REAL-Q: E2E LLM QUANTIZATION VIA DYNAMIC GRADIENT DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization (PTQ) is essential for deploying large language models (LLMs) under strict resource constraints. State-of-the-art PTQ methods quantize each layer with a single closed-form second-order solver: to remain analytically tractable, they heavily approximate the global loss (dropping cross-channel coupling, pooling output rows into groups), and they then freeze the resulting Hessian across the entire layer, with no way to refresh it as the loss landscape shifts column by column—a phenomenon we call *information misalignment*. We propose **REAL-Q** (**R**eal-time **E**2E-loss **A**ligned **L**LM **Q**uantization), a novel PTQ paradigm that breaks this compromise: instead of diluting the objective for the sake of analytic tractability, REAL-Q targets an end-to-end-aligned surrogate of the global loss and refines it via fine-grained, dynamic Block-wise Gradient Descent applied after every column block (128 columns). By coupling this fine-grained correction with a sliding window mechanism for smooth cross-layer transitions, REAL-Q effectively mitigates error propagation across the network. Experiments on LLaMA-3.1-8B-Instruct and Qwen3-4B/8B/32B cover twelve weight-only and weight–activation settings, with REAL-Q achieving the lowest KL loss and ranking first on downstream tasks in most settings.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks (Grattafiori et al., 2024). However, deploying models with billions of parameters incurs substantial compute and memory costs. Post-training quantization (PTQ) offers a practical solution by compressing model weights to low-bit representations without requiring costly retraining (Gholami et al., 2021).

Among PTQ methods, GPTQ (Frantar et al., 2023) has emerged as the foundational algorithm for weight-only compression. Building upon the Optimal Brain Surgeon (OBS) framework (Hassibi et al., 1993), its column-wise solver has been widely integrated into mainstream inference engines (e.g., vLLM, TensorRT-LLM) and extended by a vast lineage of state-of-the-art techniques (Dettmers et al., 2024; Li et al., 2025; Kim et al., 2025a). Despite its success, GPTQ has a fundamental limitation:

Inaccuracy of the Layer-Local Objective. GPTQ minimizes a layer-wise reconstruction MSE $\|\mathbf{W}\mathbf{x} - \hat{\mathbf{W}}\mathbf{x}\|_2^2$, where $\mathbf{x}$ is the full-precision activation. This local objective suffers from a dual misalignment. Upstream, it assumes full-precision input activations $\mathbf{x}$ and ignores the perturbed inputs $\hat{\mathbf{x}}$ arriving from previously quantized layers, leaving compounding errors uncorrected. Downstream, it weights all output-channel perturbations uniformly, ignoring that the true KL divergence depends on a strongly non-linear, highly coupled mapping through subsequent transformer blocks and the LM head. A surrogate that relies on unperturbed inputs and uniformly weights squared errors is therefore misaligned with the true quantization loss.

Recent works attempt to address these blind spots: GPTAQ (Li et al., 2025) recalibrates inputs to correct upstream errors, yet its per-layer loss remains a uniform reconstruction MSE. Conversely, GuidedQuant (Kim et al., 2025a) incorporates end-to-end gradients to capture downstream sensitivity. However, it still leaves upstream errors uncorrected, and it relies on structural approximations such as ignoring fine-grained cross-channel coupling and sharing the Hessian across output groups. BoA (Kim et al., 2025b) and TurboBoA (Kim et al., 2026) broaden dependency modeling within self-attention to capture cross-projection interactions and output-channel coupling, but remain confined to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: KL divergence to bf16 on WikiText-2 across four representative settings ($\times 10^{-2}$; log scale; lower is better). Percentages show REAL-Q's reduction relative to the best competing method.

the attention sublayer rather than modeling the full set of interactions within a transformer block. In addition, because the true end-to-end Hessian is prohibitively expensive to recompute, the already-approximated matrix remains static within each layer. While the true loss landscape shifts as weights are progressively quantized column by column, the misaligned Hessian causes compensation errors to accumulate as the column-wise sweep progresses. We refer to this phenomenon as *information misalignment*. Here, "information" refers to first- or second-order information such as the gradient and Hessian.

The shared limitation of these methods is structural: requiring the column-wise update to remain analytically solvable forces them to (i) restrict the scope or structure of the dependencies represented by the surrogate—for example, by pooling output rows or confining reconstruction to the attention sublayer—and (ii) freeze the resulting Hessian over the entire layer, with no way to refresh it at finer granularity as the loss landscape shifts with each column quantization step. REAL-Q (**R**eal-time **E**2E-loss **A**ligned **LLM** **Q**uantization) addresses both limitations by interleaving *fine-grained, dynamic* corrections into the column-wise sweep, via two components:

An end-to-end-aligned objective: Aggregated Fisher MSE. REAL-Q's surrogate is a transformer-block-output Fisher MSE, derived from a second-order Taylor expansion of the global KL divergence at the current block's output. Using the *full* aggregated Fisher matrix preserves the cross-channel coupling that grouped baselines must discard, while the choice of expansion point keeps backpropagation confined to at most two adjacent blocks. A loss sliding window smooths the objective across block boundaries to prevent discontinuous jumps in the optimization landscape.

Fine-grained, dynamic Block-wise Gradient Descent. Rather than committing the entire layer to one frozen second-order solver, REAL-Q interleaves a single Adam (Kingma & Ba, 2014) gradient step into the GPTQ sweep *after every column block* (128 columns). This raises the correction granularity from the per-layer level used by prior solvers to the column-block level—roughly two orders of magnitude finer—and lets each step react to the most recent quantized state, since the gradient is recomputed online against the actual partially quantized weights. Adam's diagonal preconditioner additionally adapts the per-coordinate scale on the fly, providing implicit second-order information without ever inverting a Hessian.

In summary, our main contributions are as follows:

- **Aggregated Fisher MSE with Loss Sliding Window:** Within the GPTQ-lineage sequential pipeline, we adopt an end-to-end-aligned objective in place of the layer-local MSE. We formulate an aggregated Fisher MSE that preserves cross-channel coupling while keeping the backpropagation overhead confined to localized block boundaries. To stabilize the sequential quantization process, we introduce a **loss sliding window** mechanism that smooths the

transition of the objective across consecutive transformer blocks and prevents discontinuous jumps in the loss landscape.

