about 80%, while incurring only a 2.78% average accuracy drop.

TABLE V: Zero-shot accuracy (% , **higher is better**) of Qwen-2.5-0.5B for 3-bit weight quantization with AWQ. All experiments are conducted with a group size of 32, INT8 scaling factors, and INT8 token-wise activations.

Method	Lambda	Piqa	Arc-e	Arc-c	Hella	Mean
FP16	52.49	70.02	58.67	32.42	52.12	53.14
ANT	40.83	65.23	51.35	28.41	43.31	45.83
OliVe	35.20	65.51	50.04	28.07	45.13	44.79
M-ANT	40.71	65.29	53.54	28.24	43.76	46.31
BitMod	36.39	65.18	54.84	29.10	46.71	46.44
SA-ANT-L	42.60	67.30	50.63	30.12	46.93	47.52
SA-ANT-P	45.24	67.52	59.93	31.23	47.86	50.36

In the unified group-wise quantization configuration used for the experiments in Table V, the adaptive flags of M-ANT and BitMod incur additional memory overheads of 3.8% (4 bits per group) and 1.9% (2 bits per group), respectively. In comparison, SA-ANT-L and SA-ANT-P require 4-bit and 6-bit flags, leading to overheads of 3.8% and 5.8%, respectively. Considering the accuracy improvements, we argue that these modest increases in memory costs are negligible.

B. Area and Power

We assess the hardware efficiency of SA-ANT by comparing the area and power of its compute core with those of other adaptive numeric types. To isolate the effect of numeric types from variations in dataflow and quantization settings, we standardize the accelerator design across all methods. Specifically, we employ a weight-stationary dataflow, a 128×32 multiplier array supporting weight group sizes that are multiples of 32 (e.g., 32, 64, 128) with INT8 activations, and dequantizers that apply INT8 scaling factors to the group-wise results. Each accelerator is carefully optimized to ensure a fair comparison. The compute cores of accelerators based on different adaptive numeric types are modeled in SystemVerilog and synthesized with Synopsys Design Compiler using TSMC 28 nm CMOS technology. The clock frequency, supply voltage, and temperature are set to 500 MHz, 0.9 V, and 25 °C, respectively. The post-synthesis design area is then reported. Subsequently, some tiled matrix multiplications randomly sampled from the inference of quantized Llama-3.2-1B are used as input stimuli for the designs, which are simulated with Synopsys Verilog Compile Simulator to generate waveforms. Finally, time-based power analysis is performed with Synopsys PrimeTime PX to evaluate the power of different designs. The results are summarized in Table VI.

Among the considered designs, M-ANT incurs relatively large area and power because it decomposes individual multiplications into multiple operations, which introduce additional data paths. In contrast, the extra quantization points in BitMod can be accommodated within the same decoding paradigm as FP, resulting in relatively low hardware overhead. SA-ANT-L achieves the lowest area and power by decoding quantized

TABLE VI: Area and power of compute cores with combinational (Comb.) and sequential (Seq.) logic shown separately.

Method	Area (mm ²)			Power (mW)		
	Comb.	Seq.	Total	Comb.	Seq.	Total
ANT	0.463	0.320	0.783	129.1	281.6	410.7
OliVe	0.443	0.353	0.796	159.2	316.7	475.9
M-ANT	0.692	0.420	1.113	259.3	414.8	674.1
BitMod	0.373	0.295	0.668	133.6	276.7	410.3
SA-ANT-L	0.343	0.273	0.616	119.7	258.6	378.3
SA-ANT-P	0.500	0.295	0.795	166.3	279.8	446.1

weights into low-bit INT, which can be efficiently implemented by using IPs in Synopsys DesignWare. Compared to BitMod that attains the highest average zero-shot accuracy among existing adaptive numeric types (see Table V), SA-ANT-L reduces area and power by 7.84% and 7.80%, respectively, while also delivering higher accuracy.

To analyze the compute core overhead of the SA-ANT accelerator, we further report the area breakdown of different components as shown in Fig. 6. The multiplier array dominates the compute core area, while the decoders in SA-ANT-L and SA-ANT-P contribute only a negligible portion, which is the sole overhead introduced to support SA-ANT. With this marginal cost, existing INT architectures can seamlessly support SA-ANT, making it a practical extension for LLM accelerators and enabling additional software-hardware co-optimization opportunities.

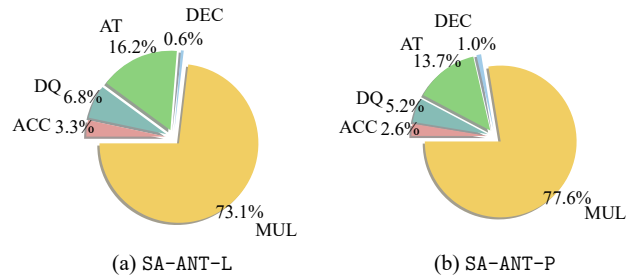


Fig. 6: Area breakdown of different components in the SA-ANT compute cores.

VI. CONCLUSION

To address the skewed normal distributions commonly observed in group-wise quantization, we propose a sign-asymmetric adaptive numeric type (SA-ANT). By asymmetrically partitioning the code-space, constructing half grids based on non-decreasing sequences, and combining them into diverse sub numeric types, SA-ANT adapts naturally to asymmetric and non-uniform distributions. We further design an efficient accelerator tailored to SA-ANT. Experimental results demonstrate that, compared to existing adaptive numeric types, SA-ANT consistently preserves stronger language modeling performance across different group sizes in 3-bit weight quantization, while also remaining compatible with software-only quantization methods. Moreover, the proposed SA-ANT accelerator achieves higher accuracy together with substantial area savings and power reductions.

