♨PoT-PTQ:
Two-step Power-of-Two Post-training for LLMs

Xinyu Wang[a,*], **Vahid Partovi Nia**[b], **Peng Lu**[c], **Jerry Huang**[c,d], **Xiao-Wen Chang**[a], **Boxing Chen**[b] and **Yufei Cui**[b,**]

[a]McGill University, Canada
[b]Noah's Ark Lab, Canada
[c]Université de Montréal, Canada
[d]Mila – Quebec AI Institute, Canada

Abstract. Large Language Models (LLMs) have demonstrated remarkable performance across various natural language processing (NLP) tasks. However, their deployment is challenging due to the substantial computational resources required. *Power-of-two* (PoT) quantization is a general tool to counteract this difficulty. Albeit previous works on PoT quantization can be efficiently dequantized on CPUs using fixed-point addition, it showed less effectiveness on GPUs. The reason is entanglement of the sign bit and sequential bit manipulations needed for dequantization. We propose a novel POT quantization framework for LLM weights that (i) outperforms state-of-the-art accuracy in extremely low-precision number formats, and (ii) enables faster inference through more efficient dequantization. To maintain the accuracy of the quantized model, we introduce a two-step post-training algorithm: (i) initialize the quantization scales with a robust starting point, and (ii) refine these scales using a minimal calibration set. The performance of our PoT post-training algorithm surpasses the current state-of-the-art in integer quantization, particularly at low precisions such as 2- and 3-bit formats. Our PoT quantization accelerates the dequantization step required for the floating point inference and leads to $3.67\times$ speed up on a NVIDIA V100, and $1.63\times$ on a NVIDIA RTX 4090, compared to uniform integer dequantization.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing (NLP) tasks, including text generation, summarization, and question answering [3, 29, 21, 26]. However, deploying them remains challenging due to their memory and computation requirements, posing obstacles when using them in the real world. Quantization has emerged as an effective strategy to reduce these costs by converting full-precision weights into lower-bit representations, e.g. INT8 and INT4, reducing memory usage and accelerating computation. However, aggressive quantization can lead to accuracy degradation, especially in generation tasks, and despite weights being stored in low-bit formats, inference typically involves dequantizing them back to FP16 for general matrix multiplication (GEMM), introducing latency and limiting speed-ups.

Post-training quantization (PTQ)[15, 17, 9, 16, 20, 6], methods offer a practical trade-off: they compress pretrained models without requiring full retraining and are compatible with small calibration sets. While existing PTQ approaches have achieved strong performance in the 3–4 bit regime, maintaining accuracy at greater quantization levels (e.g., 2-bit) remains difficult. Additionally, standard dequantization pipelines often rely on costly mixed-precision operations, further slowing inference. As LLMs scale further, extreme quantization — such as 2-bit or even binary weights — becomes increasingly desirable for edge device deployment and low-latency applications. Achieving this level of compression without compromising model quality or inference speed requires new approaches to quantization and hardware efficiency.

Power-of-Two (PoT) quantization offers a promising direction. By constraining weights to signed powers of two, it enables multiplications to be replaced with simple shift-and-add operations, leading to substantial speed gains. Originally explored in computer vision [27, 19, 13], PoT quantization aligns well with the bell-shaped distributions and exponential tails typically found in trained weight matrices. This structure makes PoT not only hardware-efficient but also statistically well-suited for deep models. However, when applied to LLMs, existing methods fail to retain accuracy due to coarse rounding and lack of effective post-training calibration. Moreover, naive dequantization can be inefficient on modern GPUs due to bit-level dependencies and sign-bit entanglement. Existing attempts to apply power-of-two quantization to LLMs have resulted in significant accuracy degradation, especially at extreme bit-widths (Table 2).

In this work, we propose a novel post-training quantization framework for LLMs using power-of-two values. Our method achieves both high accuracy in the low-bit regime and fast, hardware-friendly inference. We highlight our main contributions as follows:

- We introduce a two-stage post-training algorithm that combines robust scale initialization with lightweight calibration tailored to the PoT structure.
- We demonstrate our method consistently outperforms strong PTQ baselines at 2 and 3-bit precision across standard benchmarks.
- We develop a GPU-optimized dequantization kernel that leverages bitwise parallelism, resulting in up to $3.67\times$ speed-up on an NVIDIA V100 and $1.63\times$ on a RTX 4090 compared to standard integer dequantization.

Our results highlight power-of-two quantization as a scalable and efficient solution for high-performance LLM deployment under strin-
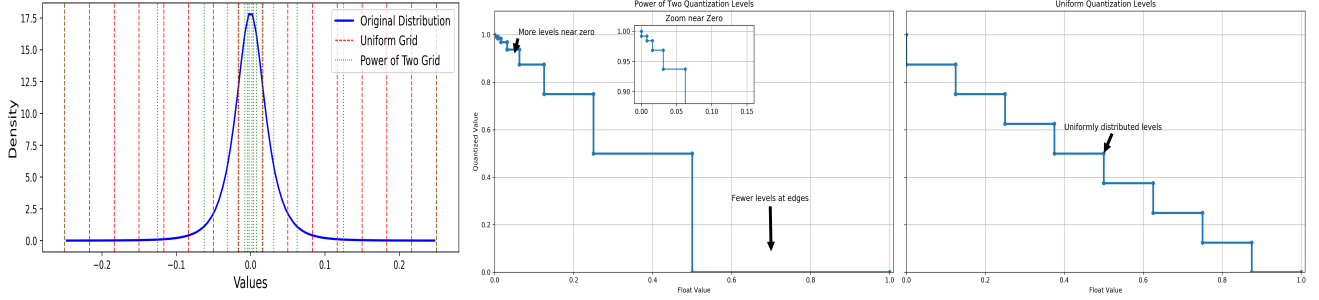
Figure 1: Left: Density distribution of the first weight matrix in `LLaMA-7B`, exemplifying the bell-shaped or exponential decay commonly observed in LLM weight distributions. Middle: Quantization levels for power-of-two (PoT) quantization, showing finer resolution near zero, aligning well with the distribution's high-density region. Right: Quantization levels for uniform quantization, which allocate levels evenly and poorly capture the dense region near zero.

gent hardware constraints.

