D^{2}Quant: Accurate Low-bit Post-Training Weight Quantization for LLMs

Xianglong Yan[1,*], Chengzhu Bao[1,*], Zhiteng Li[1], Tianao Zhang[1],
Shaoqiu Zhang[1], Ruobing Xie[2], Xingwu Sun[2], Yulun Zhang[1,†]

[1]Shanghai Jiao Tong University
[2]Tencent Hunyuan

[*]Equal contribution. [†]Corresponding author.
yulun100@gmail.com

Abstract

Large language models (LLMs) deliver strong performance, but their high compute and memory costs make deployment difficult in resource-constrained scenarios. Weight-only post-training quantization (PTQ) is appealing, as it reduces memory usage and enables practical speedup without low-bit operators or specialized hardware. However, accuracy often degrades significantly in weight-only PTQ at sub-4-bit precision, and our analysis identifies two main causes: (1) down-projection matrices are a well-known quantization bottleneck, but maintaining their fidelity often requires extra bit-width; (2) weight quantization induces activation deviations, but effective correction strategies remain underexplored. To address these issues, we propose D^{2}Quant, a novel weight-only PTQ framework that improves quantization from both the weight and activation perspectives. On the weight side, we design a Dual-Scale Quantizer (DSQ) tailored to down-projection matrices, with an absorbable scaling factor that significantly improves accuracy without increasing the bit budget. On the activation side, we propose Deviation-Aware Correction (DAC), which incorporates a mean-shift correction within LayerNorm to mitigate quantization-induced activation distribution shifts. Extensive experiments show that D^{2}Quant achieves superior weight-only PTQ performance at sub-4-bit precision. The code is available at `https://github.com/XIANGLONGYAN/D2Quant`.

1 Introduction

Large language models (LLMs) (Dubey et al., 2024; Yang et al., 2025; Zhang et al., 2022) have achieved remarkable success in natural language processing (NLP), demonstrating strong capabilities in both understanding and generation. Yet, this progress is largely driven by scaling up model size. Consequently, modern LLM families such as Qwen (Yang et al., 2025) and LLaMA (Dubey et al., 2024) continue to grow to further improve performance. However, the high memory and compute costs of LLM inference make deployment challenging in resource-constrained settings, limiting their real-world use on edge and mobile devices.

Quantization is one of the most effective ways to compress LLMs and enable deployment. In conventional neural networks (Krishnamoorthi,

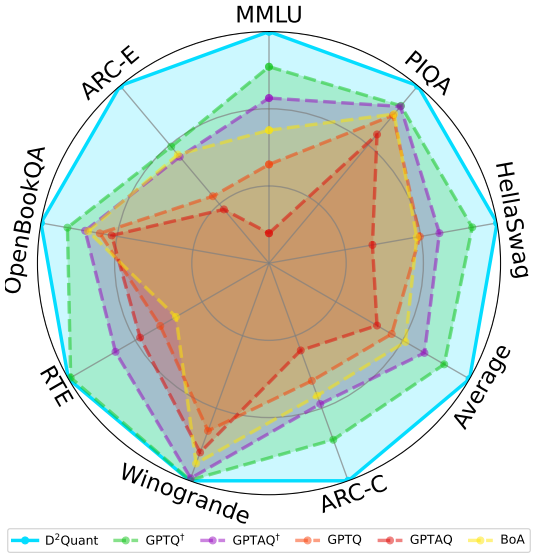
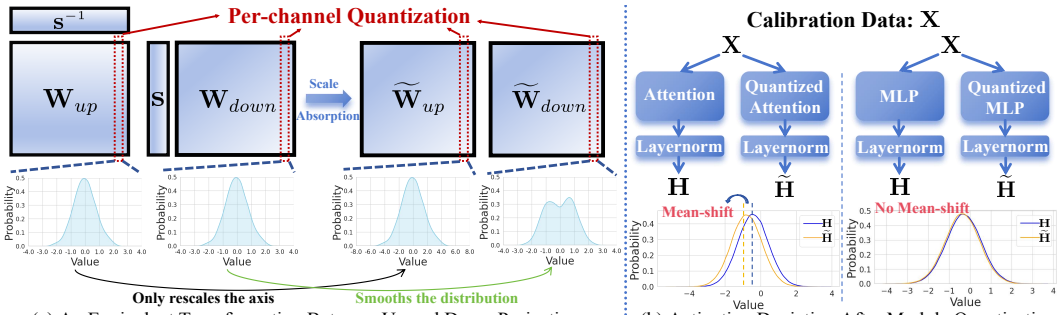


Figure 1: Performance comparison of weight-only methods on Qwen-3-8B with 2-bit quantization.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) An Equivalent Transformation Between Up and Down Projections (b) Activation Deviation After Module Quantization

Figure 2: **(a)** Equivalent transformation between up and down projections in per-channel quantization: smoothing can be applied to down projection, while up projection only introduces channel-wise scaling. **(b)** Activation deviation at the subsequent LayerNorm after quantizing attention and MLP: attention quantization causes a notable mean shift, while MLP introduces no significant shift.

2018; Esser et al., 2020; Chen et al., 2026; Wu et al., 2026), quantization is often applied to both weights and activations, and low-bit matrix multiplication is used to reduce compute and speed up inference. However, these gains typically depend on specialized operators and hardware support. In contrast, LLM inference is largely memory-bound, which makes weight-only quantization especially attractive. By compressing weights alone, it not only reduces the overall memory footprint, but also enables practical inference speedups by alleviating memory bandwidth bottlenecks, without requiring low-bit operators or specialized hardware.

Weight-only quantization for LLMs typically follows two paradigms: quantization-aware training (QAT) and post-training quantization (PTQ). In practice, PTQ is more widely adopted as it avoids retraining and thus requires significantly fewer computational and data resources. Representative PTQ methods, such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024b), perform well at 4-bit. However, reducing the bit-width below 4 often leads to a pronounced accuracy drop (see Fig. 1). To improve sub-4-bit PTQ, we make two key observations and analyses:

- It is widely recognized that down-projection matrices are highly sensitive to quantization. As shown in Fig. 2(a), we find that inserting an equivalent scaling transformation between the up- and down-projections is beneficial in a weight-only setting. It makes the down-projection easier to quantize while leaving the up-projection's quantization difficulty unchanged. This offers a principled way to improve down-projection accuracy without increasing bit budget.
- Although activations are not quantized in weight-only PTQ, they can still drift due to unavoidable weight quantization errors, which can severely affect model outputs. We measure activation deviations after quantizing the attention and MLP blocks. As shown in Fig. 2(b), we observe a clear mean shift at the post-attention LayerNorm after quantizing the attention module, while this behavior is much less evident after quantizing the MLP module. This discrepancy cues activation drift correction in weight-only quantization.

Based on these observations, we propose D^{2}Quant, a weight-only PTQ framework that improves sub-4-bit quantization from both weight and activation perspectives. **From the weight perspective**, we design a Dual-Scale Quantizer (DSQ), which reformulates the smoothing between up- and down-projection as a dual-scale quantization problem on the down-projection. By incorporating the additional scaling factor into the down-projection quantization process, we develop an efficient optimization scheme that enables the two scaling factors at different granularities to rapidly converge to their optimal values. Notably, the additional scaling factor can be fully folded into the preceding up-projection after quantization, improving down-projection fidelity with essentially no extra bit budget or inference overhead. **From the activation perspective**, we first introduce a *signal-to-noise ratio* to quantify activation drift, formalizing our earlier observation that attention quantization induces a pronounced mean shift at the post-attention LayerNorm, whereas the pre-attention LayerNorm exhibits less structured deviation after MLP quantization. Motivated by this, we propose Deviation-Aware Correction (DAC), which injects a lightweight deviation correction term into the post-attention LayerNorm to compensate for the quantization-induced mean shift.

Extensive experiments across multiple LLM families and evaluation metrics show that D^{2}Quant delivers superior performance for weight-only PTQ at sub-4-bit precision. As shown in Fig. 1, on Qwen-3-8B under 2-bit quantization, D^{2}Quant achieves an average accuracy of 57.22 over seven zero-shot tasks (vs. 54.05 for the state of the art (SOTA)).

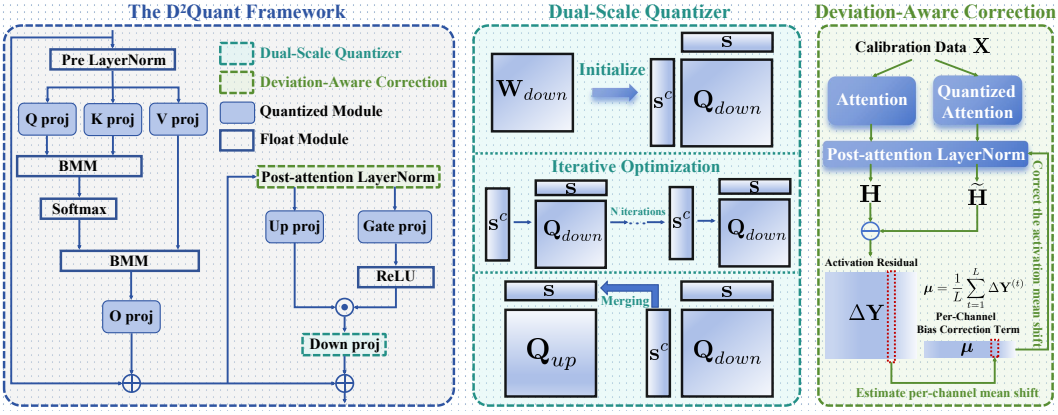


Figure 3: Overview of **D²Quant**. The left panel shows the D²Quant framework, which improves weight-only PTQ at both weight and activation levels. The middle shows the *Dual-Scale Quantizer*, which introduces an additional scale to refine the down-projection. The right depicts the *Deviation-Aware Correction*, which mitigates mean shift at post-attention LayerNorm via bias alignment.