- **Dynamic Block-wise Gradient Descent (Block-GD):** We address the *information misalignment* of prior works by replacing static analytical solvers with a dynamic first-order approach. By aligning Adam-driven gradient feedback directly with column-block updates, this mechanism adapts to the shifting loss landscape and corrects accumulated upstream errors online.
- **Empirical Performance:** We evaluate LLaMA-3.1-8B-Instruct and Qwen3-4B/8B/32B at W3A16, W2A16, and W4A4KV4, reporting fidelity, downstream and reasoning accuracy, learning rates, and end-to-end quantization costs in Table 6. Figure 1 highlights KL divergence in four representative settings.

2 Related Work

Second-order post-training quantization. The Optimal Brain Damage (OBD) (LeCun et al., 1989) and Optimal Brain Surgeon (OBS) (Hassibi et al., 1993) frameworks established the foundation for using second-order Hessian information in network compression. GPTQ (Frantar et al., 2023) successfully scaled OBS to billion-parameter LLMs. By processing weight rows concurrently and employing a shared, static inverse Hessian to compensate for quantization errors column by column, GPTQ established the dominant pipeline for modern sequential PTQ. BoA (Kim et al., 2025b) models attention-level dependencies, while TurboBoA (Kim et al., 2026) accelerates joint output-channel compensation and corrects upstream errors.

Global-aware extensions of GPTQ. Recent advancements attempt to enhance the standard sequential GPTQ pipeline by incorporating broader model context into the quantization process. **GPTAQ** (Li et al., 2025) addresses upstream error accumulation by iteratively recalibrating the input activations. Instead of relying solely on fixed, full-precision inputs, it dynamically computes calibration activations using the outputs from previously quantized layers, thereby incorporating upstream quantization drift into the layer-wise reconstruction objective. **GuidedQuant** (Kim et al., 2025a) incorporates end-to-end downstream sensitivity into the local solver. It performs an initial backward pass over the calibration set to obtain per-token saliency scores, which are subsequently used to weight the input covariance matrix. To maintain the computational tractability of the analytical second-order update, GuidedQuant partitions the weight matrix rows into groups and computes a shared, static approximated Hessian for each group to drive the column-wise quantization. ResComp (Li et al., 2026) accounts for compensation-induced residual errors, and YAQA (Tseng et al., 2026) approximates full-model KL curvature with Kronecker-factored Hessian sketches.

Other related directions. EfficientQAT (Chen et al., 2025) combines block-wise weight/quantizer training with end-to-end step-size tuning. Extended discussions on rotation pre-processing (e.g., QuaRot (Ashkboos et al., 2024b), which we adopt), weight-only representations, quantization-aware training, and classical Fisher applications are deferred to Appendix B. This appendix also provides a detailed comparison with BRECQ (Li et al., 2021); while BRECQ shares conceptual similarities in block-level loss modeling, its overall quantization pipeline is fundamentally different from ours.

3 Preliminaries

3.1 Problem Formulation

Consider a pre-trained LLM with L transformer blocks, where each block ℓ contains linear modules (e.g., $\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v, \mathbf{W}^o$ for attention and $\mathbf{W}^{\text{up}}, \mathbf{W}^{\text{gate}}, \mathbf{W}^{\text{down}}$ for the MLP). For a linear module with weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, the goal of PTQ is to find a quantized weight $\hat{\mathbf{W}}$ that minimizes the degradation in model output quality.

For calibration token sequences $\mathcal{D} = \{s_1, \ldots, s_N\}$, we measure quantization quality by the KL divergence between the original and quantized model outputs:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{s \sim \mathcal{D}} \left[D_{\text{KL}} \left(p_{\mathbf{W}}(\cdot \mid s) \parallel p_{\hat{\mathbf{W}}}(\cdot \mid s) \right) \right]. \tag{1}$$

Following recent studies (Kong et al., 2026), we use KL divergence as our primary metric because it most precisely characterizes the output-distribution shift induced by quantization. We additionally report perplexity, zero-shot QA, and reasoning accuracy.

3.2 GPTQ RECAP

GPTQ (Frantar et al., 2023) processes linear modules column by column. Write $\mathbf{H}^{-1} = \mathbf{R}^\top \mathbf{R}$ with $\mathbf{R}$ upper triangular. At step j, GPTQ quantizes column j to $\hat{\mathbf{w}}_j$, computes the residual $\boldsymbol{\delta}_j = \mathbf{w}_j - \hat{\mathbf{w}}_j$, and compensates the remaining columns:

$$\mathbf{W}_{:,j+1:n} \leftarrow \mathbf{W}_{:,j+1:n} - \frac{\boldsymbol{\delta}_j}{R_{jj}} \mathbf{R}_{j,j+1:n}, \tag{2}$$

where $\mathbf{x}_t$ is the module's input activation and $\mathbf{H} = \frac{2}{T}\sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^\top$ is the reconstruction-MSE Hessian (damped in practice). For a column block with index set I_b, GPTQ applies Eq. 2 within the block, then updates the trailing columns as $\mathbf{W}_{:,b_{\text{end}}:n} \leftarrow \mathbf{W}_{:,b_{\text{end}}:n} - \mathbf{E}_b \mathbf{R}_{I_b, b_{\text{end}}:n}$, where $\mathbf{E}_b$ stacks $\boldsymbol{\delta}_j / R_{jj}$ for $j \in I_b$ and b_{end} is the first trailing column.

3.3 GUIDEDQUANT: SALIENCY-WEIGHTED HESSIAN

GuidedQuant (Kim et al., 2025a) weights local reconstruction by end-to-end sensitivity. An initial backward pass computes per-token saliency scores $s_t^{(\ell)} = \|\partial \mathcal{L}_{\text{NLL}} / \partial \mathbf{y}_t^{(\ell)}\|^2$, where $\mathcal{L}_{\text{NLL}}$ is the end-to-end negative log-likelihood. Rows share a Hessian approximation within each of N_g groups (we use $N_g = 4$ in all experiments):

$$\mathbf{H}_g^{(\ell)} = \frac{1}{T}\sum_{t=1}^{T} s_{t,g}^{(\ell)} \cdot \mathbf{x}_t \mathbf{x}_t^\top, \quad g = 1, \ldots, N_g, \tag{3}$$

where $s_{t,g}^{(\ell)}$ is the saliency score restricted to group g. We use the normalized second-moment convention, one half of the weighted-MSE Hessian; this common scaling leaves GPTQ compensation unchanged with relative damping. Output-channel decoupling omits cross-channel interactions, while group sharing further ties the row-wise sensitivity weights.