2 Background

2.1 Quantization Methods for LLMs

Recent advances in LLM quantization primarily fall into two categories: *weight-only*, which reduces the bit-width of weights while keeping activations in higher precision, and *weight-activation*, where both components are compressed. Weight-only quantization is particularly attractive for auto-regressive generation as it directly reduces the memory/bandwidth cost during inference while keeping the model forward computation simple and stable.

Several post-training quantization (PTQ) methods have been proposed to enable low-bit inference without retraining. `LLM.int8` [5] performs 8-bit quantization with a dedicated outlier channel retained in higher precision. SmoothQuant [25] applies a per-channel scaling between weights and activations to mitigate outlier effects, and Outlier Suppression [24] takes a similar approach by explicitly isolating and handling activation outliers.

Focusing on weight-only PTQ, GPTQ [9] proposes a one-shot calibration method based on second-order optimization, achieving strong performance down to 3-bit quantization. AWQ [16] improves robustness via mixed-precision quantization that selectively preserves salient weights in higher precision. OmniQuant [20] introduces block-wise learnable clipping bounds to reduce quantization error, while SqueezeLLM [10] explores non-uniform quantization using k-means clustering. QuIP [4] refines GPTQ's optimization and extends support to 2-bit quantization with improved calibration. SpQR [7] and OWQ [11] further push compression limits by mixing high and low-bit representations based on importance scores.

While these methods enable impressive accuracy at 3-bit and 4-bit regimes, they face challenges at extreme low bit-widths (e.g., 2-bit), where discretization artifacts become more prominent. Moreover, many mixed-precision designs introduce dequantization inefficiencies, as multipliers still need to convert integer values to FP16 for matrix multiplication kernels.

In what follows, we turn to power-of-two quantization, which offers a promising direction for reducing both quantization error and dequantization overhead.

2.2 PoT Quantization

PoT quantization restricts each weight value to a signed power of two, such as $\pm 2^{-2}, \pm 2^{-1}, \pm 2^{0}$, forming a logarithmically spaced set of quantization levels. Unlike uniform quantization, which allocates evenly spaced intervals, PoT uses a geometric progression that offers both statistical and computational advantages.

(1) Statistical alignment with weight distributions. As shown in Figure 1 (left), weights in LLMs typically follow bell-shaped or exponentially decaying distributions. The geometric structure of PoT levels aligns more naturally with these distributions—especially in low-magnitude regions—resulting in better approximation under limited bit-widths. In contrast, uniform quantization over-allocates levels to large values and poorly captures the dense region near zero.

(2) Hardware-efficient dequantization. PoT quantization enables fast dequantization through shift and add operations. Each quantized weight is reconstructed as $\pm 2^{\mathsf{E}} \cdot S$, where S is a per-group scaling factor. Since 2^{E} can be implemented using bit shifts, PoT dequantization is particularly efficient for matrix multiplication on modern GPUs.

Despite these attractive properties, directly applying existing post-training quantization (PTQ) methods [28, 14, 8, 12, 13] to PoT often results in severe accuracy degradation. Most prior techniques are designed for uniform quantization and do not adapt well to the coarse, non-linear nature of PoT levels. The primary challenge lies in properly determining scaling factors that map weights into the limited exponent range, especially under aggressive bit constraints like 2-bit or 3-bit. Improper scaling leads to large quantization errors that existing activation-based calibration cannot resolve.

These limitations highlight the need for a dedicated PoT quantization framework that explicitly accounts for the structural properties of LLMs and the discrete nature of power-of-two representations. In the following sections, we present such a framework based on principled scale initialization and lightweight calibration.

## 3 Two-Step Power-of-Two Post-Training Quantization

3.1 Preliminaries

We focus on weight-only quantization for transformer-based LLMs using Power-of-Two (PoT) representations. Let $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d}$ denote the weight matrix at layer l, where each element $\mathbf{W}_{ij}^{(l)}$ is quantized independently. PoT quantization represents each weight as a signed power-of-two value scaled by a learnable factor:

$$\widetilde{\mathbf{W}}_{ij}^{(l)} = \mathbf{S}_{ij}^{(l)} \cdot \mathbf{P}_{ij}^{(l)} \cdot 2^{\mathsf{E}_{ij}^{(l)}}, \tag{1}$$

where

- $\mathbf{S}_{ij}^{(l)} \in \mathbb{R}_{+}$ is the scale shared within a quantization group;
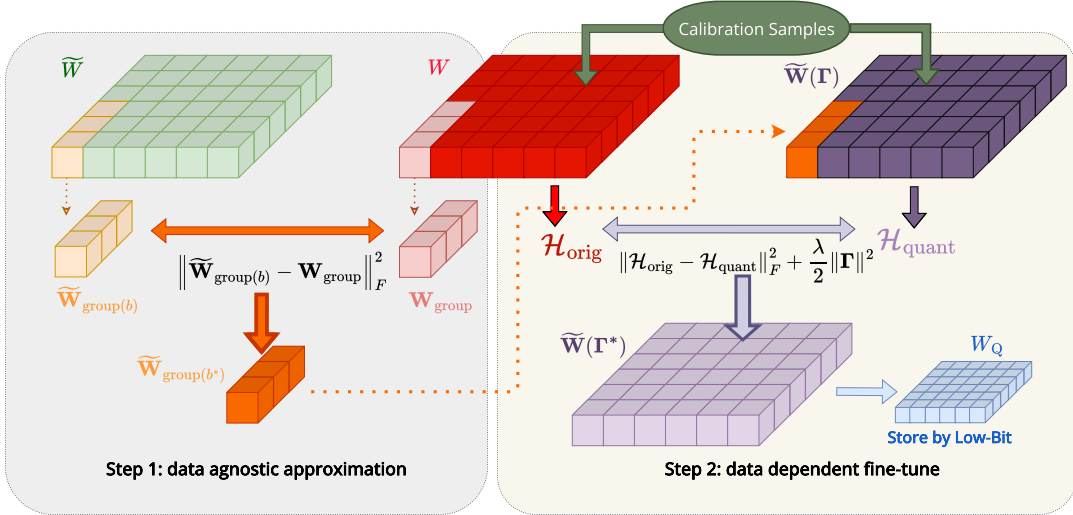
Figure 2: The two step algorithm. **DASI (Section 3.2)**: The scale is adjusted by aligning the weight matrices (left). **DDFT (Section 3.3)**: Activations are aligned using minimal calibration data (right).