Our main contributions are summarized as follows:

- At the weight level, we design a **Dual-Scale Quantizer (DSQ)** tailored to down-projection weight matrices, optimizing an absorbable auxiliary scaling factor to improve accuracy without increasing the bit budget.
- At the activation level, we propose **Deviation-Aware Correction (DAC)**, which performs mean-shift correction in the post-attention LayerNorm to mitigate activation drift, yielding improved model performance.
- By combining DSQ and DAC, **D²Quant** achieves superior sub-4-bit weight-only PTQ, effectively mitigating severe performance degradation in this regime.

2 Related Works

2.1 Large Language Model Quantization

Current LLM quantization techniques can be broadly categorized into quantization-aware training (QAT) and post-training quantization (PTQ). **QAT** (Shao et al., 2024; Du et al., 2024; Ashkboos et al., 2025; Liu et al., 2025b) integrates quantization into training, enabling the model to adapt to low-precision representations. To alleviate the data barrier in QAT, LLM-QAT (Liu et al., 2024b) introduces data-free distillation. EfficientQAT (Chen et al., 2025) improves training efficiency via a two-stage procedure. Moreover, several works push low-precision training to sub-2-bit regimes (Xu et al., 2024; Wang et al., 2023; Jo et al., 2024). However, QAT requires training and thus incurs substantial data and computational overhead, whereas **PTQ** (Yao et al., 2022; Wei et al., 2022; Lee et al., 2024; Wu et al., 2025) is training-free and considerably more resource-efficient. Recent studies further demonstrate that PTQ can achieve strong performance (Dettmers et al., 2022; Zhao et al., 2024; Lin et al., 2025). SmoothQuant (Xiao et al., 2023) introduces a smoothing operation that transfers the quantization difficulty from activations to weights, enabling near-lossless W8A8 quantization. A series of rotation-based methods (Ashkboos et al., 2024; Hu et al., 2025; Lin et al., 2024a; Liu et al., 2025a; Sun et al., 2025) further smooth outliers in activations, advancing PTQ toward lower-bit regimes. Notably, most of the above PTQ approaches quantize both weights and activations for low-bit inference. We discuss weight-only PTQ, a key PTQ branch, in the next paragraph.

2.2 Weight-Only Post-training Quantization

Since LLM inference is typically memory-bound, compressing weights alone not only reduces memory footprint but also accelerates inference by reducing memory traffic. It requires no specialized low-bit operators or hardware support, making weight-only PTQ (Dettmers et al., 2024; Kim et al., 2024; Chee et al., 2023; Zhang et al., 2026; Li et al., 2024) widely used in practical deployments. AWQ (Lin et al., 2024b) selectively protects salient weight channels based on activation statistics. GPTQ (Frantar et al., 2023) performs layer-wise quantization by minimizing output perturbation with an approximate Hessian. Building on GPTQ, GPTAQ (Li et al., 2025a) and QEP (Arai & Ichikawa, 2025) account for input errors, while BoA (Kim et al., 2025) improves Hessian estimation, further boosting accuracy. SliM-LLM (Huang et al., 2025) adopts mixed precision with greedy bit allocation, assigning more bits to more important weights. Several methods (Shang et al., 2024; Li et al., 2025b; Yan et al., 2026, 2025; Huang et al., 2024) further push weight-only PTQ to sub-2-bit

regimes. In addition, QuIP# (Tseng et al., 2024a), QTIP (Tseng et al., 2024b), GPTVQ (Baalen et al., 2024), and VPTQ (Liu et al., 2024a) adopt vector quantization (VQ), grouping weights into vectors and quantizing them jointly to better capture inter-weight correlations at extremely low bit widths. However, performance often degrades sharply once the bit-width drops below 4, limiting the practicality of weight-only PTQ in this regime. Our work focuses on weight-only PTQ and aims to improve performance under sub-4-bit quantization.

3 Method

In this section, we introduce D^{2}Quant, as illustrated in Fig. 3. Sec. 3.1 reviews low-bit quantization preliminaries and notation. Sec. 3.2 presents the Dual-Scale Quantizer (DSQ) for quantizing down-projection matrices, followed by Sec. 3.3, which introduces Deviation-Aware Correction (DAC) to address activation drift. Sec. 3.4 combines DSQ and DAC into the D^{2}Quant pipeline.

3.1 Preliminary

Low-Bit Quantization. To facilitate efficient storage and high-speed inference, low-bit uniform quantization is utilized to map floating-point tensors into discrete low-precision representations. Formally, given a weight tensor $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$, we apply b-bit per-channel quantization to derive an integer tensor $\mathbf{W}_q \in \mathcal{Q}^{C_{\text{out}} \times C_{\text{in}}}$, where $\mathcal{Q} = \{0, 1, \ldots, 2^b - 1\}$. The quantization is formulated as:

$$\mathbf{W}_q = \text{clip}\left(\left\lfloor \frac{\mathbf{W}}{\mathbf{s}} \right\rceil + \mathbf{z},\ 0,\ 2^b - 1\right), \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the rounding-to-nearest-integer operator, and $\text{clip}(\cdot)$ restricts the values to the range of $\mathcal{Q}$. The parameters $\mathbf{s} \in \mathbb{R}^{C_{\text{out}} \times 1}$ and $\mathbf{z} \in \mathcal{Q}^{C_{\text{out}} \times 1}$ represent the per-channel scale factors and zero-points, respectively, which are broadcast along the input-channel dimension. Specifically, the scale factor $\mathbf{s}$ and zero-point $\mathbf{z}$ are computed from the channel-wise dynamic range of $\mathbf{W}$ as:

$$\mathbf{s} = \frac{\max(\mathbf{W}) - \min(\mathbf{W})}{2^b - 1}, \quad \mathbf{z} = -\left\lfloor \frac{\min(\mathbf{W})}{\mathbf{s}} \right\rceil. \tag{2}$$

Given the quantized integer tensor $\mathbf{W}_q$ along with the corresponding scale factors $\mathbf{s}$ and zero-points $\mathbf{z}$, we recover its floating-point approximation via dequantization:

$$\widehat{\mathbf{W}} = \mathbf{s} \odot (\mathbf{W}_q - \mathbf{z}), \tag{3}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{s}, \mathbf{z}$ are broadcast to match the shape of $\mathbf{W}_q$.

3.2 Dual-Scale Quantizer

Equivalent Up–Down Scaling Transformation. The MLP module typically transforms the hidden states $\mathbf{X} \in \mathbb{R}^{L \times C_{\text{in}}}$ through a gated mechanism. Formally, given the input $\mathbf{X}$, the forward pass is defined as:

$$\mathbf{Y} = \left(\sigma(\mathbf{X}\mathbf{W}_{\text{gate}}^\top) \odot (\mathbf{X}\mathbf{W}_{\text{up}}^\top)\right) \mathbf{W}_{\text{down}}^\top, \tag{4}$$

where $\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}} \in \mathbb{R}^{H \times C_{\text{in}}}$ and $\mathbf{W}_{\text{down}} \in \mathbb{R}^{C_{\text{in}} \times H}$ denote the gate, up-projection, and down-projection matrices, respectively. A mathematically equivalent reparameterization can be introduced between the up- and down-projections via a per-channel scaling vector $\boldsymbol{\eta} \in \mathbb{R}^{1 \times H}$:

$$\widetilde{\mathbf{W}}_{\text{up}}^\top = \mathbf{W}_{\text{up}}^\top \text{diag}(\boldsymbol{\eta}),\ \widetilde{\mathbf{W}}_{\text{down}}^\top = \text{diag}(\boldsymbol{\eta})^{-1} \mathbf{W}_{\text{down}}^\top, \tag{5}$$

which results in the exact same computation:

$$\mathbf{Y} = \left(\sigma(\mathbf{X}\mathbf{W}_{\text{gate}}^\top) \odot (\mathbf{X}\widetilde{\mathbf{W}}_{\text{up}}^\top)\right) \widetilde{\mathbf{W}}_{\text{down}}^\top. \tag{6}$$

This transformation applies a shared scaling to the up- and down-projection matrices in opposite directions. Since both $\mathbf{W}_{\text{up}}$ and $\mathbf{W}_{\text{down}}$ are quantized with per-channel granularity, this transformation can have asymmetric effects: it can leave the quantization of $\mathbf{W}_{\text{up}}$ unaffected, as the scaling can be uniformly absorbed within each channel; meanwhile, it can smooth the distribution of $\mathbf{W}_{\text{down}}$, potentially reducing its dynamic range and easing quantization.