4 REAL-Q

4.1 OVERVIEW

REAL-Q combines saliency-weighted column-wise compensation with three components: (1) an **Aggregated Fisher-Weighted MSE** for end-to-end gradient guidance; (2) **Dynamic Block-GD** after each column block to correct residual errors; and (3) a **Loss Sliding Window** interpolating objectives across consecutive transformer blocks. Figure 2 compares the pipelines; Appendix A provides pseudocode.

4.2 OBJECTIVE

Optimizing the end-to-end KL divergence (Eq. 1) at every column-block step requires repeated computation through all subsequent transformer blocks and the LM head. We instead use an *Aggregated Fisher MSE* surrogate.

Consider $\mathcal{L}_{\text{KL}}$ as a function of the per-token transformer block output $\mathbf{y}_t$. At the full-precision output $\mathbf{y}_t^*$, both the KL and its gradient vanish. A second-order Taylor expansion gives:

$$\mathbb{E}_t\big[\mathcal{L}_{\text{KL}}(\mathbf{y}_t)\big] \approx \frac{1}{2}\mathbb{E}_t\big[\Delta \mathbf{y}_t^\top \mathbf{H}_t \Delta \mathbf{y}_t\big], \tag{4}$$

where $\Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_t^*$ and $\mathbf{H}_t = \nabla^2_{\mathbf{y}_t} \mathcal{L}_{\text{KL}}\big|_{\mathbf{y}_t^*}$. Writing the expansion as a per-token average is a *token-block-diagonal* approximation: it discards the cross-token second-order terms that downstream attention induces, so the covariance statements below hold under this approximation. With $\mathbf{g}_t = \nabla_{\mathbf{y}_t} \mathcal{L}_{\text{NLL}}$ computed using labels sampled from the full-precision model's output distribution p, the Fisher–Hessian identity gives $\mathbb{E}_t[\mathbf{H}_t] = \mathbb{E}_t[\mathbf{g}_t \mathbf{g}_t^\top]$ in expectation under that distribution.

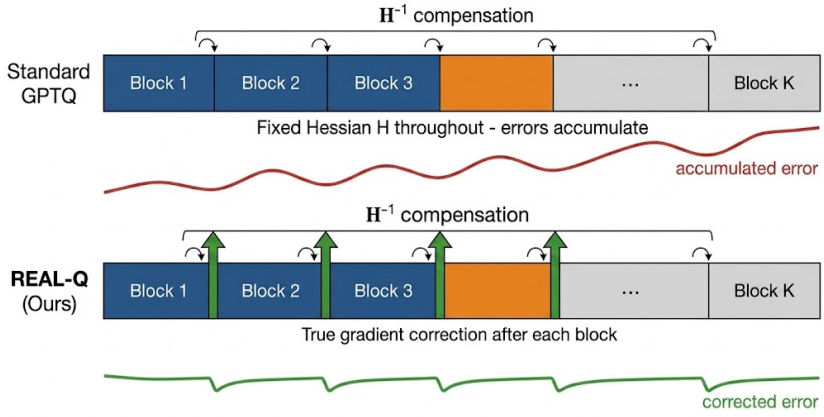


Figure 2: Comparison of static analytical quantization and REAL-Q. In prior static methods (e.g., standard GPTQ or GuidedQuant), a fixed Hessian approximation is used throughout, allowing quantization errors to accumulate unchecked across column blocks (red). In REAL-Q, a dynamic Block-GD step is applied immediately after each block, actively correcting accumulated structural errors and keeping the end-to-end reconstruction loss tightly bounded (green).

Aggregated Fisher loss (ours). We aggregate the Fisher matrix over calibration tokens:

$$\mathbf{F} = \mathbb{E}_t\left[\mathbf{g}_t\mathbf{g}_t^\top\right] \approx \frac{1}{T}\sum_{t=1}^{T}\mathbf{g}_t\mathbf{g}_t^\top \in \mathbb{R}^{d\times d}, \tag{5}$$

where d is the transformer block output dimension. The Block-GD surrogate loss is:

$$\mathcal{L}_{\text{Fisher}} = \frac{1}{2}\mathbb{E}_t\left[\Delta\mathbf{y}_t^\top\mathbf{F}\Delta\mathbf{y}_t\right] = \frac{1}{2}\cdot\frac{1}{T}\sum_{t=1}^{T}\Delta\mathbf{y}_t^\top\mathbf{F}\Delta\mathbf{y}_t. \tag{6}$$

One full matrix $\mathbf{F}$ per transformer block retains cross-channel covariances without per-token Fisher storage or channel grouping; here "full" means all channel pairs within a token, not the full sequence Fisher. Appendix D.1 describes its pre-computation and the final-block objective.

Proposition 4.1 (Decoupled Fisher Aggregation)**.** *The expected second-order penalty is* $\frac{1}{2}\mathbb{E}_t[\Delta\mathbf{y}_t^\top\mathbf{H}_t\Delta\mathbf{y}_t]$. *Mean-field decoupling gives* $\frac{1}{2}\mathbb{E}_t[\Delta\mathbf{y}_t^\top\,\mathbb{E}_t[\mathbf{H}_t]\,\Delta\mathbf{y}_t]$. *The gap due to this decoupling is exactly* $\frac{1}{2}\sum_{i,j}Cov\big((\mathbf{H}_t)_{ij},(\Delta\mathbf{y}_t\Delta\mathbf{y}_t^\top)_{ij}\big)$. *Replacing the mean Hessian by its Fisher estimate yields Eq. 6; finite-sample estimation introduces a separate error (Appendix C.1).*

4.3 Dynamic Block-wise Gradient Descent

After analytically processing column block b, we evaluate $\mathcal{L}^{(b)}$ on a calibration mini-batch: the sliding-window Fisher loss (Eq. 8) for $\ell < L$, or the true KL for the final transformer block. We backpropagate this loss to the current linear module's remaining unquantized weights and apply one Adam (Kingma & Ba, 2014) step, with per-parameter optimizer states:

$$\mathbf{W}_{:,b_{\text{end}}:n} \leftarrow \mathbf{W}_{:,b_{\text{end}}:n} - \eta\cdot\text{Adam}\big(\nabla_{\mathbf{W}_{:,b_{\text{end}}:n}}\mathcal{L}^{(b)}(\mathbf{W}^{(b)})\big), \tag{7}$$

where η is the learning rate. Quantized weights remain fixed, and the analytical Hessian is not recomputed during the module's quantization sweep.