- $\mathbf{P}_{ij}^{(l)} \in \{-1, +1\}$ is the sign bit, defined as $\mathbf{P}_{ij}^{(l)} = \text{sign}(\mathbf{W}_{ij}^{(l)})$;
- $\mathsf{E}_{ij}^{(l)} \in \mathbb{N}_0$ is the quantized exponent, computed as:

$$\mathsf{E}_{ij}^{(l)} = \text{clamp}\left(\text{round}\left(\log_2\left(\left|\mathbf{W}_{ij}^{(l)}\right|/\mathbf{S}_{ij}^{(l)}\right)\right), 0, q_{\max}\right), \quad (2)$$

where $q_{\max} = 2^{n-1} - 1$ for n-bit quantization, and the clamp operation limits a value to a specified interval; specifically for real values x, a, and b

$$\text{clamp}(x, a, b) = \min(\max(x, a), b). \quad (3)$$

We can write (1) in the matrix form:

$$\widetilde{\mathbf{W}}^{(l)} = \mathbf{S}^{(l)} \circ \mathbf{P}^{(l)} \circ 2^{\mathsf{E}^{(l)}}, \quad (4)$$

where $\circ$ denotes Hadamard (element-wise) multiplication, and all variables retain their element-wise definitions over the matrix.

To reduce memory and computational costs, we apply **group-wise quantization**: each column of $\mathbf{W}^{(l)}$ is divided into fixed-size groups (e.g., 64 or 128 rows), and all weights within a group share the same scale $\mathbf{S}_{ij}^{(l)}$. This is enforced by:

$$\mathbf{S}_{ij}^{(l)} = \mathbf{S}_{kj}^{(l)}, \quad \text{if } \lfloor i/G \rfloor = \lfloor k/G \rfloor, \quad (5)$$

where G is the group size.

This representation forms the foundation of our quantization framework. In the following sections, we describe how to initialize and calibrate the scale factors $\mathbf{S}^{(l)}$, enabling accurate and efficient PoT quantization for large language models.

3.2 Step 1: Data-Agnostic Scale Initialization (DASI)

POT quantization introduces distinct challenges due to its logarithmic and discrete structure. In this stage, we initialize group-wise scaling factors purely from weights, without relying on activation data. This scale initialization is crucial for ensuring quantization robustness under low-bit settings.

Motivation: Sensitivity and Non-Smooth Error Surface. PoT quantization results in a highly non-smooth loss landscape, owing to discrete exponent rounding. As illustrated in Figure 3 (middle), small variations in scale lead to abrupt shifts in exponent values, producing

---

Algorithm 1 Parallel Data-Agnostic Scale Initialization (Step 1)

1: **for** each weight group $\mathbf{W}_{\text{group}}$ **in parallel do**
2:     $s_0 \leftarrow \dfrac{\max |\mathbf{W}_{\text{group}}|}{2^{q_{\max}} - 1}$ ▷ Initialize base scale
3:     $\mathcal{B} \leftarrow \{0.01 \cdot i \mid i = 1, \ldots, 200\}$ ▷ Set candidate multipliers
4:     $Q_1^{\min} \leftarrow \infty$ ▷ Initialize minimum error
5:     **for** each $b \in \mathcal{B}$ **do**
6:         $s_b \leftarrow s_0 \cdot b$ ▷ Compute candidate scale
7:         $\mathsf{E}(b) \leftarrow \text{clamp}\left(\text{round}\left(\log_2\left(|\mathbf{W}_{\text{group}}|/s_b\right)\right), 0, q_{\max}\right)$
8:         $\widetilde{\mathbf{W}}_{\text{group}}(b) \leftarrow s_b \cdot \text{sign}(\mathbf{W}_{\text{group}}) \circ 2^{\mathsf{E}(b)}$
9:         $Q_1(b) \leftarrow \left\|\mathbf{W}_{\text{group}} - \widetilde{\mathbf{W}}_{\text{group}}(b)\right\|_2^2$
10:        **if** $Q_1(b) < Q_1^{\min}$ **then**
11:            $Q_1^{\min} \leftarrow Q_1(b)$
12:            $b^* \leftarrow b$
13:        **end if**
14:     **end for**
15:     $s^* \leftarrow s_0 \cdot b^*$ ▷ Optimal scale for this group
16:     Store s^* into the corresponding position in $\mathbf{S}^{(l)}$
17: **end for**
18: **return** $\mathbf{S}^{(l)}$

---

sharp error transitions—unlike uniform quantization, which exhibits a smoother loss surface. Furthermore, Figure 3 (right) shows that optimal scales often diverge from the naive initialization ($b = 1$). These properties make gradient-based optimization unreliable and motivate a grid search strategy.

Group Definition. We adopt group-wise quantization, where each column of $\mathbf{W}^{(l)} \in \mathbb{R}^{\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}}$ is partitioned into disjoint groups of size G. Each group contains a contiguous subvector of weights from the same column. Denote such a group as $\mathbf{W}_{\text{group}} \in \mathbb{R}^G$. All elements in a group share the same scaling factor.