Dual-Scale Quantizer. To better leverage scaling flexibility, we formulate down-projection quantization as a dual-scale problem by embedding an additional (column-wise) scale directly into the quantization process, rather than applying it as static smoothing. We begin by revisiting standard per-channel quantization, as defined in Eqs. 1–3, where each weight tensor is quantized using channel-wise scale and zero-point parameters. For simplicity, we abstract this process as a generic quantization operator $Q(\cdot)$, yielding $\widehat{\mathbf{W}} = Q(\mathbf{W})$. To refine the quantized weights, we introduce an additional

column-wise scale factor $\mathbf{s}^c \in \mathbb{R}^{1\times H}$, resulting in the dual-scale quantized form:

$$\widehat{\mathbf{W}} = Q(\mathbf{W}) \odot \mathbf{s}^c. \tag{7}$$

Our objective is to minimize the reconstruction error between the original and quantized weights:

$$\min \left\| \mathbf{W} - \widehat{\mathbf{W}} \right\|_F^2 = \left\| \mathbf{W} - Q(\mathbf{W}) \odot \mathbf{s}^c \right\|_F^2 . \tag{8}$$

To efficiently solve this objective, we adopt an iterative optimization strategy. Specifically, we first freeze the quantization operator $Q(\cdot)$ and solve for the optimal $\mathbf{s}^c$ in closed form. Then, we fix $\mathbf{s}^c$ and update the quantized weights by applying Q to the normalized weights $\mathbf{W}/\mathbf{s}^c$. This process is repeated until convergence. In practice, the iterative procedure converges within a few steps and effectively integrates the column-wise scale into standard quantization. After down-projection quantization, since the up-projection has already been quantized with per-channel scales, the additional column-wise factor $\mathbf{s}^c$ can be directly merged by multiplying it into the existing scales. This preserves inference equivalence without introducing any runtime overhead.

3.3 Deviation-Aware Correction

Signal-to-Noise Ratio (SNR) Analysis. Based on our observations, weight-only quantization in transformer blocks induces activation shifts at subsequent LayerNorms. Quantizing attention causes a pronounced mean shift at the post-attention LayerNorm, while MLP quantization results in weaker, less structured deviations at the pre-attention LayerNorm of the next block. To quantify these effects, we introduce a signal-to-noise ratio (SNR) metric. Let $\mathbf{X} \in \mathbb{R}^{L\times H}$ denote the calibration input, where L is the token count and H the hidden dimension. Taking post-attention LayerNorm as an example, the full-precision output is:

$$\mathbf{Y}_{fp} = \text{PostAttnLN}(\text{Attention}(\mathbf{X})), \tag{9}$$

and the quantized counterpart as:

$$\mathbf{Y}_q = \text{PostAttnLN}(Q(\text{Attention}(\mathbf{X}))). \tag{10}$$

We define the activation deviation as:

$$\Delta\mathbf{Y} = \mathbf{Y}_{fp} - \mathbf{Y}_q, \tag{11}$$

which captures the activation shift at the post-attention LayerNorm caused by quantizing the attention module. Here, $\Delta\mathbf{Y} \in \mathbb{R}^{L\times H}$ contains the per-token deviations across L tokens and H-dimensional features. The mean and variance of the deviation across tokens are computed as:

$$\boldsymbol{\mu} = \frac{1}{L}\sum_{t=1}^{L} \Delta\mathbf{Y}^{(t)}, \quad \boldsymbol{\sigma}^2 = \frac{1}{L}\sum_{t=1}^{L} \left(\Delta\mathbf{Y}^{(t)} - \boldsymbol{\mu}\right)^2, \tag{12}$$

where all operations are element-wise over the feature dimension H, and broadcasting is applied as needed. The signal-to-noise ratio (SNR) for the i-th feature is then defined as:

$$\text{SNR}_i = \frac{\mu_i^2}{\sigma_i^2}. \tag{13}$$

A higher SNR indicates a consistent directional shift across tokens, while a lower SNR reflects unstructured or negligible deviation. Fig. 4 shows the average SNR across all layers for both pre-attention LayerNorm and post-attention LayerNorm on LLaMA-3-8B. We observe that the SNR of the post-attention LayerNorm is consistently and significantly higher than that of the pre-attention LayerNorm, confirming that pronounced mean shifts indeed occur after attention quantization.

Bias Alignment for Post-Attention LayerNorm. Guided by the SNR analysis, we apply deviation correction only at post-attention LayerNorm layers, where the activation shifts are strong and consistent across tokens. For each such layer, we estimate a correction bias $\boldsymbol{\mu}$ by computing the mean deviation $\Delta\mathbf{Y}$ across a small calibration set. This bias is then added to the quantized output during inference:

$$\mathbf{Y}_{\text{aligned}} = \mathbf{Y}_q + \boldsymbol{\mu}. \tag{14}$$

The correction term $\boldsymbol{\mu}$ is stored as an additional bias parameter inside the corresponding LayerNorm and is applied jointly during inference. Its parameter size and runtime overhead are negligible compared to the overall model.

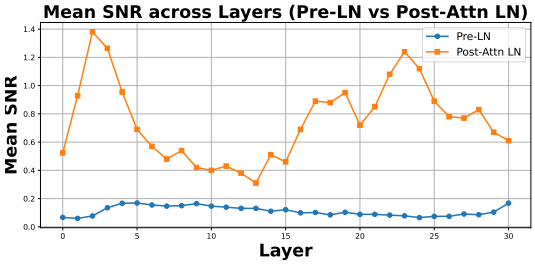


Figure 4: Post-attention LayerNorm shows higher SNR than pre-attention LayerNorm on LLaMA-3-8B, indicating more structured activation shifts.

Error Reduction via Deviation Correction. To evaluate deviation correction, we analyze its impact on activation reconstruction error. The deviation between full-precision and quantized LayerNorm outputs is $\Delta\mathbf{Y} = \mathbf{Y}_{fp} - \mathbf{Y}_q$. For the i-th feature dimension, the expected squared deviation (MSE) across tokens can be decomposed as:

$$\mathbb{E}[\|\Delta\mathbf{Y}_i\|^2] = \mu_i^2 + \sigma_i^2, \tag{15}$$

where μ_i and σ_i^2 are the mean and variance of the deviation in the i-th channel, respectively. We apply deviation correction by adding the bias $\boldsymbol{\mu}$ estimated from calibration data to the quantized output, which shifts the deviation to:

$$\Delta\mathbf{Y}_{\text{aligned}} = \mathbf{Y}_{fp} - \mathbf{Y}_{\text{aligned}} = \Delta\mathbf{Y} - \boldsymbol{\mu}. \tag{16}$$

The expected squared error then becomes:

$$\mathbb{E}[\|\Delta\mathbf{Y}_{\text{aligned},i}\|^2] = \sigma_i^2. \tag{17}$$

Thus, the relative error reduction is:

$$\frac{\mu_i^2 + \sigma_i^2 - \sigma_i^2}{\mu_i^2 + \sigma_i^2} = \frac{\mu_i^2}{\mu_i^2 + \sigma_i^2}. \tag{18}$$

This ratio quantifies how much error is eliminated by correcting the mean shift. Notably, this is exactly the form of the signal-to-noise ratio (SNR)–based reduction:

$$\frac{\mu_i^2}{\mu_i^2 + \sigma_i^2} = \frac{\text{SNR}_i}{1 + \text{SNR}_i}, \quad \text{where} \quad \text{SNR}_i = \frac{\mu_i^2}{\sigma_i^2}. \tag{19}$$

Hence, dimensions with higher SNR benefit more from deviation correction, since the dominant error from mean shift can be effectively removed.

Algorithm 1 Main Framework of D^{2}Quant: inner details of each function are provided in the appendix.

func D^{2}Quant($\mathcal{M}$, $\mathcal{X}$)
Input: $\mathcal{M}$ - Pre-trained model with L blocks
$\mathcal{X}$ - Calibration data
Output: $\widehat{\mathcal{M}}$ - Quantized model

$\widehat{\mathcal{M}} \leftarrow \mathcal{M}$ ▷ `Initialize quantized model`
for $l = 1$ to L **do**
  $X_l \leftarrow \text{GetPostAttnLNAct}(\mathcal{M}_l, \mathcal{X})$
  for $W \in \{W_q^l, W_k^l, W_v^l, W_o^l\}$ **do**
    $W \leftarrow \text{Quantizer}(W)$
  end for
  $\widehat{\mathcal{M}}_l \leftarrow \text{WriteBack}(W_q^l, W_k^l, W_v^l, W_o^l)$
  $\widehat{X}_l \leftarrow \text{GetPostAttnLNAct}(\widehat{\mathcal{M}}_l, \mathcal{X})$
  $\text{PostAttnLN}_l \leftarrow \text{DAC}(\text{PostAttnLN}_l, X_l, \widehat{X}_l)$
  $\widehat{\mathcal{M}}_l \leftarrow \text{WriteBack}(\text{PostAttnLN}_l)$
  for $W \in \{W_{up}^l, W_{gate}^l\}$ **do**
    $W \leftarrow \text{Quantizer}(W)$
  end for
  $W_{down}^l, s_c \leftarrow \text{DSQ}(W_{down}^l)$
  $W_{up}^l \leftarrow \text{MergeScale}(W_{up}^l, s_c)$
  $\widehat{\mathcal{M}}_l \leftarrow \text{WriteBack}(W_{up}^l, W_{gate}^l, W_{down}^l)$
  $\mathcal{X} \leftarrow \text{Forward}(\widehat{\mathcal{M}}_l, \mathcal{X})$ ▷ `Update calibration data for the next block`
end for
return $\widehat{\mathcal{M}}$

3.4 D^{2}Quant Pipeline

As shown in Algorithm 1, D^{2}Quant follows a block-wise weight-only PTQ pipeline integrating DSQ and DAC. For each block, we first collect full-precision post-attention LayerNorm activations as alignment targets. We then quantize the attention module, recompute the corresponding activations, and apply DAC to compensate for the quantization-induced mean shift. Next, we quantize the FFN up/gate projections and apply DSQ to the down-projection to improve its quantization accuracy. The additional column-wise scale is folded into the quantized up-projection for efficient inference. Finally, we propagate calibration data through the updated quantized block to prepare inputs for the next block.