4.4 Loss Sliding Window

The *loss sliding window* (Figure 4) interpolates Fisher objectives between adjacent transformer blocks. Let b_{global} index the B_{total} column-block steps across all linear modules in transformer block ℓ, while b remains the module-local index. The blended objective is:

$$\mathcal{L}_{\text{slide}}^{(b_{\text{global}})} = \alpha^{(b_{\text{global}})}\cdot\mathcal{L}_{\text{Fisher}}^{(\ell)} + \big(1-\alpha^{(b_{\text{global}})}\big)\cdot\mathcal{L}_{\text{Fisher}}^{(\ell+1)}, \tag{8}$$

where $\alpha^{(b_{\text{global}})} = 1 - \frac{b_{\text{global}}-1}{B_{\text{total}}-1}$ for $B_{\text{total}} > 1$. Its gradient propagates from block $\ell + 1$ into block ℓ, restricting backpropagation to two transformer blocks while incorporating the effect on the next block.

5 Theoretical Analysis

We present two theoretical results justifying REAL-Q: formalizing the inherent error accumulation of static Hessian solvers, and deriving a sufficient cosine condition under which Block-GD descends on the true global loss. Full proofs are deferred to Appendix C.2 and C.3.

Error Accumulation in Static Solvers. Globally-guided PTQ methods rely on a static approximation of the end-to-end Hessian ($\mathbf{H}$). However, as column quantization progresses, the true downstream-aware Hessian shifts. Letting $\Delta\mathbf{H}^{(j)} = \mathbf{H}^{(j)}_{\text{ideal}} - \mathbf{H}$ denote this gap, the static solver incurs a cumulative penalty. Assuming a linear structural drift, we prove in Appendix C.2 that this expected uncompensated error scales as:

$$\mathbb{E}[\mathcal{E}] = \Omega(n \cdot \bar{\epsilon}^2 \cdot \|\mathbf{H}\|_2), \tag{9}$$

where n is the column dimension and $\bar{\epsilon}^2$ is the expected squared residual. Under the stated assumption, this linear error growth with respect to module width n indicates that static analytical solvers cannot prevent compounding cross-column errors.

Descent Condition Analysis under First-Order Updates. To study the effect of gradient alignment, consider the selected objective $\mathcal{L}^{(b)}$ in Eq. 7. Assuming the true end-to-end objective $J(\mathbf{W})$ is β-smooth, the true loss after a gradient step $\mathbf{W}^{(+)} = \mathbf{W} - \eta\mathbf{g}_{\text{surr}}$ satisfies the descent-lemma upper bound

$$J(\mathbf{W}^{(+)}) \leq J(\mathbf{W}) - \eta\|\nabla J(\mathbf{W})\|\|\mathbf{g}_{\text{surr}}\|\cos\theta + \frac{\eta^2\beta}{2}\|\mathbf{g}_{\text{surr}}\|^2, \tag{10}$$

where $\cos\theta$ is the cosine similarity between the nonzero surrogate and true gradients. For $\eta > 0$, a sufficient condition for strict descent ($J(\mathbf{W}^{(+)}) < J(\mathbf{W})$) is:

$$\cos\theta > \frac{\eta\beta}{2} \cdot \frac{\|\mathbf{g}_{\text{surr}}\|}{\|\nabla J(\mathbf{W})\|}. \tag{11}$$

This condition is sufficient but not necessary and applies to unpreconditioned gradient steps, not directly to Adam or STE updates. Figure 3 reports gradient alignment without testing the threshold.

6 Experiments

6.1 Experimental Setup

Setup. We evaluate LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3-4B/8B/32B (Yang et al., 2025) at W3A16, W2A16, and W4A4KV4. We compare only scalar-quantization methods because scalar quantization is more broadly supported by inference engines and typically enables faster inference than vector-codebook alternatives; REAL-Q modifies the compensation solver rather than the quantizer itself and is therefore orthogonal to, and in principle compatible with, arbitrary quantizers. The PTQ baselines (GPTAQ, GuidedQuant, ResComp, and TurboBoA) and REAL-Q share the same symmetric scalar weight quantizer, group size 128, activation ordering, frozen WikiText-2 (Merity et al., 2017) calibration tokens (256×2048), and QuaRot rotation. A4/K4/V4 use symmetric per-token quantization with clipping ratio 0.9. Since REAL-Q directly updates weights through gradient-based corrections, we also compare it with EfficientQAT (Chen et al., 2025), a lightweight QAT method. EfficientQAT currently does not support QuaRot, so we report its unrotated results. We additionally compare against YAQA (Tseng et al., 2026), controlled to use the same scalar weight quantizer. Because neither EfficientQAT nor YAQA performs any dedicated optimization for activation quantization, both collapse under W4A4KV4.

Metrics. We report held-out WikiText-2 KL/PPL, ten-task zero-shot QA accuracy, and GSM8K/MATH-500/HumanEval+ accuracy. REAL-Q uses 32 backward samples per step, column blocks of 128, and a sliding window. Table 6 lists the selected LR: the reverse-cosine endpoint for weight-only quantization and a constant for W4A4KV4. The final-block LR is 10^{-5} for Qwen3-4B and 10^{-6} otherwise. Additional evaluation and optimizer settings are in Appendix D.1.

6.2 Main Results

Tables 1–3 summarize KL divergence, perplexity, and mean reasoning accuracy; QA means, per-task reasoning results, GPU-hours, and learning rates are provided in Appendix E. REAL-Q achieves the lowest KL in all twelve settings and the highest observed mean reasoning accuracy in most settings, with particularly strong reasoning gains under W4A4KV4, demonstrating its ability to compensate for activation quantization errors. EfficientQAT attains lower PPL than REAL-Q in several settings, because QAT directly optimizes language-modeling loss on the calibration data rather than solely aligning the quantized model with full-precision outputs.