Quantization Objective. Given a candidate scale $s = s_0 \cdot b$, we compute the quantized exponent for each element in the group:

$$\mathsf{E}_{\text{group}}(b) = \text{clamp}\left(\text{round}\left(\log_2\left(\frac{|\mathbf{W}_{\text{group}}|}{s_0 \cdot b}\right)\right), 0, q_{\max}\right), \quad (6)$$

where $q_{\max} = 2^{n-1} - 1$ is the maximum exponent level for n-bit quantization, and scalars s_0 and b are defined later. We then define
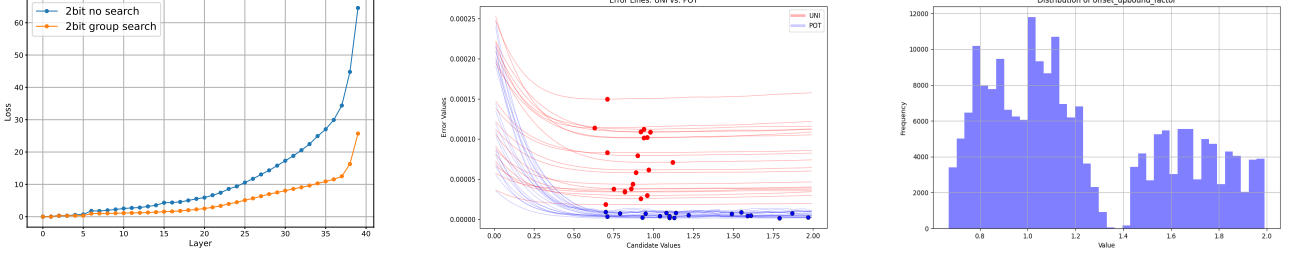
Figure 3: **(Left)** Loss in O/W Step 1. **(Middle)** Loss curve of 3-bit PoT quantization shows non-smooth transitions. **(Right)** Histogram of selected scaling multipliers b^* — many deviate from the naive choice $b = 1$.

the dequantized group reconstruction:

$$\widetilde{\mathbf{W}}_{\text{group}}(b) = s_0 \cdot b \cdot \mathbf{P}_{\text{group}} \circ 2^{\mathsf{E}_{\text{group}}(b)}, \tag{7}$$

where $\mathbf{P}_{\text{group}} = \text{sign}(\mathbf{W}_{\text{group}})$.

The goal is to minimize the mean squared error (MSE) between the original and reconstructed weights:

$$Q_1(b) = \left\| \mathbf{W}_{\text{group}} - \widetilde{\mathbf{W}}_{\text{group}}(b) \right\|_2^2, \tag{8}$$

and select the optimal scale multiplier:

$$b^* = \arg\min_{b \in \mathcal{B}} Q_1(b), \quad s^* = s_0 \cdot b^*, \tag{9}$$

where $\mathcal{B}$ is a set of scaling multipliers defined later.

Initial Scale Estimation. We initialize a base scale s_0 to align the largest magnitude in the group with the top PoT level:

$$s_0 = \frac{\max |\mathbf{W}_{\text{group}}|}{2^{q_{\max}} - 1}. \tag{10}$$

Grid Search Procedure. We evaluate a discrete set of scaling multipliers $\mathcal{B} = \{0.01 \cdot i \mid i = 1, \ldots, 200\}$, and compute $Q_1(b)$ for each. The best multiplier b^* and corresponding scale s^* are retained. This procedure is repeated for every group independently, and the resulting group-wise scales are aggregated into the full matrix $\mathbf{S}^{(l)}$.

Parallelization. Each weight group is quantized independently, allowing Step 1 to be fully parallelized across GPU threads. This parallelism enables efficient scale initialization even for large models. The procedure is summarized in Algorithm 1, which performs a grid search to minimize reconstruction error per group. A visual overview of Step 1 is shown in **the left panel of Figure 2**, where weight matrices are aligned through scale adjustment without any data dependency.

3.3 Step 2: Data-Dependent Fine-Tuning (DDFT) for Layer-Wise Quantization Reconstruction

Following the data-agnostic scale initialization in Step 1 (Section 3.2), we introduce a lightweight fine-tuning stage to further refine the group-wise scaling factors $\mathbf{S}^{(l)}$. This step uses a small calibration set to improve the alignment between the outputs of the quantized and original models. A visual overview of this procedure is shown in the right panel of Figure 2, where activations are aligned using data-dependent scale adjustments. Importantly, this is achieved by learning a low-dimensional residual parameter $\mathbf{\Gamma}$, thereby maintaining low training cost and avoiding full model retraining.

Motivation. Despite accurate weight reconstruction, the initial scales derived in Step 1 do not directly guarantee output-level consistency due to nonlinear interactions between quantized weights and activation patterns. Fine-tuning with calibration data can mitigate this mismatch. Our method adjusts only the scaling factors via a learnable residual, offering a highly efficient yet effective quantization-aware calibration.

Optimization Objective. Let $\mathbf{X} \in \mathbb{R}^{B \times T \times d}$ be the input hidden states, where B is the batch size, T the sequence length, and d the hidden dimension. Define the output of the l-th transformer block as:

$$\mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X}) = \text{MLP}^{(l)}\left(\mathbf{X} + \text{Attn}^{(l)}(\mathbf{X})\right), \tag{11}$$

$$\text{Attn}^{(l)}(\mathbf{X}) = \text{softmax}\left(\mathbf{Q}^{(l)}\mathbf{K}^{(l)\top}/\sqrt{d}\right)\mathbf{V}^{(l)}, \tag{12}$$

$$\text{MLP}^{(l)}(\mathbf{X}) = \sigma\left(\mathbf{X}\mathbf{W}_1^{(l)}\right)\mathbf{W}_2^{(l)}, \tag{13}$$

where $\mathbf{Q}^{(l)} = \mathbf{X}\mathbf{W}_Q^{(l)}$, $\mathbf{K}^{(l)} = \mathbf{X}\mathbf{W}_K^{(l)}$, and $\mathbf{V}^{(l)} = \mathbf{X}\mathbf{W}_V^{(l)}$ are the projected queries, keys, and values. $\sigma(\cdot)$ denotes a nonlinear activation such as GELU.

This formulation also supports grouped-query attention (GQA) [1], where queries are projected in more heads than keys and values; our quantization pipeline remains compatible since scale adjustment is performed independently per group.

The optimization loss is formulated as:

$$\min_{\mathbf{\Gamma}} Q_2(\mathbf{\Gamma}) = \left\| \mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X}) - \mathcal{F}^{(l)}(\widetilde{\mathbf{W}}^{(l)}(\mathbf{\Gamma}), \mathbf{X}) \right\|_F^2 + \frac{\lambda}{2}\|\mathbf{\Gamma}\|_F^2, \tag{14}$$

where $\widetilde{\mathbf{W}}^{(l)}(\mathbf{\Gamma})$ is the reconstructed quantized weight matrix parameterized by the learnable residual $\mathbf{\Gamma}$, and λ controls the strength of the regularization term. The shared scale structure within each quantization group follows Equation (5).