Table 1: **Performance of 2-bit weight quantization on the Qwen-3 series.** We report perplexity (↓) on WikiText-2 and C4, accuracy (↑) on MMLU, and on seven commonsense tasks with their average (Avg.). † denotes methods with QuaRot rotation. Best results are in **bold**.

Model	Method	Wiki2(↓)	C4(↓)	MMLU(↑)	PIQA	Hella.	ARC-E	ARC-C	Wino.	RTE	OBQA	Avg.(↑)
Qwen-3 8B	FP16	9.72	15.43	72.96	77.64	74.90	80.81	57.00	68.03	77.98	41.80	68.31
	GPTQ	23.28	55.55	37.05	65.23	47.27	48.11	31.14	55.56	57.04	30.40	47.82
	GPTQ†	15.27	28.60	46.53	66.92	53.03	56.52	35.92	61.88	71.48	32.60	54.05
	GPTAQ	20.24	52.02	30.37	61.81	42.05	45.88	28.67	58.33	60.29	29.60	46.66
	GPTAQ†	15.61	29.91	43.49	66.81	49.42	54.80	33.02	61.64	64.26	31.40	51.62
	BoA	17.67	40.37	40.37	65.45	46.93	55.18	32.34	59.75	54.51	31.2	49.34
	D^{2}Quant	**14.10**	**25.96**	**49.94**	**70.51**	**55.76**	**66.84**	**39.25**	**61.96**	**71.84**	**34.40**	**57.22**
Qwen-3 14B	FP16	8.64	13.82	77.11	79.76	78.89	83.04	60.24	73.01	77.62	46.80	71.34
	GPTQ	12.71	28.68	53.20	71.93	**61.87**	62.84	39.08	64.72	49.46	36.40	55.19
	GPTQ†	12.92	38.23	46.90	71.98	54.43	67.30	40.78	66.30	66.43	37.40	57.80
	GPTAQ	12.91	28.19	51.33	71.06	60.46	66.58	41.47	66.77	61.73	39.00	58.15
	GPTAQ†	12.48	23.93	56.12	71.55	58.85	64.73	40.36	65.90	**81.59**	35.00	59.71
	BoA	14.10	29.33	44.95	70.24	54.25	58.04	35.49	65.51	74.01	32.80	55.76
	D^{2}Quant	**11.88**	**22.40**	**58.36**	**72.31**	60.98	**71.51**	**44.54**	**66.85**	74.73	**39.20**	**61.45**
Qwen-3 32B	FP16	7.61	12.45	80.76	82.05	82.62	83.21	61.26	73.01	76.17	46.00	72.05
	GPTQ	11.13	25.27	63.04	73.78	68.76	64.06	44.71	68.75	**74.01**	40.20	62.04
	GPTQ†	10.61	19.38	63.03	72.52	68.80	67.34	48.12	67.88	67.87	**41.60**	62.02
	GPTAQ	10.78	24.12	62.54	73.94	68.33	66.84	44.62	69.06	67.51	38.20	61.21
	GPTAQ†	10.44	20.03	62.61	74.32	68.50	67.85	46.33	68.51	69.31	39.80	62.09
	BoA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	D^{2}Quant	**9.71**	**17.10**	**64.31**	**76.12**	**70.47**	**75.51**	**51.11**	**70.09**	68.23	40.40	**64.56**

4 Experiments

4.1 Settings

Implementation Details. All experiments are conducted using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2020) on NVIDIA A800-80GB GPUs. Except for benchmark evaluations on 70B-scale models, which are performed using three GPUs, all quantization and evaluation experiments are conducted on a single GPU. We use 128 samples from the WikiText-2 dataset (Merity et al., 2017) with a sequence length of 2048 as the calibration set during quantization, and all quantized models adopt a fixed quantization block size of 128. We implement 15 iterations for Dual-Scale Quantizer (DSQ) to ensure the convergence of quantization parameters.

Baselines. We compare our method against GPTQ (Frantar et al., 2023) and GPTAQ (Li et al., 2025a), two representative weight-only PTQ approaches for LLMs. In addition, we implement the randomized Hadamard transform proposed in QuaRot (Ashkboos et al., 2024) as a weight pre-processing step on top of these baselines to smooth the weight distributions. We denote the resulting variants as GPTQ† (GPTQ+QuaRot) and GPTAQ† (GPTAQ+QuaRot). We further compare with BoA (Kim et al., 2025), a recent PTQ method that uses more accurate Hessian estimation.

Models and Evaluation. We evaluate our method on a range of pre-trained LLMs, including LLaMA-3 (8B/70B) (Dubey et al., 2024), LLaMA-3.1 (8B/70B), and Qwen-3 (8B/14B/32B) (Yang et al., 2025). We assess quantized models using both language modeling and downstream benchmarks. We report perplexity on WikiText-2 (Merity et al., 2017) and C4 (Raffel et al., 2020) with a sequence length of 2048 tokens, and measure zero-shot accuracy on PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), ARC-Easy/Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020), RTE (Chakrabarty et al., 2021), and OpenBookQA (Mihaylov et al., 2018). We evaluate on MMLU (Hendrycks et al., 2021), a multi-domain benchmark for knowledge-intensive reasoning.

4.2 Main Results

Perplexity Evaluation. First, we evaluate the language modeling performance of D^{2}Quant under 2-bit weight quantization. Tab. 1 and Tab. 2 report perplexity on WikiText-2 and C4 across different model families. Compared to baselines including GPTQ, GPTQ†, GPTAQ, GPTAQ†, and BoA, D^{2}Quant consistently achieves lower perplexity at all scales. On Qwen-3-8B, it reduces WikiText-2 perplexity from 20.24 (GPTAQ) to 14.10 and C4 from 52.02 to 25.96. On Qwen-3-14B, it further lowers C4 perplexity to 22.40 versus 28.19 with GPTAQ. For Qwen-3-32B, D^{2}Quant achieves 9.71 (WikiText-2) and 17.10 (C4), outperforming all baselines by a large margin. Similar trends hold for LLaMA models: On LLaMA-3.1-8B, D^{2}Quant obtains 11.54 on WikiText-2 and 27.19 on C4, substantially better than GPTQ (18.61 / 120.14) and GPTAQ (14.42 / 53.12). These consistent gains across diverse model families clearly demonstrate the robustness and effectiveness of D^{2}Quant in preserving language modeling quality under aggressive 2-bit quantization.

Table 2: **2-bit weight quantization results on the LLaMA-3/3.1 series.** Perplexity (↓) on WikiText-2/C4 and accuracy (↑) on MMLU and seven commonsense tasks, with the average (Avg.) over the seven tasks. † denotes methods with QuaRot rotation. Best results are in **bold**.