Table 1: WikiText-2 KL divergence ($\times 10^{-2}$; lower is better). bf16 defines the reference distribution, so its KL is omitted. Bold marks the best quantized result per setting.

Precision	bf16	GPTAQ	GuidedQuant	ResComp	TurboBoA	EfficientQAT	YAQA	REAL-Q
LLaMA-3.1-8B-Instruct								
W3A16	—	11.86	11.68	12.40	10.95	21.65	193.54	**9.71**
W2A16	—	71.93	85.18	75.78	93.61	366.74	1077.40	**54.51**
W4A4KV4	—	21.08	25.72	21.64	21.39	151.13	365.06	**19.75**
Qwen3-4B								
W3A16	—	19.18	17.28	19.81	28.32	33.35	16.97	**14.98**
W2A16	—	101.51	107.10	107.11	184.26	451.55	161.95	**65.99**
W4A4KV4	—	45.56	43.84	41.67	59.51	594.45	809.10	**33.01**
Qwen3-8B								
W3A16	—	13.20	12.49	14.46	16.29	25.44	14.03	**11.57**
W2A16	—	70.34	73.89	68.28	507.65	401.43	1941.40	**56.45**
W4A4KV4	—	30.57	30.93	29.62	36.67	616.60	797.29	**25.42**
Qwen3-32B								
W3A16	—	15.58	14.96	15.25	428.87	25.86	14.15	**12.94**
W2A16	—	71.57	66.48	64.12	98.73	273.53	1590.48	**48.25**
W4A4KV4	—	41.08	44.82	40.13	40.40	871.47	1249.63	**25.60**

Table 2: WikiText-2 perplexity (lower is better). Bold marks the best quantized result per setting; bf16 is shown as a reference.

Precision	bf16	GPTAQ	GuidedQuant	ResComp	TurboBoA	EfficientQAT	YAQA	REAL-Q
LLaMA-3.1-8B-Instruct								
W3A16	7.21	8.12	8.11	8.17	8.05	**7.92**	50.43	7.94
W2A16	7.21	14.56	16.76	15.19	18.15	257.20	3.48×10^{5}	**12.32**
W4A4KV4	7.21	8.85	9.38	8.93	8.87	29.12	275.24	**8.72**
Qwen3-4B								
W3A16	13.66	15.61	15.05	15.53	16.56	**13.18**	15.22	14.08
W2A16	13.66	28.80	26.06	30.51	65.97	666.11	41.73	**19.10**
W4A4KV4	13.66	18.93	17.71	18.05	20.96	2059.36	1.87×10^{4}	**15.16**
Qwen3-8B								
W3A16	9.72	10.74	10.40	10.80	10.91	**10.13**	10.53	10.35
W2A16	9.72	16.70	15.61	16.11	1013.21	353.35	9.23×10^{8}	**14.99**
W4A4KV4	9.72	12.18	11.74	12.00	12.63	2573.78	1.57×10^{4}	**11.35**
Qwen3-32B								
W3A16	7.61	8.03	8.03	8.00	385.14	**7.65**	8.02	7.76
W2A16	7.61	12.16	11.43	11.39	15.83	82.73	3.49×10^{7}	**9.83**
W4A4KV4	7.61	9.44	9.81	9.41	9.35	2.91×10^{4}	1.41×10^{6}	**8.28**

6.3 Ablation Study

We evaluate the complete configuration and eight one-factor variants on Qwen3-4B at W3A16 and W4A4KV4 (Table 4). We increase the column block size from 128 to 512, reduce backward samples

Table 3: Mean reasoning accuracy (%) on GSM8K, MATH-500, and HumanEval+ (higher is better). Bold marks the best quantized result per setting; bf16 is shown as a reference.

Precision	bf16	GPTAQ	GuidedQuant	ResComp	TurboBoA	EfficientQAT	YAQA	REAL-Q
LLaMA-3.1-8B-Instruct								
W3A16	59.77	41.05	45.49	42.73	46.93	40.13	2.29	**47.62**
W2A16	59.77	1.04	1.02	0.97	0.00	0.00	0.00	**2.16**
W4A4KV4	59.77	43.14	40.47	42.51	42.40	1.60	0.31	**43.20**
Qwen3-4B								
W3A16	68.44	54.89	57.40	50.05	49.05	28.04	55.11	**57.81**
W2A16	68.44	0.26	0.89	0.34	0.00	0.09	**1.91**	1.15
W4A4KV4	68.44	33.48	38.39	38.72	20.46	0.12	1.45	**51.98**
Qwen3-8B								
W3A16	64.26	50.45	53.13	50.84	57.25	45.94	56.22	**59.15**
W2A16	64.26	1.92	1.58	1.12	1.70	1.71	0.00	**2.46**
W4A4KV4	64.26	52.17	45.70	50.25	47.58	0.49	2.33	**59.25**
Qwen3-32B								
W3A16	67.56	65.46	62.55	61.15	1.58	57.35	62.60	**66.82**
W2A16	67.56	2.54	5.76	2.43	6.26	0.28	0.00	**15.42**
W4A4KV4	67.56	52.63	53.39	53.77	57.06	0.18	0.71	**64.80**

from 32 to 8, disable the loss sliding window, or set the main LR to zero. Calibration tokens and random seeds are fixed, and the independent final-block LR is unchanged.

Table 4: Complete configuration and four single-factor ablations per precision. KL is reported in units of 10^{-2}. The baseline uses $B = 128$, 32 backward samples/batch, sliding window on, and the LR shown in the group heading; the remaining rows report the corresponding single-factor variants.