REFERENCES

- [1] T. Brown, B. Mann, N. Ryder *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [2] S. Zhang, S. Roller, N. Goyal *et al.*, “OPT: Open Pre-trained Transformer Language Models,” *arXiv preprint arXiv:2205.01068*, 2022.
- [3] Q. Team *et al.*, “Qwen2 Technical Report,” *arXiv preprint arXiv:2407.10671*, vol. 2, no. 3, 2024.
- [4] A. Grattafiori, A. Dubey, A. Jauhri *et al.*, “The Llama 3 Herd of Models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [5] J. Kaplan, S. McCandlish, T. Henighan *et al.*, “Scaling Laws for Neural Language Models,” *arXiv preprint arXiv:2001.08361*, 2020.
- [6] A. Gholami, Z. Yao, S. Kim *et al.*, “AI and Memory Wall,” *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.
- [7] J. Achiam, S. Adler, S. Agarwal *et al.*, “GPT-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [8] E. Frantar, S. Ashkboos, T. Hoefler *et al.*, “GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers,” *arXiv preprint arXiv:2210.17323*, 2022.
- [9] G. Xiao, J. Lin, M. Seznec *et al.*, “SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models,” in *International conference on machine learning*. PMLR, 2023, pp. 38 087–38 099.
- [10] W. Shao, M. Chen, Z. Zhang *et al.*, “OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models,” in *The Twelfth International Conference on Learning Representations*.
- [11] S. Dai, R. Venkatesan, M. Ren *et al.*, “VS-Quant: Per-vector Scaled Quantization for Accurate Low-Precision Neural Network Inference,” *Proceedings of Machine Learning and Systems*, vol. 3, pp. 873–884, 2021.
- [12] B. Darvish Rouhani, R. Zhao, V. Elango *et al.*, “With Shared Microexponents, A Little Shifting Goes a Long Way,” in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.
- [13] J. Lin, J. Tang, H. Tang *et al.*, “AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration,” *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024.
- [14] W. Hu, H. Zhang, C. Guo *et al.*, “M-ANT: Efficient Low-bit Group Quantization for LLMs via Mathematically Adaptive Numerical Type,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1112–1126.
- [15] Y. Chen, A. F. AbouElhamayed, X. Dai *et al.*, “BitMoD: Bit-serial Mixture-of-Datatype LLM Acceleration,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1082–1097.
- [16] S.-y. Liu, Z. Liu, X. Huang *et al.*, “LLM-FP4: 4-Bit Floating-Point Quantization for LLMs,” *arXiv preprint arXiv:2310.16836*, 2023.
- [17] S. Agarwal, L. Ahmad, J. Ai *et al.*, “Gpt-oss-120b & gpt-oss-20b Model Card,” *arXiv preprint arXiv:2508.10925*, 2025.
- [18] C. Guo, C. Zhang, J. Leng *et al.*, “ANT: Exploiting Adaptive Numerical Data Type for Low-bit Deep Neural Network Quantization,” in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 1414–1433.
- [19] C. Guo, J. Tang, W. Hu *et al.*, “OliVe: Accelerating Large Language Models via Hardware-friendly Outlier-Victim Pair Quantization,” in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–15.
- [20] X. Geng, S. Liu, J. Jiang *et al.*, “Compact Powers-of-Two: An Efficient Non-Uniform Quantization for Deep Neural Networks,” in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.
- [21] X. Geng, S. Liu, L. Liu *et al.*, “QUQ: Quadruplet Uniform Quantization for Efficient Vision Transformer Inference,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [22] M. Nagel, M. Fournarakis, R. A. Amjad *et al.*, “A White Paper on Neural Network Quantization,” *arXiv preprint arXiv:2106.08295*, 2021.
- [23] E. Almazrouei, H. Alobeidli, A. Alshamsi *et al.*, “The Falcon Series of Open Language Models,” *arXiv preprint arXiv:2311.16867*, 2023.
- [24] S. Merity, C. Xiong, J. Bradbury *et al.*, “Pointer Sentinel Mixture Models,” *arXiv preprint arXiv:1609.07843*, 2016.
- [25] J. Dodge, M. Sap, A. Marasović *et al.*, “Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus,” *arXiv preprint arXiv:2104.08758*, 2021.
- [26] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, “Building a large annotated corpus of English: The Penn Treebank,” *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
- [27] L. Gao, S. Biderman, S. Black *et al.*, “The Pile: An 800GB Dataset of Diverse Text for Language Modeling,” *arXiv preprint arXiv:2101.00027*, 2020.
- [28] D. Paperno, G. Kruszewski, A. Lazaridou *et al.*, “The LAMBADA dataset: Word prediction requiring a broad discourse context,” in *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 2016, pp. 1525–1534.
- [29] Y. Bisk, R. Zellers, J. Gao *et al.*, “PIQA: Reasoning about Physical Commonsense in Natural Language,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.
- [30] P. Clark, I. Cowhey, O. Etzioni *et al.*, “Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge,” *arXiv preprint arXiv:1803.05457*, 2018.
- [31] R. Zellers, A. Holtzman, Y. Bisk *et al.*, “HellaSwag: Can a Machine Really Finish Your Sentence?” *arXiv preprint arXiv:1905.07830*, 2019.