Model	Method	Wiki2(↓)	C4(↓)	MMLU(↑)	PIQA	Hella.	ARC-E	ARC-C	Wino.	RTE	OBQA	Avg.(↑)
LLaMA-3 8B	FP16	6.14	9.44	62.16	80.69	79.17	77.86	53.16	73.32	67.87	45.00	68.15
	GPTQ	17.33	67.14	23.27	54.79	42.95	31.31	22.18	54.14	52.71	30.20	41.18
	GPTQ†	28.97	79.97	23.10	55.82	34.74	34.55	21.33	53.51	52.71	24.80	39.64
	GPTAQ	14.17	132.13	23.02	57.67	40.41	35.23	23.29	56.35	52.35	27.00	41.76
	GPTAQ†	14.28	35.91	24.46	**62.68**	47.36	**45.58**	**27.99**	57.62	**54.87**	30.20	**46.61**
	BoA	23.13	66.96	22.95	58.11	39.15	39.27	25.00	55.33	52.71	28.8	42.62
	D^{2}Quant	**11.88**	**34.62**	**30.02**	60.23	**50.77**	42.13	**27.99**	**60.69**	52.71	**30.60**	46.45
LLaMA-3 70B	FP16	2.86	7.17	75.15	84.44	84.97	86.15	64.51	80.58	68.59	48.40	73.95
	GPTQ	9.31	53.63	25.23	**74.65**	46.80	63.59	39.25	57.62	53.07	29.00	52.00
	GPTQ†	19.76	49.56	23.34	52.07	42.22	29.88	21.42	54.93	52.71	31.6	40.69
	GPTAQ	9.18	27.26	**40.26**	74.48	**62.00**	62.42	36.69	61.25	**56.32**	32.4	55.08
	GPTAQ†	15.97	41.45	24.78	56.37	44.05	32.53	20.39	55.72	52.71	29.6	41.62
	BoA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	D^{2}Quant	**9.04**	**24.92**	32.95	72.31	56.82	**64.60**	**40.10**	**66.69**	53.43	**37.4**	**55.91**
LLaMA-3.1 8B	FP16	6.24	9.54	63.32	81.07	78.90	81.27	53.41	73.80	70.40	44.80	69.09
	GPTQ	18.61	120.14	23.76	59.41	44.77	42.13	25.68	53.75	52.35	30.20	44.04
	GPTQ†	24.60	77.78	23.17	60.66	34.98	38.68	22.95	53.59	52.35	25.80	41.29
	GPTAQ	14.42	53.12	22.95	59.47	39.73	43.73	27.13	54.46	52.71	30.20	43.92
	GPTAQ†	13.61	32.92	24.58	64.20	46.47	47.43	27.90	56.43	53.43	30.00	46.55
	BoA	23.27	61.30	23.01	61.21	41.36	41.67	24.57	55.41	52.35	26.4	43.28
	D^{2}Quant	**11.54**	**27.19**	**28.85**	**69.26**	**53.73**	**52.48**	**32.25**	**60.85**	**56.32**	**32.40**	**51.04**
LLaMA-3.1 70B	FP16	2.81	7.11	75.31	84.28	85.00	86.53	64.85	79.24	70.04	48.00	73.99
	GPTQ	12.09	248.64	38.11	71.65	56.48	**62.92**	38.40	61.09	**57.04**	32.60	54.31
	GPTQ†	14.00	34.39	26.45	60.83	55.42	43.10	24.83	58.80	51.26	28.00	46.03
	GPTAQ	16.38	72.04	23.03	62.35	43.58	45.12	27.13	53.99	54.51	28.40	45.01
	GPTAQ†	12.53	28.52	30.19	64.09	50.00	42.47	24.91	57.30	54.15	31.40	46.33
	BoA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	D^{2}Quant	**8.63**	**17.74**	**40.46**	**73.94**	**61.87**	62.58	**39.93**	**69.61**	53.79	**38.00**	**57.10**

Zero-Shot Accuracy Evaluation. We further evaluate D^{2}Quant on seven representative zero-shot reasoning benchmarks. As shown in Tab. 1 and Tab. 2, D^{2}Quant outperforms existing baselines across most tasks and model sizes. On the Qwen series, for example, D^{2}Quant raises the average accuracy of Qwen-3-32B from 61.21 (GPTAQ) to 64.56, achieving a +3.35 gain. On LLaMA-3.1-8B, the improvement is even more pronounced, with average accuracy rising from 44.04 (GPTQ) to 51.04, a +7.00 gain. These consistent improvements clearly demonstrate the strong ability of D^{2}Quant to retain robust reasoning and commonsense capabilities under aggressive 2-bit quantization.

MMLU Evaluation. To further assess the reasoning and knowledge retention of quantized models, we evaluate D^{2}Quant on the MMLU benchmark, with results summarized in Tab. 1 and Tab. 2. D^{2}Quant improves MMLU accuracy across nearly all model scales and architectures. On Qwen-3-32B, it achieves a gain of 1.27 points over GPTQ, while on LLaMA-3.1-8B the improvement reaches 5.09 points. These results indicate that D^{2}Quant is highly effective at mitigating knowledge degradation from aggressive 2-bit weight quantization, leading to stronger factual and reasoning performance in general-purpose language tasks without requiring any task-specific adaptation.

4.3 Ablation Study

Effect of DSQ and DAC Components. Table 3 reports a component-wise ablation on Qwen-3-8B under 2-bit quantization. We evaluate two perplexity metrics (WikiText-2 and C4), the MMLU benchmark, and an average accuracy score across seven zero-shot classification tasks (Acc). Introducing Dual-Scale Quantizer (DSQ) yields consistent gains across all metrics, notably improving MMLU by +3.48 and average accuracy by +3.27. Deviation-Aware Correction (DAC) also provides consistent gains, with notable improvements in downstream accuracy. When combined, our D^{2}Quant (DSQ+DAC) achieves the best results across all benchmarks.

Impact of Calibration Set Size for DAC. Table 4 presents the performance of DAC with varying calibration set sizes on Qwen-3-8B using 2-bit quantization. The 16-sample setting yields worse results, while larger calibration sets generally provide better performance. Among the tested sizes, 128 samples achieve the lowest WikiText-2 and C4 perplexities and the highest MMLU accuracy, whereas 64 and 256 samples tie for the highest

Table 3: **Effect of DSQ and DAC Components.** Component-wise breakdown analysis of DSQ and DAC on Qwen-3-8B (2-bit).

Model	Method	Wiki2(↓)	C4(↓)	MMLU(↑)	Acc(↑)
Q3-8B	Baseline	14.72	27.47	45.49	53.94
	+DSQ	14.41	26.95	48.97	57.21
	+DAC	14.63	27.00	45.82	54.63
	+DSQ+DAC	**14.10**	**25.96**	**49.94**	**57.22**

Table 4: **Impact of Calibration Set Size for DAC.** DAC performance with varying calibration sizes on Qwen-3-8B (2-bit).

Model	Method (Cal. Size)	Wiki2↓	C4↓	MMLU↑	Acc↑
Q3-8B	Baseline	14.72	27.47	45.49	53.94
	+DAC (16)	15.75	31.15	42.06	52.51
	+DAC (32)	14.86	27.40	49.44	56.07
	+DAC (64)	14.71	26.83	48.39	**57.80**
	+DAC (128)	**14.10**	**25.96**	**49.94**	57.22
	+DAC (256)	14.19	26.09	49.67	**57.80**

Table 5: **Design Ablation Study for DSQ.** Comparison of static smoothing and DSQ at different iterations on Qwen-3-8B (2-bit).

Model	Method	Wiki2↓	C4↓	MMLU↑	Acc↑
Q3-8B	Baseline	14.72	27.47	45.49	53.94
	+Static Smooth	14.85	27.72	44.99	52.28
	+DSQ (iterations=0)	15.73	31.90	46.05	53.71
	+DSQ (iterations=1)	14.90	28.62	48.70	57.01
	+DSQ (iterations=3)	14.66	26.98	48.76	57.15
	+DSQ (iterations=15)	**14.41**	**26.95**	**48.97**	**57.21**

average accuracy. We therefore use 128 samples as a practical balance between calibration cost and correction accuracy.

Design Ablation for DSQ. Table 5 presents a design ablation of DSQ on Qwen-3-8B under 2-bit quantization. Static smoothing applied to down-projection weights does not consistently improve performance and can slightly degrade both perplexity and accuracy, suggesting that fixed rescaling is insufficient to address quantization distortions. In contrast, DSQ dynamically incorporates column-wise scaling into the quantization objective, yielding consistent improvements across tasks. The results also highlight the importance of iterative refinement: performance improves steadily with more iterations and saturates around 15, indicating convergence. We thus adopt 15 iterations as the default setting, balancing effectiveness and overall efficiency.

4.4 Time and Memory Analyses

As shown in Fig. 5, D^{2}Quant exhibits favorable efficiency in both model size and quantization time. In terms of model size, our method achieves an almost identical footprint to GPTQ under the same bit-width, since the additional parameters introduced by adding bias correction in the LayerNorm are negligible compared to the overall model size. Regarding quantization time, D^{2}Quant incurs a modest overhead relative to GPTQ and GPTAQ, mainly due to the LayerNorm updates and the iterative DSQ optimization on the down-projection. Nevertheless, it remains faster than BoA, which relies on expensive Hessian-based optimization, demonstrating a strong balance between performance and efficiency. Thus, D^{2}Quant enables scalable LLM deployment, making it particularly well-suited for the growing demands of LLMs.

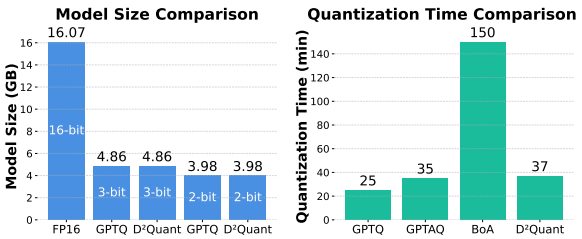


Figure 5: Comparison of model size and quantization time across PTQ methods on LLaMA-3-8B.

5 Conclusion

In this work, we revisit weight-only PTQ for LLMs at sub-4-bit precision and make two key observations: (1) a smoothing-equivalent transformation between the up- and down-projection significantly eases down-projection quantization without affecting the up-projection; (2) attention module quantization induces a pronounced mean shift at the post-attention LayerNorm. Building on these insights, we propose D^{2}Quant, a unified weight-only PTQ framework that improves sub-4-bit quantization performance. On the weight side, the proposed Dual-Scale Quantizer (DSQ) improves the robustness of down-projection quantization without increasing the bit budget or inference overhead. On the activation side, Deviation-Aware Correction (DAC) mitigates quantization-induced activation shifts in post-attention LayerNorm. Extensive experiments across multiple LLM families and evaluation benchmarks demonstrate that D^{2}Quant consistently outperforms prior SOTA weight-only PTQ methods in the sub-4-bit regime. Importantly, D^{2}Quant is composed of structurally simple components, many of which are absorbable, making the framework easy to integrate into mainstream inference pipelines. More broadly, this work suggests that effective weight-only PTQ requires jointly preserving critical weight structures and correcting quantization-induced activation shifts. We hope this perspective offers useful insights for developing more robust and deployable low-bit quantization methods, especially as LLMs continue to scale.