Variant	KL↓	PPL↓	QA↑	Reasoning↑	GPU-h↓
Qwen3-4B — W3A16; LR 4×10^{-6}					
Baseline (all enabled)	**14.98**	**14.08**	60.86	**57.81**	0.371
$B = 512$	15.71	14.54	60.38	52.06	**0.182**
Backward samples/batch $= 8$	15.25	14.15	60.37	56.32	0.278
No sliding window	15.45	14.17	**61.07**	53.18	0.314
Main LR $= 0$	15.60	14.22	59.79	52.32	0.223
Qwen3-4B — W4A4KV4; LR 4×10^{-6}					
Baseline (all enabled)	**33.01**	**15.16**	**57.73**	**51.98**	0.368
$B = 512$	36.51	16.37	56.17	47.08	**0.187**
Backward samples/batch $= 8$	34.42	15.39	56.38	48.09	0.272
No sliding window	35.66	15.46	55.59	47.39	0.324
Main LR $= 0$	40.32	15.90	55.00	37.08	0.231

The complete configuration achieves the best KL and PPL in both settings and the highest QA and mean reasoning accuracy at W4A4KV4. Compared with the complete configuration, setting the main LR to zero reduces mean reasoning accuracy in both settings, supporting the effectiveness of Block-GD. The loss sliding window is particularly effective in the more challenging and less stable activation-quantization setting.

6.4 Loss Approximation Quality

To empirically assess the quality of our aggregated Fisher MSE as a surrogate, we evaluate the geometric alignment between the surrogate gradients and the true end-to-end KL gradient. Figure 3 plots this mean cosine similarity on Qwen3-0.6B (W4A16). REAL-Q maintains high alignment with the true global trajectory. By contrast, the layer-wise MSE (GPTQ/GPTAQ) and saliency-guided MSE (GuidedQuant) hover near zero, consistent with their limited ability to capture the true descent direction. Full experimental setup is detailed in Appendix D.2.

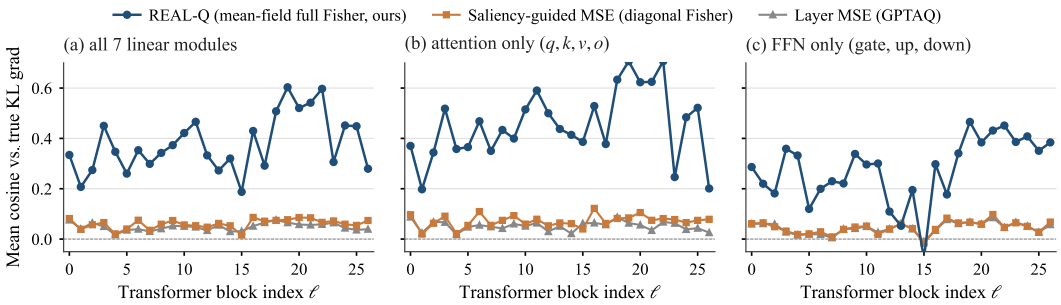


Figure 3: Mean cosine similarity between each surrogate's gradient and the true end-to-end KL gradient on Qwen3-0.6B (W4A16). REAL-Q's full Fisher (blue) remains consistently aligned with the true gradient, whereas the saliency-guided MSE (GuidedQuant, orange) and module MSE (GPTQ / GPTAQ, grey) sit near zero alignment.

Additional details. GPU-hours and Fisher storage costs are reported in Appendices D.3 and D.4.

7 Conclusion and Future Work

REAL-Q addresses the precision–tractability trade-off in modern PTQ by replacing static analytical solvers with a dynamic, coarse-to-fine optimization hierarchy. By pairing a full Aggregated Fisher MSE with Block-GD, our method reduces *information misalignment* and preserves cross-channel coupling. Furthermore, our theoretical analysis shows that, under a linear structural-drift assumption, static solvers incur $\Omega(n)$ compounding errors, and gives a sufficient cosine condition for unpreconditioned gradient steps to decrease the true global loss.

The shift from static analytical solvers to dynamic gradient-based corrections in REAL-Q opens up several promising avenues for future research to further refine the methodology:

(1) Enhancing Surrogate Fidelity for Severe Non-linearities. As shown by the cosine alignment results in Appendix D.2, while our aggregated Fisher MSE provides accurate gradient guidance for attention modules, its correlation with the true end-to-end gradient degrades on certain FFN linear modules (notably `up_proj` and `down_proj`). We attribute this to the SwiGLU non-linearity, which makes a pure second-order Taylor truncation a less faithful approximation of the downstream loss landscape. Future work could explore higher-order Taylor expansions or non-linear proxy functions specifically designed to capture FFN activation dynamics.

(2) Cross-Module Optimizer State Transfer. Currently, the Adam optimizer is re-initialized for each new linear module being quantized. Consequently, its first- and second-moment estimates have a very limited number of update steps to warm up before the module's quantization is finalized. Developing an optimizer with structured state-transfer mechanisms across linear modules—or designing custom optimizers tailored for short, non-stationary sequential trajectories—could significantly accelerate convergence and further improve REAL-Q's correction capabilities.

AI USE STATEMENT

Generative AI tools assisted with language and LaTeX editing, literature and citation checks, and consistency checks and corrections of mathematical notation and derivations. They were not used to run experiments or generate the reported results. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 4 describes the method, and Appendix A provides pseudocode. The theoretical assumptions and proofs are given in Section 5 and Appendix C. Section 6.1 and Appendix D document the calibration data, quantization and optimizer settings, learning-rate, evaluation settings, and computation and storage costs. An anonymized ZIP archive of our code is provided as supplementary material.

REFERENCES

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations (ICLR)*, 2024a.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. Low-rank quantization-aware training for LLMs. *arXiv preprint arXiv:2406.06385*, 2024.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, 2025. doi: 10.18653/v1/2025.acl-long.498. URL https://aclanthology.org/2025.acl-long.498/.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare quantization of neural networks with mixed-precision. In *International Conference on Computer Vision (ICCV)*, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. URL https://zenodo.org/records/10256836.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 1993.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/c6ec1844bec96d6d32ae95ae694e23d8-Abstract-Datasets_and_Benchmarks.html.

Jinuk Kim, Marwa El Halabi, Wonpyo Park, Clemens J. S. Schaefer, Deokjae Lee, Yeonhong Park, Jae W. Lee, and Hyun Oh Song. GuidedQuant: Large language model quantization via exploiting end loss guidance. In *International Conference on Machine Learning (ICML)*, 2025a.

Junhan Kim, Ho-Young Kim, Eulrang Cho, Chungman Lee, Joonyoung Kim, and Yongkweon Jeon. BoA: Attention-aware post-training quantization without backpropagation. In *International Conference on Machine Learning (ICML)*, pp. 30132–30152, 2025b. URL https://proceedings.mlr.press/v267/kim25j.html.