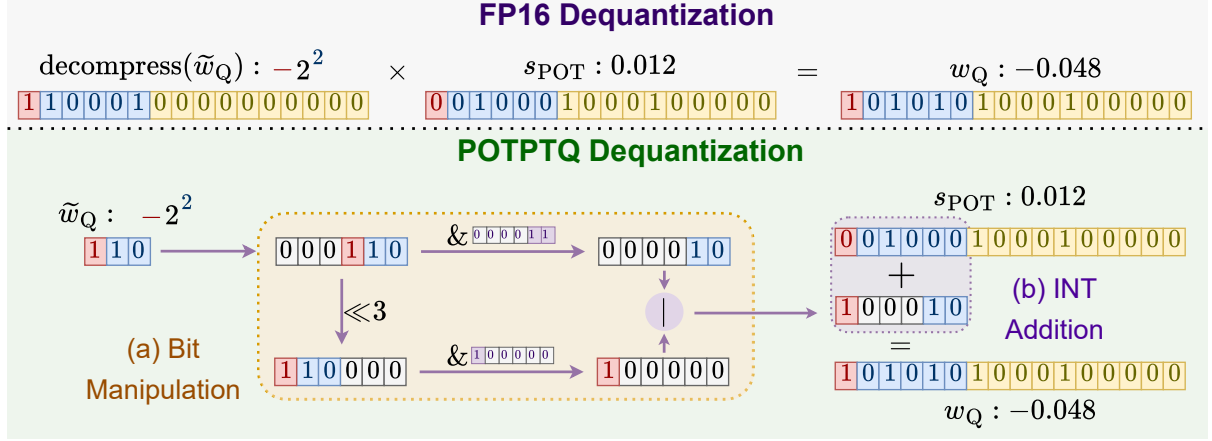**Learnable Scale Adjustment.** We define a refined group-wise scale

$$\hat{\mathbf{S}}_{ij}^{(l)}(\mathbf{\Gamma}) = \mathbf{S}_{ij}^{(l)} \cdot (1 + \mathbf{\Gamma}_{ij}), \tag{15}$$

where $\mathbf{\Gamma}_{ij}$ is initialized to 0 and optimized with gradient-based methods. The dequantization follows:

$$\mathsf{E}_{ij}^{(l)}(\mathbf{\Gamma}) = \text{clamp}\left(\text{round}\left(\log_2\left(\frac{|\mathbf{W}_{ij}^{(l)}|}{\hat{\mathbf{S}}_{ij}^{(l)}(\mathbf{\Gamma})}\right)\right), 0, q_{\max}\right), \tag{16}$$

$$\widetilde{\mathbf{W}}_{ij}^{(l)}(\mathbf{\Gamma}) = \hat{\mathbf{S}}_{ij}^{(l)}(\mathbf{\Gamma}) \cdot \mathbf{P}_{ij} \cdot 2^{\mathsf{E}_{ij}^{(l)}(\mathbf{\Gamma})}. \tag{17}$$

Gradient Approximation via STE. The rounding operation in Equation (16) is non-differentiable. We apply the **Straight-Through**

Figure 4: Multiplication between FP16 scale and power-of-two weights for a 3-bit example implemented through efficient bit manipulation and fixed-point integer addition to speed up power of two dequantization.

Estimator (STE) [2]:

$$\frac{\partial \mathsf{E}_{ij}^{(l)}}{\partial \hat{\mathbf{S}}_{ij}^{(l)}} \approx \frac{\partial}{\partial \hat{\mathbf{S}}_{ij}^{(l)}} \log_2 \left(\frac{|\mathbf{W}_{ij}^{(l)}|}{\hat{\mathbf{S}}_{ij}^{(l)}} \right). \tag{18}$$

This allows us to back-propagate through the quantization process using the smooth surrogate.

Efficiency. This optimization introduces only one learnable scalar per weight group, and fine-tuning requires only a few epochs on a small calibration set. It is compatible with both SGD and Adam optimizers and scales well to large models without modifying the original model architecture or performing any full-rank retraining.

Procedure. The training procedure is summarized in Algorithm 2.

Algorithm 2 Fine-Tuning the Learnable Parameter $\boldsymbol{\Gamma}$ for Scaling Factors

---

1: **Initialize:** $\boldsymbol{\Gamma} \leftarrow \mathbf{0}$
2: **Set Parameters:** Learning rate η, weight decay λ, epochs N
3: **for** epoch $= 1$ to N **do**
4:     **for** each calibration batch $\mathbf{X}$ **do**
5:         Compute original output: $\mathcal{H}_{\text{orig}} \leftarrow \mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X})$
6:         Update scales: $\hat{\mathbf{S}}^{(l)} \leftarrow \mathbf{S}^{(l)} \circ (\mathbf{1}\mathbf{1}^{\top} + \boldsymbol{\Gamma})$
7:         Quantize exponent:
8:           $\mathsf{E}^{(l)} \leftarrow \text{clamp}(\text{round}(\log_2(|\mathbf{W}^{(l)}|/\hat{\mathbf{S}}^{(l)})), 0, q_{\max})$
9:         Dequantize: $\widetilde{\mathbf{W}}^{(l)} \leftarrow \hat{\mathbf{S}}^{(l)} \circ \mathbf{P} \circ 2^{\mathsf{E}^{(l)}}$
10:        Compute quantized output: $\mathcal{H}_{\text{quant}} \leftarrow \mathcal{F}^{(l)}(\widetilde{\mathbf{W}}^{(l)}, \mathbf{X})$
11:        Compute loss: $Q_2 \leftarrow \|\mathcal{H}_{\text{orig}} - \mathcal{H}_{\text{quant}}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{\Gamma}\|_F^2$
12:        Update $\boldsymbol{\Gamma}$ via gradient descent
13:     **end for**
14: **end for**
15: **Return:** Refined scale matrix $\hat{\mathbf{S}}^{(l)} = \mathbf{S}^{(l)} \circ (\mathbf{1}\mathbf{1}^{\top} + \boldsymbol{\Gamma})$

In Line 8, all operations on matrices are carried out elementwise.