Acknowledgments

This work was supported by the CCF-Tencent Rhino-Bird Open Research Fund and the CAAI-Tencent Rhino-Bird Open Research Fund.

References

Arai, Y. and Ichikawa, Y. Quantization Error Propagation: Revisiting Layer-Wise Post-Training Quantization. In *NeurIPS*, 2025.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs. In *NeurIPS*, 2024.

Ashkboos, S., Nikdan, M., Tabesh, S., Castro, R. L., Hoefler, T., and Alistarh, D. HALO: Hadamard-Assisted Lower-Precision Optimization for LLMs. In *NeurIPS*, 2025.

Baalen, M. v., Kuzmin, A., Koryakovskiy, I., Nagel, M., Couperus, P., Bastoul, C., Mahurin, E., Blankevoort, T., and Whatmough, P. GPTVQ: The Blessing of Dimensionality for LLM Quantization. In *Workshop on Efficient Systems for Foundation Models II @ ICML*, 2024.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

Chakrabarty, T., Ghosh, D., Poliak, A., and Muresan, S. Figurative language in recognizing textual entailment. In *ACL*, 2021.

Chee, J., Cai, Y., Kuleshov, V., and Sa, C. D. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. In *NeurIPS*, 2023.

Chen, M., Shao, W., Xu, P., Wang, J., Gao, P., Zhang, K., and Luo, P. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models. In *ACL*, 2025.

Chen, Z., Yang, Z., Liu, X., Zhang, W., Wang, M., Fu, Y., Kong, L., and Zhang, Y. BinaryDemoire: Moiré-Aware Binarization for Image Demoiréing. *arXiv preprint arXiv:2602.03176*, 2026.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. In *NeurIPS*, 2022.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. In *ICLR*, 2024.

Du, D., Zhang, Y., Cao, S., Guo, J., Cao, T., Chu, X., and Xu, N. BitDistiller: Unleashing the Potential of Sub-4-Bit LLMs via Self-Distillation. In *ACL*, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. In *ICLR*, 2020.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *ICLR*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *ICLR*, 2021.

Hu, X., Cheng, Y., Yang, D., Xu, Z., Yuan, Z., Yu, J., Xu, C., Jiang, Z., and Zhou, S. OSTQuant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. In *ICLR*, 2025.

Huang, W., Liu, Y., Qin, H., Li, Y., Zhang, S., Liu, X., Magno, M., and Qi, X. BILLM: Pushing the limit of post-training quantization for LLMs. In *ICML*, 2024.

Huang, W., Qin, H., Liu, Y., Li, Y., Liu, Q., Liu, X., Benini, L., Magno, M., Zhang, S., and Qi, X. SliM-LLM: Salience-Driven Mixed-Precision Quantization for Large Language Models. In *ICML*, 2025.

Jo, D., Kim, T., Kim, Y., and Kim, J.-J. Mixture of scales: Memory-efficient token-adaptive binarization for large language models. In *NeurIPS*, 2024.

Kim, J., Kim, H.-y., Cho, E., Lee, C., Kim, J., and Jeon, Y. BOA: Attention-aware Post-training Quantization without Backpropagation. In *ICML*, 2025.

Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. SqueezeLLM: Dense-and-Sparse Quantization. In *ICML*, 2024.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Lee, C., Jin, J., Kim, T., Kim, H., and Park, E. OWQ: Outlier-Aware Weight Quantization for Efficient Fine-Tuning and Inference of Large Language Models. In *AAAI*, 2024.

Li, L., Li, Q., Zhang, B., and Chu, X. Norm Tweaking: High-performance Low-bit Quantization of Large Language Models. In *AAAI*, 2024.

Li, Y., Yin, R., Lee, D., Xiao, S., and Panda, P. GPTAQ: Efficient Finetuning-Free Quantization for Asymmetric Calibration. In *ICML*, 2025a.

Li, Z., Yan, X., Zhang, T., Qin, H., Xie, D., Tian, J., Kong, L., Zhang, Y., Yang, X., et al. ARB-LLM: Alternating refined binarizations for large language models. In *ICLR*, 2025b.

Lin, H., Xu, H., Wu, Y., Cui, J., Zhang, Y., Mou, L., Song, L., Sun, Z., and Wei, Y. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *NeurIPS*, 2024a.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*, 2024b.

Lin, Y., Tang, H., Yang, S., Zhang, Z., Xiao, G., Gan, C., and Han, S. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *MLSys*, 2025.

Liu, Y., Wen, J., Wang, Y., Ye, S., Zhang, L. L., Cao, T., Li, C., and Yang, M. VPTQ: Extreme Low-bit Vector Post-Training Quantization for Large Language Models. In *EMNLP*, 2024a.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. In *ACL*, 2024b.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., and Blankevoort, T. SpinQuant: LLM quantization with learned rotations. In *ICLR*, 2025a.

Liu, Z., Zhao, C., Huang, H., Chen, S., Zhang, J., Zhao, J., Roy, S., Jin, L., Xiong, Y., Shi, Y., et al. ParetoQ: Improving scaling laws in extremely low-bit LLM quantization. In *NeurIPS*, 2025b.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *ICLR*, 2017.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., and et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. WinoGrande: An adversarial winograd schema challenge at scale. In *AAAI*, 2020.

Shang, Y., Yuan, Z., Wu, Q., and Dong, Z. PB-LLM: Partially Binarized Large Language Models. In *ICLR*, 2024.

Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., and Luo, P. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. In *ICLR*, 2024.

Sun, Y., Liu, R., Bai, H., Bao, H., Zhao, K., Li, Y., Hu, J., Yu, X., Hou, L., Yuan, C., et al. FlatQuant: Flatness matters for LLM quantization. In *ICML*, 2025.

Tseng, A., Chee, J., Sun, Q., Kuleshov, V., and Sa, C. D. QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks. In *ICML*, 2024a.

Tseng, A., Sun, Q., Hou, D., and Sa, C. D. QTIP: Quantization with Trellises and Incoherence Processing. In *NeurIPS*, 2024b.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. BitNet: Scaling 1-bit Transformers for Large Language Models. *arXiv preprint arXiv:2310.11453*, 2023.

Wei, X., Gong, R., Li, Y., Liu, X., and Yu, F. QDrop: Randomly Dropping Quantization for Extremely Low-bit Post-Training Quantization. In *ICLR*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020.

Wu, J., Li, Z., Hui, Z., Zhang, Y., Kong, L., and Yang, X. QuantCache: Adaptive importance-guided quantization with hierarchical latent and layer caching for video generation. In *ICCV*, 2025.

Wu, T., Chen, Z., Xu, C., Chai, B., Guo, Y., Liu, Y., Kong, L., and Zhang, Y. LSGQuant: Layer-Sensitivity Guided Quantization for One-Step Diffusion Real-World Video Super-Resolution. In *ICML*, 2026.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *ICML*, 2023.

Xu, Y., Han, X., Yang, Z., Wang, S., Zhu, Q., Liu, Z., Liu, W., and Che, W. OneBit: Towards extremely low-bit large language models. In *NeurIPS*, 2024.

Yan, X., Zhang, T., Li, Z., Qin, H., and Zhang, Y. Progressive binarization with semi-structured pruning for LLMs. *arXiv preprint arXiv:2502.01705*, 2025.

Yan, X., Bao, C., Li, Z., Zhang, T., Yang, K., Qin, H., Xie, R., Sun, X., and Zhang, Y. PT^2-LLM: Post-training ternarization for large language models. In *ICLR*, 2026.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yao, Z., Aminabadi, R. Y., Zhang, M., Wu, X., Li, C., and He, Y. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. In *NeurIPS*, 2022.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *ACL*, 2019.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., and Chen. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhang, T., Li, Z., Yan, X., Qin, H., Guo, Y., and Zhang, Y. Quant-dLLM: Post-training extreme low-bit quantization for diffusion large language models. In *ICLR*, 2026.

Zhao, Y., Lin, C.-Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., and Kasikci, B. Atom: Low-bit quantization for efficient and accurate LLM serving. In *MLSys*, 2024.

A More Experimental Results

As shown in Table 6, D^{2}Quant demonstrates strong performance on both language modeling and downstream tasks at 3-bit precision. Compared with GPTQ, GPTAQ and their combination with QuaRot, it reduces perplexity on WikiText-2 and C4 while boosting accuracy on MMLU, and zero-shot tasks like PIQA, HellaSwag and others. On Qwen-3-8B, it lowers perplexity of WikiText-2/C4 from 11.01/17.73 (GPTAQ) to 10.45/16.71; on Qwen-3-14B, C4 perplexity decreases from 15.33 (GPTQ) to 15.03; and on Qwen-3-32B, C4 perplexity decreases to 13.19, outperforming all baselines. For LLaMA models, similar results are observed, for instance, on LLaMA-3.1-8B, D^{2}Quant reduces WikiText-2/C4 perplexity from 7.59/13.18 (GPTAQ) to 7.12/12.18. We further evaluate D^{2}Quant on seven representative zero-shot reasoning benchmarks and the MMLU benchmark to assess both commonsense reasoning and knowledge retention. On LLaMA-3.1-8B, D^{2}Quant raises the average accuracy to 65.88, achieving better performance over other methods. Moreover, D^{2}Quant improves MMLU accuracy of Qwen-3-32B to 77.92, surpassing GPTQ(77.52) and GPTAQ(77.77). These results clearly demonstrate that D^{2}Quant's deviation-aware correction and dual-scale quantization effectively preserves both language modeling quality and downstream knowledge under 3-bit quantization.