Junhan Kim, Yeo Jeong Park, Seungwoo Son, Chungman Lee, Ho-young Kim, Joonyoung Kim, and Yongkweon Jeon. TurboBoA: Faster and exact attention-aware quantization without backpropagation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.04929.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jason Kong, Nilesh Prasad Pandey, Flavio Ponzina, and Tajana Rosing. A KL lens on quantization: Fast, forward-only sensitivity for mixed-precision SSM-transformer models. *arXiv preprint arXiv:2604.13440*, 2026.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

Shuaiting Li, Juncan Deng, Kedong Xu, Rongtao Deng, Hong Gu, Minghan Jiang, Haibin Shen, and Kejie Huang. Rethinking residual errors in compensation-based LLM quantization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2604.07955.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. GPTAQ: Efficient finetuning-free quantization for asymmetric calibration. In *International Conference on Machine Learning (ICML)*, pp. 36690–36706, 2025. URL https://proceedings.mlr.press/v267/li25dn.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group Fisher pruning for practical network compression. In *International Conference on Machine Learning (ICML)*, 2021.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2024a.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024b.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtarik. PV-Tuning: Beyond straight-through estimation for extreme LLM compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, 2015.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *International Conference on Machine Learning (ICML)*, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning (ICML)*, 2024.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2505.22988.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

A REAL-Q ALGORITHM PSEUDOCODE AND SLIDING-WINDOW ILLUSTRATION

Algorithm 1 REAL-Q Quantization

Require: Pre-trained model with weights $\{\mathbf{W}^{(\ell)}\}$, calibration set $\mathcal{D}$, block size B, learning rate η

Stage 0: Run end-to-end backward on $\mathcal{D}$ to compute aggregated Fisher matrix $\mathbf{F}^{(\ell)}$ and saliency $\mathbf{s}^{(\ell)}$ for all layers ℓ
for layer $\ell = 1, \ldots, L$ **do**
  for each linear module in layer ℓ **do**
    Compute $\mathbf{H}_g$ and upper-triangular $\mathbf{R}_g$ with $\mathbf{H}_g^{-1} = \mathbf{R}_g^\top \mathbf{R}_g$ for each row-group g
    for block $b = 1, \ldots, \lceil n/B \rceil$ **do**
      // Standard GPTQ within block (executed in parallel across groups)
      for column j in block b **do**
        $\hat{\mathbf{w}}_{g,j} \leftarrow \text{quant}(\mathbf{W}_{g,j})$; $\boldsymbol{\delta}_{g,j} \leftarrow \mathbf{W}_{g,j} - \hat{\mathbf{w}}_{g,j}$ $(\forall g)$
        Update within block via Eq. 2 using $\mathbf{R}_g$; store $\boldsymbol{\delta}_{g,j}/(R_g)_{jj}$ in $\mathbf{E}_{g,b}$
      end for
      // Second-order compensation for trailing columns
      $\mathbf{W}_{g,b_{\text{end}}:n} \leftarrow \mathbf{W}_{g,b_{\text{end}}:n} - \mathbf{E}_{g,b}(\mathbf{R}_g)_{I_b,b_{\text{end}}:n}$ $(\forall g)$
      // Block-GD updates only the current module's trailing columns, across all rows
      // b_{global} counts column blocks across modules within layer ℓ (Eq. 8)
      Take the next calibration mini-batch $\mathcal{B}_b \subset \mathcal{D}$ (32 sequences); evaluate the following loss on $\mathcal{B}_b$
      if $\ell = L$ **then**
        $\mathcal{L}^{(b)} \leftarrow$ true KL divergence $\mathcal{L}_{\text{KL}}$ against the LM head
      else
        $\mathcal{L}^{(b)} \leftarrow$ sliding-window loss $\mathcal{L}_{\text{slide}}^{(b_{\text{global}})}$ (Eq. 8) using Fisher MSE
      end if
      $\mathbf{g}^{(b)} \leftarrow \nabla_{\mathbf{W}_{:,b_{\text{end}}:n}} \mathcal{L}^{(b)}$
      $\mathbf{W}_{:,b_{\text{end}}:n} \leftarrow \mathbf{W}_{:,b_{\text{end}}:n} - \eta \cdot \text{Adam}(\mathbf{g}^{(b)})$
    end for
  end for
end for

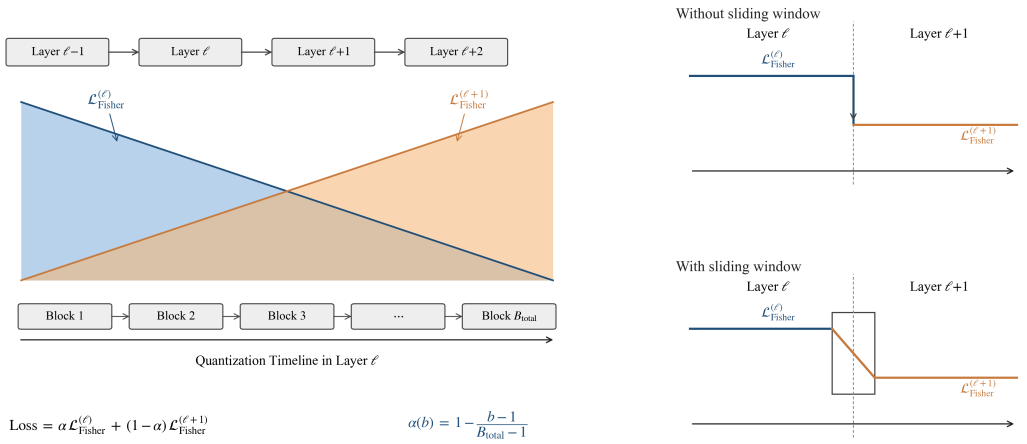


Figure 4: Illustration of the loss sliding window mechanism (Section 4.4). The diagram's b denotes b_{global} in Eq. 8. Within transformer block ℓ, the weight of $\mathcal{L}_{\text{Fisher}}^{(\ell)}$ (blue) decreases linearly while that of $\mathcal{L}_{\text{Fisher}}^{(\ell+1)}$ (orange) increases across all modules' column blocks. **Right:** schematic contrast between an abrupt objective switch and a gradual interpolation.