4 Efficient Dequantization for PoT Quantized Weights

In our proposed quantization framework, we adopt a PoT dequantization scheme to enable fast and hardware-friendly inference. Unlike traditional uniform quantization that requires floating-point operations,

PoT quantization reconstructs weights using bit-shift and integer arithmetic, which are significantly more efficient on modern hardware. This section presents both a conceptual comparison and the detailed implementation of our optimized PoT dequantization method.

4.1 Power-of-Two vs. Uniform Quantization

Conventional uniform quantization reconstructs the original weight tensor $\widetilde{\mathbf{W}}^{(l)}$ as follows:

$$\widetilde{\mathbf{W}}^{(l)} = (\widetilde{\mathbf{W}}_{\mathrm{Q}}^{(l)} + \mathbf{Z}) \circ \mathbf{S}, \tag{19}$$

where $\widetilde{\mathbf{W}}_{\mathrm{Q}}^{(l)}$ are the quantized values, $\mathbf{Z}$ is the zero-point offset, and S is the per-group scaling factor. This approach requires floating-point multiplication and addition during inference, which may incur significant computational latency.

In contrast, PoT quantization reconstructs weights using:

$$\widetilde{\mathbf{W}}^{(l)} = \mathbf{S}^{(l)} \circ \widetilde{\mathbf{W}}_{\mathrm{Q}}^{(l)}, \tag{20}$$

$$\widetilde{\mathbf{W}}_{\mathrm{Q}}^{(l)} = (-1)^{\mathsf{S}} \circ 2^{\mathsf{E}}, \tag{21}$$

where $\mathbf{S}^{(l)}$ is a learnable scaling factor and $\widetilde{\mathbf{W}}_{\mathrm{Q}}^{(l)}$ encodes each quantized weight using a sign bit (S) and an exponent value (E). This encoding allows us to efficiently multiply quantized weights with activations by simply adjusting exponent values, without requiring explicit multiplication.

Each FP16 value x consists of three parts: the sign bit S, the exponent bits E, and the mantissa bits M:

$$x = \overbrace{\mathsf{S}}^{\text{Sign}} \overbrace{\mathsf{E}_1, \ldots, \mathsf{E}_5}^{\text{Exponent}} \overbrace{\mathsf{M}_1, \ldots, \mathsf{M}_{10}}^{\text{Mantissa}}.$$

The full FP16 value is computed as:

$$x = (-1)^{\mathsf{S}} \times 2^{\mathsf{E}-15} \times 1.\mathsf{M}. \tag{22}$$

Multiplying an FP16 activation x with a PoT quantized weight w, represented as $(-1)^{\mathsf{S}'+\mathsf{S}} \times 2^{\mathsf{E}+\mathsf{E}'-15} \times 1.\mathsf{M}$, is computationally efficient due to the exponent additivity property.

4.2 PoT Dequantization: Bit Manipulation and Integer Addition

To reconstruct dequantized weights in FP16 format, we design a two-stage dequantization pipeline leveraging:

1. **Bit Manipulation:** Efficiently assembling the signed exponent value from the PoT quantized format.

   (a) Extract the Exponent Bits: Given a 3-bit quantized weight $w_Q = S_0E_1E_2$, we isolate the exponent portion using an AND mask `000011`, yielding $0000E_1E_2$.

   (b) Combine Sign and Exponent: We right-shift the quantized bits by three positions and use an AND operation with `100000` to extract the sign bit. We then use a bitwise OR to combine the sign and exponent into the signed representation $S_00000E_1E_2$.

2. **Fixed-Point Integer Addition:** Efficiently assembling the signed exponent value from the PoT quantized format.
   The result is right-shifted by 10 bits to match FP16 exponent alignment and then added to the precomputed FP16 scale s, yielding the final dequantized FP16 weight. This step is carried out in parallel across all quantized values and avoids floating-point operations entirely.

These steps constitute a lightweight, parallelizable pipeline that enables high-throughput inference in GEMM-heavy LLM workloads.

Table 1: WikiText-2 perplexity of quantized LLaMA models. Baselines use uniform quantization; PoT uses power-of-two.

Bits	Model	Size	RTN	GPTQ	AWQ	OMNI	PoTPTQ
3.25	LLaMA	7B	7.01	6.55	6.46	6.16	**6.12**
		13B	5.88	5.62	5.51	5.46	**5.42**
		30B	4.87	4.80	4.63	4.58	**4.50**
	Llama2	7B	6.66	6.29	6.24	6.21	**6.03**
		13B	5.51	5.42	5.32	5.28	**5.24**
2.25	LLaMA	7B	1.90×10^3	44.01	2.6×10^5	**9.77**	9.79
		13B	7.81×10^2	15.6	2.8×10^5	**7.93**	7.96
		30B	68.04	10.92	2.4×10^5	7.13	**7.01**
	Llama2	7B	4.20×10^3	36.77	2.2×10^5	11.23	**11.03**
		13B	122.08	28.14	1.2×10^5	8.33	**8.29**

5 Experiments and Results

We evaluate POTPTQ, our proposed two-step **Power-of-Two (PoT)** quantization framework, for weight-only post-training quantization (PTQ) of large language models. Experiments are conducted on LLaMA [22] and Llama2 [23] models with 7B, 13B, and 30B parameters. The method targets ultra-low precision quantization at 2 and 3-bit levels. All quantization and calibration procedures are performed on a single Tesla V100 GPU (32GB), while kernel benchmarks are additionally run on an RTX 4090 to assess inference efficiency.

5.1 Experimental Setup

Step 1 (Data-Agnostic Initialization): A grid search over the interval $[0, 2]$ with step size 0.01 is performed for each quantization group (group size = 128) to identify the optimal initial scale. **Step 2 (Data-Dependent Fine-Tuning):** Using 128 randomly sampled 2048-token sequences from WikiText-2 [18], we apply light fine-tuning. The learning rate is set to 1×10^{-3}, with weight decay 1×10^{-1}. Fine-tuning is run for 10 epochs (3-bit) and 40 epochs (2-bit).