Table 6: **Performance of 3-bit weight quantization on LLaMA-3, LLaMA-3.1 and Qwen-3 series.** We report perplexity (↓) on WikiText-2 and C4, accuracy (↑) on MMLU, and on seven commonsense tasks with their average (Avg.). † denotes methods with QuaRot rotation.

Model	Method	Wiki2(↓)	C4(↓)	MMLU(↑)	PIQA	Hella.	ARC-E	ARC-C	Wino.	RTE	OBQA	Avg(↑)
L3-8B	FP16	6.14	9.44	62.16	80.69	79.17	77.86	53.16	73.32	67.87	45.00	68.15
	GPTQ	7.91	13.73	54.64	76.22	74.79	67.00	42.49	71.03	65.70	40.60	62.55
	GPTQ†	7.60	12.59	53.27	75.08	73.76	70.96	43.69	69.93	64.98	41.20	62.80
	GPTAQ	7.73	13.23	55.33	76.39	74.78	66.16	43.52	69.85	61.37	42.00	62.01
	GPTAQ†	7.18	12.34	54.91	77.37	74.59	75.72	47.95	71.19	63.54	40.60	64.42
	D^{2}Quant	7.10	12.25	50.95	78.35	74.64	73.57	46.25	71.82	58.48	42.00	63.59
L3.1-8B	FP16	6.24	9.54	63.32	81.07	78.90	81.27	53.41	73.80	70.40	44.80	69.09
	GPTQ	7.69	12.75	55.67	78.40	74.42	74.16	46.42	73.16	64.26	42.20	64.72
	GPTQ†	7.50	12.73	55.90	75.79	74.19	70.16	43.17	69.22	65.70	42.00	62.89
	GPTAQ	7.59	13.18	56.08	77.91	74.55	73.15	47.44	70.56	59.21	42.20	63.57
	GPTAQ†	7.22	12.29	56.56	77.64	74.14	76.85	49.57	71.03	64.26	41.40	64.98
	D^{2}Quant	7.12	12.18	54.40	78.84	74.31	78.07	48.46	70.64	67.87	43.00	65.88
Q3-8B	FP16	9.72	15.43	72.96	77.64	74.90	80.81	57.00	68.03	77.98	41.80	68.31
	GPTQ	10.86	17.44	67.98	75.41	70.59	76.05	50.00	67.40	77.98	42.00	65.63
	GPTQ†	10.45	16.86	67.95	76.17	70.43	75.67	52.56	66.54	72.20	41.80	65.05
	GPTAQ	11.01	17.73	67.19	75.73	69.40	72.94	47.87	67.96	78.34	40.40	64.66
	GPTAQ†	10.55	16.87	68.26	76.22	71.20	78.32	54.69	67.09	76.90	39.60	66.29
	D^{2}Quant	10.45	16.71	68.30	77.42	70.95	80.01	55.89	67.72	74.37	40.80	66.74
Q3-14B	FP16	8.64	13.82	77.11	79.76	78.89	83.04	60.24	73.01	77.62	46.80	71.34
	GPTQ	9.33	15.33	72.62	78.24	76.21	80.68	56.83	72.45	81.59	45.60	70.23
	GPTQ†	9.25	15.05	73.59	78.73	75.45	79.88	58.79	72.53	80.51	44.00	69.98
	GPTAQ	9.34	15.40	72.73	77.91	76.01	82.07	57.25	71.67	79.78	44.60	69.90
	GPTAQ†	9.26	14.97	72.90	78.13	75.47	80.85	57.59	72.85	78.70	43.40	69.57
	D^{2}Quant	9.19	15.03	73.17	78.94	75.35	81.82	59.64	72.61	77.26	45.40	70.15
Q3-32B	FP16	7.61	12.45	80.76	82.05	82.62	83.21	61.26	73.01	76.17	46.00	72.05
	GPTQ	8.38	14.39	77.52	79.49	80.49	79.12	60.15	72.45	80.87	45.40	71.14
	GPTQ†	7.99	13.25	77.23	79.60	80.37	82.79	61.35	69.22	76.53	48.20	71.15
	GPTAQ	8.25	14.19	77.77	79.16	80.35	81.27	58.62	73.40	77.62	46.80	71.03
	GPTAQ†	8.03	13.28	77.57	80.52	79.94	81.82	58.87	72.38	78.70	45.80	71.15
	D^{2}Quant	8.00	13.19	77.92	79.43	80.04	82.28	59.98	72.69	76.90	45.60	70.99

B Detailed Pseudocode

We provide more detailed pseudocode in this section to supplement the high-level descriptions presented in the main text. These expanded implementations clarify the specific steps and logic involved in each component of our method.

Algorithm 2 Dual-Scale Quantizer (DSQ)

Input: Down-projection weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$

Output: Quantized weight matrix $\hat{\mathbf{W}}$ and column scale s_c

1: **Initialization:** initialize column scale s_c, binary matrices $B_0, B_1 \in \{\pm 1\}^{m \times n}$, and row scales $r_0, r_1 \in \mathbb{R}^m$

2: **for** $t = 1$ **to** max_iter **do**

3: **Update row scales** (r_0, r_1) independently for each row $i = 1, \ldots, m$:

$$\begin{bmatrix} r_{0,i} \\ r_{1,i} \end{bmatrix} = (A_i D_i - B_{b,i}^2)^{-1} \begin{bmatrix} D_i & -B_{b,i} \\ -B_{b,i} & A_i \end{bmatrix} \begin{bmatrix} p_i \\ q_i \end{bmatrix}$$

4: **Update shared column scales** $s_{c,j}$ for each column $j = 1, \ldots, n$:

$$S_{ij} = r_{0,i} B_{0,ij} + r_{1,i} B_{1,ij}, \quad s_{c,j} = \frac{\sum_{i=1}^{m} W_{ij} S_{ij}}{\sum_{i=1}^{m} S_{ij}^2}$$

5: **Update binary matrices** (B_0, B_1) via exhaustive search:

$$(B_{0,ij}, B_{1,ij}) = \arg \min_{(s_0, s_1) \in \{\pm 1\}^2} |W_{ij} - s_{c,j}(s_0 r_{0,i} + s_1 r_{1,i})|$$

6: **end for**

7: **Reconstruct quantized weight:**

$$\hat{\mathbf{W}} = s_c \odot (r_0 \odot B_0 + r_1 \odot B_1)$$

8: **Return:** $\hat{\mathbf{W}}, s_c$

Algorithm 3 Deviation-Aware Correction (DAC) for LayerNorm

Input: LayerNorm module LN, calibration activations $\mathbf{X}_{\text{fp}}$ (full-precision), $\mathbf{X}_q$ (quantized)
Output: Corrected LayerNorm module LN

1: Compute mean activation deviation across tokens from calibration data:

$$b = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{X}_{\text{fp},t} - \mathbf{X}_{q,t})$$

2: Introduce a deviation correction bias to the LayerNorm, which is added to the original LayerNorm output during inference.:

$$\beta_{\text{dac}} = b$$

3: **Return** updated LayerNorm module LN

Algorithm 4 WriteBack

Input: Updated parameters Θ of a Transformer block

Output: Updated Transformer block $\widehat{\mathcal{M}}_l$

1: **for** each parameter $\theta \in \Theta$ **do**

2: Replace the corresponding parameter in $\mathcal{M}_l$ with θ

3: **end for**

4: **Return** updated block $\widehat{\mathcal{M}}_l$

Algorithm 5 GetPostAttnLNAct

Input: Transformer block $\mathcal{M}_l$, calibration data $\mathcal{X}$
Output: Post-attention LayerNorm activations X_l

Initialize empty activation set $X_l \leftarrow \emptyset$
for each batch $\mathbf{X} \in \mathcal{X}$ **do**
  Apply pre-attention LayerNorm:

$$\tilde{\mathbf{X}} = \mathrm{LN}_{\text{attn}}(\mathbf{X})$$

  Compute queries, keys, and values:

$$\mathbf{Q} = \tilde{\mathbf{X}} W_Q, \quad \mathbf{K} = \tilde{\mathbf{X}} W_K, \quad \mathbf{V} = \tilde{\mathbf{X}} W_V$$

  Compute attention scores and weighted sum:

$$\mathbf{S} = \mathrm{Softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\Big), \quad \mathbf{A} = \mathbf{S}\mathbf{V}$$

  Apply output projection:

$$\mathbf{A}_o = \mathbf{A} W_O$$

  Apply residual connection:

$$\mathbf{H}^{\text{attn}} = \mathbf{X} + \mathbf{A}_o$$

  Apply post-attention LayerNorm:

$$\mathbf{Y} = \mathrm{PostAttnLN}(\mathbf{H}^{\text{attn}})$$

  Append $\mathbf{Y}$ to X_l
end for
Return X_l

Algorithm 6 MergeScale

Input: Up-projection weight W_{up}, column scale s_c
Output: Scale-merged up-projection weight $\tilde{W}_{up}$

Reshape s_c to match the input dimension of W_{up}
Absorb the scale into W_{up}:

$$\tilde{W}_{up} = W_{up} \cdot \mathrm{diag}(s_c)$$

Return $\tilde{W}_{up}$

C Detailed Proof of Error Reduction via Deviation Correction

Let $\Delta Y_i = Y_{fp,i} - Y_{q,i}$ denote the activation deviation in the i-th feature dimension, where $Y_{fp,i}$ is the full-precision activation and $Y_{q,i}$ is the quantized activation. Let $\mu_i = \mathbb{E}_{\text{calib}}[\Delta Y_i]$ denote the mean of the deviation. We assume that this estimate is representative of the expectation during the actual inference phase, which is reasonable:

$$\mathbb{E}_{\text{inference}}[\Delta Y_i] \approx \mu_i. \tag{20}$$

Let $\sigma_i^2 = \text{Var}(\Delta Y_i)$ denote the variance of the deviation. Then the expected squared error (MSE) in the i-th dimension before deviation correction is:

$$\mathbb{E}[\Delta Y_i^2] = \text{Var}(\Delta Y_i) + (\mathbb{E}[\Delta Y_i])^2 = \sigma_i^2 + \mu_i^2. \tag{21}$$

Applying deviation correction, we adjust the quantized output by adding the estimated mean deviation μ_i: $Y_{\text{aligned},i} = Y_{q,i} + \mu_i$, where $Y_{q,i}$ is the original quantized output in the i-th feature dimension.

The corrected deviation becomes:

$$\Delta Y_i' = Y_{\text{fp},i} - Y_{\text{aligned},i} = \Delta Y_i - \mu_i, \tag{22}$$

Therefore, the mean of the corrected deviation becomes zero:

$$\mathbb{E}[\Delta Y_i'] = \mathbb{E}[\Delta Y_i - \mu_i] = \mu_i - \mu_i = 0, \tag{23}$$

and preserves the same variance:

$$\text{Var}(\Delta Y_i') = \text{Var}(\Delta Y_i) = \sigma_i^2. \tag{24}$$

Therefore, the expected squared error after deviation correction is reduced to:

$$\mathbb{E}[(\Delta Y_i')^2] = \sigma_i^2. \tag{25}$$

The relative reduction in MSE is:

$$\frac{\mathbb{E}[\Delta Y_i^2] - \mathbb{E}[(\Delta Y_i')^2]}{\mathbb{E}[\Delta Y_i^2]} = \frac{\mu_i^2 + \sigma_i^2 - \sigma_i^2}{\mu_i^2 + \sigma_i^2} = \frac{\mu_i^2}{\mu_i^2 + \sigma_i^2} = \frac{\frac{\mu_i^2}{\sigma_i^2}}{1 + \frac{\mu_i^2}{\sigma_i^2}} = \frac{\text{SNR}_i}{1 + \text{SNR}_i}, \tag{26}$$

where $\text{SNR}_i = \mu_i^2 / \sigma_i^2$. This shows that dimensions with higher SNR benefit more from deviation correction.

D Dialog Examples

We evaluate D^2Quant on LLaMA-3-8B at an ultra-low 2-bit precision using zero-shot dialogue prompts. As illustrated in the Dialog Examples, baseline methods like GPTAQ and GPTQ† suffer from severe semantic collapse, often producing unintelligible or factually incorrect outputs. In contrast, D^2Quant consistently generates accurate, contextually relevant, and coherent responses. This superiority underscores its robustness in mitigating knowledge degradation and sustaining structural integrity even under extreme bit-width constraints.

E Discussions and Future Works

Broader Applicability of DSQ. This work applies the Dual-Scale Quantizer (DSQ) to down-projection by leveraging column scale absorption into the preceding up-projection. The same idea may extend to QKV and up/gate projections, where column scales can be folded into the preceding LayerNorm. However, this requires all branches (e.g., Q, K, V) to share a column scale, posing design constraints. Extending DSQ under these constraints is a promising direction.

Identifying and Correcting More Activation Shift Patterns. This work highlights a consistent mean shift at the post-attention LayerNorm caused by quantizing the attention module. However, weight-only quantization may induce other types of activation distribution shifts in different parts of the model. Identifying such patterns and designing general correction mechanisms beyond mean bias could improve the robustness and generalization of quantized models, making this a valuable direction for future work.

Prompt : What does CPU stand for?
GPTAQ : What does CPU stand for? And what does it do for? Well, it has a great deal to do for. **GPTQ† :** He is the right, in the head is the in the future to carry the organization in the present. **D^{2}Quant :** It stands for Central Processing Unit.
Prompt : The chemical symbol for gold is _____
GPTAQ : The chemical symbol for gold. **GPTQ† :** The chemical symbol for gold is and in thea of a a of the course. **D^{2}Quant :** The chemical symbol for gold is Au (Latin: Aurum).
Prompt : The Renaissance began in _____
GPTAQ : In which country did the Renaissance begin? The answer is not known, but it is clear that the Renaissance began in the 14th century. **GPTQ† :** In which country did the Renaissance begin, at the present is in the Sap of a a. **D^{2}Quant :** The Renaissance began in Italy in the late 14th century. It was a period of artistic revival of classical themes.
Prompt : Romeo and Juliet was written by _____
GPTAQ : Romeo and Juliet was written by the same person, but Juliet is the main character of the story. **GPTQ† :** Romeo and Juliet was written by and to a head to create a to present. **D^{2}Quant :** Romeo and Juliet was written by William Shakespeare. It is a romantic tragedy and one of the greatest plays in English literature.
Prompt : The Statue of Liberty is located in _____
GPTAQ : The Statue of Liberty is located in the Hudson River at the mouth of the river. **GPTQ† :** The Statue of Liberty is located in the following. **D^{2}Quant :** The Statue of Liberty is located in New York.
Prompt : Harry Potter was written by _____
GPTAQ : Harry Potter was written by the same person who wrote The Mysterious Case, both of which were published in the same magazine. **GPTQ† :** Harry Potter was written by the a in the present in the future. **D^{2}Quant :** Harry Potter was written by J.K. Rowling. In this book, Harry Potter is the main character of the story.
Prompt : The process plants use to make food is called _____
GPTAQ : The process plants use to make food is the same as the one used by humans. Plants use a variety of methods to make food. **GPTQ† :** The process plants use to make food, as;da he of the 6: 1 in the 7; to carry in the current. **D^{2}Quant :** The process plants use to make food is called photosynthesis. Photosynthesis takes place in plants.
Prompt : Napoleon Bonaparte was a leader of _____
GPTAQ : Napoleon Bonaparte was a leader of the French Revolution. **GPTQ† :** Napoleon Bonaparte was a leader of the following is of the following o character. **D^{2}Quant :** Napoleon Bonaparte was a leader of the French Revolution and the first emperor of France. He was born on August 15, 1769.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe the down-projection sensitivity and activation-drift observations, the DSQ and DAC components, and the sub-4-bit weight-only PTQ setting. The experiments report results across the evaluated model families and benchmarks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix E discusses the constraints on extending DSQ to other projections and the limitation of DAC to mean-shift correction at post-attention LayerNorm.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 3.2 gives the algebraic up–down scaling equivalence. Section 3.3 and Appendix C derive the bias–variance decomposition and DAC error reduction under the stated assumption that calibration statistics are representative of inference.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 specifies the evaluated models, calibration data, quantization settings, and benchmarks. Algorithm 1 summarizes the pipeline, and the public implementation at `https://github.com/XIANGLONGYAN/D2Quant` provides quantization, data-preparation, and evaluation code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The implementation is publicly available at `https://github.com/XIANGLONGYAN/D2Quant`, with installation, data-preparation, and evaluation instructions. The experiments use existing public benchmarks, including WikiText-2 and C4; no new dataset is introduced.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 reports model families, bit-widths, calibration sample count and sequence length, DSQ iterations, and evaluation benchmarks. The released implementation provides configuration options and example commands.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The reported evaluations are single-run results. Error bars, confidence intervals, and statistical significance tests are not reported; repeated evaluations across all model families and bit-widths were not conducted.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 specifies NVIDIA A800-80GB GPUs, including the single-GPU setting and the three-GPU evaluation setting for 70B models. Figure 5 reports quantization time on LLaMA-3-8B. Aggregate compute for the full research project is not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies quantization of existing pretrained models using public benchmarks. It does not involve human-subject experiments or collection of new personal data, and the authors follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on reducing the computational and storage costs of existing LLMs. It does not include a dedicated discussion of both positive and negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The release consists of quantization software rather than a newly collected dataset or a new pretrained foundation model. Existing models and benchmarks are obtained through their original providers and remain subject to their original access conditions.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper cites the existing models, datasets, and software that it uses. However, the manuscript does not enumerate the specific license names and terms for every existing asset, so this item is answered No.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released software is available at `https://github.com/XIANGLONGYAN/D2Quant`. Its README documents installation, data preparation, configuration, and quantization/evaluation commands, and the repository includes a LICENSE file. No newly collected dataset is introduced.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subject research, so IRB approval or an equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The LLMs are the targets being quantized and evaluated. An auxiliary LLM is not used as a component of the proposed quantization algorithm or its core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.