5.2 Perplexity Evaluation

Table 1 presents perplexity results on WikiText-2 across multiple LLaMA model sizes. POTPTQ consistently achieves lower perplexity than existing PTQ methods, including RTN [6], GPTQ [9], AWQ [16], and OmniQuant [20], demonstrating its effectiveness for ultra-low-bit quantization.

To evaluate whether existing PTQ methods can directly adopt PoT quantization, we adapt RTN, GPTQ, AWQ, and OmniQuant to operate under power-of-two constraints and report their results in Table 2. These naïvely adapted methods suffer notable degradation in perplexity, confirming that uniform-to-PoT substitution without proper scale handling leads to suboptimal outcomes. In contrast, POTPTQ explicitly optimizes scales for PoT representation, yielding significantly better results.

Table 2: Perplexity of baseline PTQ methods adapted to PoT format versus our PoT method. Naive adaptations degrade performance; our method preserves accuracy.

Avg Bits	Model	Size	AWQ_POT	GPTQ_POT	OMNI_POT	PoTPTQ
3.125	LLaMA	7B	6.52	8.27×10^4	6.37	**6.25**
		13B	5.61	5.85×10^4	5.60	**5.50**
		30B	4.72	2.61×10^4	4.75	**4.58**
	Llama2	7B	6.49	NaN	6.46	**6.22**
		13B	5.43	6.41×10^4	5.45	**5.34**
2.125	LLaMA	7B	2.69×10^5	2.92×10^5	888	**10.86**
		13B	2.80×10^5	1.83×10^5	487	**8.54**
		30B	2.39×10^5	1.44×10^5	297	**7.47**
	Llama2	7B	2.24×10^5	2.78×10^5	3730	**12.80**
		13B	1.27×10^5	1.03×10^5	812	**9.18**

5.3 Harness Evaluation

Given that POTPTQ clearly surpasses RTN, GPTQ, and AWQ, and performs competitively with OmniQuant in terms of perplexity, we conduct a downstream evaluation using the Open LLM Leaderboard harness. This benchmark includes a range of QA and reasoning tasks beyond language modeling.

Table 3 compares POTPTQ and OmniQuant. Our method matches or outperforms OmniQuant on most tasks and achieves higher average performance, indicating that it preserves not only token-level modeling but also broader functional capabilities required in real-world deployments.

5.4 Ablation Study

To isolate the contributions of each step in our pipeline, we conduct an ablation study using 2-bit quantization on LLaMA-7B and LLaMA-13B. As shown in Table 4, Step 1 alone (data-agnostic initialization) achieves reasonable performance, indicating the efficacy of our scale grid search. Applying Step 2 (fine-tuning) without Step 1, however, results in inferior performance, likely due to poor initial scale estimates. The best performance is consistently achieved by combining both steps, confirming their complementary effects: Step 1 provides robust initialization, and Step 2 refines the solution with minimal calibration data.

5.5 Fine-Tuning Efficiency and Quantization Time

To evaluate the effectiveness of Step 2, we track the epoch-wise perplexity and loss on WikiText-2 using the 13B LLaMA model. As shown in Table 5, both perplexity and loss decrease consistently over

Table 3: Harness evaluation on six tasks comparing PoT and OmniQuant under 3.25-bit and 2.25-bit settings. Bold indicates better performance.

Model	Method	Avg Bits = 3.25							Avg Bits = 2.25						
		arc-c	arc-e	boolq	hs	piqa	wg	Avg	arc-c	arc-e	boolq	hs	piqa	wg	Avg
LLaMA 7B	Omni	35.6	**64.8**	**71.1**	53.9	77.2	64.5	61.2	26.7	**52.1**	62.2	**40.7**	67.2	55.5	50.7
	PoT	**35.8**	64.1	70.9	**54.3**	**77.5**	**65.2**	**61.3**	**28.1**	50.1	**64.4**	40.1	**67.9**	**57.3**	**51.3**
LLaMA 13B	Omni	39.8	**72.7**	**67.0**	56.8	77.2	68.7	63.7	**31.3**	**60.1**	63.1	**46.1**	**72.0**	61.8	55.7
	PoT	**40.7**	71.8	65.6	**57.0**	**78.8**	**70.3**	**64.0**	30.1	59.5	**66.1**	45.6	70.3	**62.7**	55.7
LLaMA 30B	Omni	46.0	74.1	71.2	**61.3**	79.6	74.1	67.7	32.4	**65.6**	66.1	49.9	72.3	**62.9**	58.2
	PoT	**47.2**	73.7	**71.6**	60.8	**80.1**	**75.0**	**68.1**	**33.8**	64.9	**66.8**	**50.2**	**73.1**	62.5	**58.5**
Llama2 7B	Omni	37.3	**67.6**	71.2	54.5	**76.5**	65.7	62.1	**26.0**	45.0	61.2	39.4	64.5	**54.4**	48.4
	PoT	**38.1**	66.3	**72.0**	**55.2**	76.3	**66.6**	**62.4**	25.8	**52.1**	**63.8**	**40.3**	**65.2**	54.2	**50.2**
Llama2 13B	Omni	41.9	**72.3**	69.9	57.8	78.0	67.7	64.6	**30.0**	**57.0**	63.7	**44.5**	68.0	53.1	52.7
	PoT	**42.5**	70.7	**70.1**	**58.0**	**79.2**	**68.3**	**64.8**	29.4	56.9	**68.2**	43.4	**68.8**	**56.9**	**53.9**

10 epochs with only 128 calibration sequences. This highlights the efficiency of our fine-tuning process, which requires minimal data while achieving steady improvement. By the 10th epoch, the 2-bit model achieves a perplexity of 12.90 and loss of 7.89, demonstrating that our output-level alignment objective (Eq. 14) leads to significantly improved model fidelity.

In terms of wall-clock performance, we evaluate the total quantization time required to complete both stages of our framework: Step 1 (data-agnostic grid search for scale initialization) and Step 2 (data-dependent scale refinement via lightweight fine-tuning). All experiments are conducted on a single NVIDIA Tesla V100 GPU with 32GB of memory.

For LLaMA-7B, the complete quantization pipeline takes approximately 0.71 hours (~43 minutes). This includes exhaustive scale grid search across weight groups and 10 epochs of fine-tuning using a small calibration set.

Notably, the entire quantization pipeline—spanning initialization, calibration, and weight reconstruction—can be executed within a few hours on commodity hardware, without any need for multi-GPU parallelism or model retraining. This makes our framework highly suitable for real-world deployments where turnaround time and hardware constraints are critical considerations.

5.6 Dequantization Speed

To measure inference-time efficiency, we benchmark our custom PoT dequantization kernel on both Tesla V100 and RTX 4090 GPUs. As detailed in Section 4.2, the kernel uses integer arithmetic to reconstruct weights efficiently from PoT encodings. Table 6 shows that our kernel achieves a $3.66\times$ speedup on V100 and a $1.48\times$ speedup on 4090, compared to standard FP16 dequantization. These results demonstrate that PoT quantization is not only accurate, but also highly efficient for hardware-accelerated inference.

5.7 Summary

POTPTQ delivers state-of-the-art performance for ultra-low-bit weight-only quantization in LLMs. It outperforms existing PTQ baselines under both uniform and PoT formats, generalizes well across datasets, and preserves downstream task capabilities. Furthermore, the approach is scalable, calibration-efficient, and inference-friendly—making it a strong candidate for deployment in real-world, resource-constrained environments.

Table 4: Ablation study on the effect of each quantization step for LLaMA models. Step 1 denotes scale initialization; Step 2 denotes calibration-based fine-tuning.

Model	Step 1 (Init)	Step 2 (Tuning)	Perplexity
LLaMA 7B	✗	✗	408,838.25
	✓	✗	20,135.70
	✗	✓	51.87
	✓	✓	**9.79**
LLaMA 13B	✗	✗	40,328.34
	✓	✗	8,267.57
	✗	✓	103.45
	✓	✓	**7.96**

Table 5: Epoch-wise perplexity and loss on WikiText-2 with LLaMA-13B using PoT quantization. Our output-aligned fine-tuning objective consistently reduces perplexity with only 128 calibration samples.

Epoch	3-bit PPL	3-bit Loss	2-bit PPL	2-bit Loss
1	6.85	6.21	1.01×10^6	25.75
2	5.93	3.86	3209.49	20.45
3	5.70	3.02	239.27	17.13
4	5.59	2.54	94.25	15.39
5	5.54	2.27	49.95	13.50
6	5.51	2.13	31.46	11.99
7	5.49	2.04	23.30	10.55
8	5.48	1.97	18.27	9.60
9	5.48	1.93	15.31	8.65
10	**5.48**	**1.90**	**12.90**	**7.89**

Table 6: Dequantization Efficiency: GPU Warp Cycles. Our PoT kernel achieves significantly lower warp latency across architectures.

GPU	Architecture	Uniform (FP16)	PoT (Ours)
Tesla V100	Volta	110	30 (3.67× faster)
RTX 4090	Ada Lovelace	98	60 (1.63× faster)

6 Conclusion

We presented a novel post-training quantization framework that leverages power-of-two (PoT) representations to enable efficient and hardware-friendly inference. Our method introduces a two-stage algorithm consisting of data-agnostic scale initialization and data-driven

fine-tuning, effectively addressing the accuracy limitations commonly observed in traditional PoT quantization schemes.

Through comprehensive experiments on LLaMA models at 2-bit and 3-bit precision, we demonstrate that our approach consistently outperforms existing PTQ methods in both perplexity and real-world deployment scenarios. In addition to its strong accuracy, our framework supports parallelizable grid search and lightweight calibration, making it practical for deployment on standard hardware without requiring retraining or large calibration datasets.

Furthermore, our optimized integer-only dequantization kernel significantly accelerates inference, achieving up to $3.67\times$ speedup on NVIDIA V100 and $1.63\times$ on RTX 4090 compared to traditional FP16-based methods. These results highlight the potential of PoT quantization as a scalable and effective solution for low-latency LLM deployment.

Acknowledgements

References

[1] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *EMNLP*, pages 4895–4901. Association for Computational Linguistics, 2023.

[2] Y. Bengio, N. Léonard, and A. C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] J. Chee, Y. Cai, V. Kuleshov, and C. D. Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

[5] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

[6] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c3ba4962c05c49636d4c6206a97e9c8a-Paper-Conference.pdf.

[7] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[8] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2359–2368, 2021.

[9] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. OPTQ: accurate quantization for generative pre-trained transformers. In *ICLR*. OpenReview.net, 2023.

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

[11] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *AAAI Conference on Artificial Intelligence*, 2023. URL https://api.semanticscholar.org/CorpusID:267095435.

[12] X. Li, B. Liu, Y. Yu, W. Liu, C. Xu, and V. Partovi Nia. S3: Sign-sparse-shift reparametrization for effective training of low-bit shift networks.

[13] X. Li, B. Liu, R. H. Yang, V. Courville, C. Xing, and V. P. Nia. Denseshift: Towards accurate and efficient low-bit power-of-two quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17010–17020, 2023.

[14] Y. Li, X. Dong, and W. Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *ICLR*. OpenReview.net, 2020.

[15] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu. {BRECQ}: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

[16] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

[17] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving, 2024.

[18] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016.

[19] D. Przewlocka-Rus, S. S. Sarwar, H. E. Sumbul, Y. Li, and B. De Salvo. Power-of-two quantization for low bitwidth and hardware compliant neural networks. *arXiv preprint arXiv:2203.05025*, 2022.

[20] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

[21] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[22] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[23] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[24] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *ArXiv*, abs/2304.09145, 2023. URL https://api.semanticscholar.org/CorpusID:258187503.

[25] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[26] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

[27] H. Yao, P. Li, J. Cao, X. Liu, C. Xie, and B. Wang. Rapq: Rescuing accuracy for power-of-two low-bit post-training quantization. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

[28] H. You, X. Chen, Y. Zhang, C. Li, S. Li, Z. Liu, Z. Wang, and Y. Lin. Shiftaddnet: A hardware-inspired deep network. *Advances in Neural Information Processing Systems*, 33:2771–2783, 2020.

[29] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Advances in Neural Information Processing Systems, 34:14555–14566, 2021.