B Extended Related Work

Weight-only PTQ methods. A complementary line of work targets aggressive low-bit weight-only quantization, where activations stay in higher precision. AWQ (Lin et al., 2024) scales salient weight channels prior to quantization to protect them from rounding error; SqueezeLLM (Kim et al., 2024) uses sensitivity-based non-uniform clustering of weights; SpQR (Dettmers et al., 2024) keeps a small number of outlier weights at higher precision and quantizes the rest more aggressively; OmniQuant (Shao et al., 2024) learns clipping ranges and weight transformations block-wise on calibration data; AQLM (Egiazarian et al., 2024) uses additive vector quantization for sub-3-bit weight compression. These methods are largely orthogonal to REAL-Q: they design new representations or optimize auxiliary quantization parameters (scales/shifts), while REAL-Q targets the sequential weight-updating solver itself. REAL-Q could in principle be combined with any of these representations.

Incoherence and rotation-based methods. A separate family applies orthogonal transforms to the weight and activation matrices before quantization to reduce outlier magnitudes and produce more uniform distributions. QuIP (Chee et al., 2023) introduces the incoherence-processing framework with random orthogonal matrices and adaptive rounding; QuIP# (Tseng et al., 2024) extends this with structured (Hadamard) rotations and lattice codebooks. SliceGPT (Ashkboos et al., 2024a) uses orthogonal transformations to enable post-training structural pruning by safely reducing model dimensionality. Furthermore, methods like SpinQuant (Liu et al., 2024b) and FlatQuant (Sun et al., 2025) actively learn rotation matrices or affine transformations to optimize weight and activation distributions. These pre-processing steps are complementary to the OBS-style update used by REAL-Q—we adopt QuaRot (Ashkboos et al., 2024b) as default rotation pre-processing in all REAL-Q experiments.

Quantization-aware training and fine-tuning. A different class of methods trades calibration efficiency for additional training: LLM-QAT (Liu et al., 2024a) performs full quantization-aware training with distillation; LR-QAT (Bondarenko et al., 2024) reduces the cost of QAT via low-rank adapters; EfficientQAT (Chen et al., 2025) jointly trains block parameters before tuning quantization step sizes end-to-end; PV-tuning (Malinovskii et al., 2024) fine-tunes the discrete codebook used by extreme low-bit quantization methods. These methods use additional training, unlike a single sequential PTQ pass. REAL-Q sits firmly on the PTQ side of this trade-off: it incorporates first-order gradient corrections entirely within the single-pass GPTQ calibration pipeline, never updating dequantized weights globally or requiring full training epochs.

Hessian and Fisher information in compression. Second-order information has been extensively used for network compression. WoodFisher (Singh & Alistarh, 2020) demonstrated the effectiveness of using the empirical Fisher matrix as a tractable proxy for the exact Hessian in OBS-based structural pruning. Similarly, Liu et al. (2021) derives Fisher-based metrics for structured pruning. HAWQ (Dong et al., 2019) and HAWQ-V2 (Dong et al., 2020) use Hessian eigenvalues and traces to guide mixed-precision quantization. K-FAC (Martens & Grosse, 2015) approximates the Fisher matrix as Kronecker products for efficient natural gradient computation (Amari, 1998).

Comparison with BRECQ. BRECQ (Li et al., 2021) uses block-output diagonal-Fisher reconstruction for CNN quantization, optimizing regularized relaxed rounding variables and activation step sizes. For an idealized single linear reconstruction unit with fixed inputs and score weights, the reconstruction loss is:

$$\mathcal{L}_{\text{diag,lin}} = \frac{1}{T}\sum_{t=1}^{T}(\Delta\mathbf{W}\mathbf{x}_t)^\top \operatorname{diag}(\mathbf{g}_t \odot \mathbf{g}_t)\,(\Delta\mathbf{W}\mathbf{x}_t), \tag{12}$$

where $\Delta\mathbf{W} = \mathbf{W} - \hat{\mathbf{W}}$ and $\mathbf{g}_t$ is the task-loss gradient at the full-precision module output. Expanding by output row i gives

$$\mathcal{L}_{\text{diag,lin}} = \frac{1}{T}\sum_{i=1}^{m}\sum_{t=1}^{T} g_{t,i}^2\,(\Delta\mathbf{W}_{i,:}\mathbf{x}_t)^2 = \sum_{i=1}^{m}\Delta\mathbf{W}_{i,:}\underbrace{\left(\tfrac{1}{T}\sum_{t=1}^{T} g_{t,i}^2\,\mathbf{x}_t\mathbf{x}_t^\top\right)}_{=:\,\mathbf{H}_i}\Delta\mathbf{W}_{i,:}^\top. \tag{13}$$

For the same inputs and score weights, this has the row-decoupled form of Eq. 3 with $N_g = m$. This identity is restricted to linear reconstruction; a nonlinear block's Jacobian can couple internal weights even with diagonal output weighting.

C Theoretical Analysis with Proofs

C.1 Proof of Decoupled Fisher Aggregation (Proposition 4.1)

Let $\mathbf{A}_t = \Delta\mathbf{y}_t \Delta\mathbf{y}_t^\top$ and $\bar{\mathbf{H}} = \mathbb{E}_t[\mathbf{H}_t]$. The second-order penalty and its mean-field counterpart are

$$Q = \frac{1}{2}\mathbb{E}_t[\Delta\mathbf{y}_t^\top \mathbf{H}_t \Delta\mathbf{y}_t] = \frac{1}{2}\mathrm{Tr}(\mathbb{E}_t[\mathbf{H}_t\mathbf{A}_t]), \tag{14}$$

$$Q_{\mathrm{dec}} = \frac{1}{2}\mathbb{E}_t[\Delta\mathbf{y}_t^\top \bar{\mathbf{H}} \Delta\mathbf{y}_t] = \frac{1}{2}\mathrm{Tr}(\bar{\mathbf{H}}\,\mathbb{E}_t[\mathbf{A}_t]). \tag{15}$$

Since both matrices are symmetric, their trace product is their Frobenius inner product. Thus

$$Q - Q_{\mathrm{dec}} = \frac{1}{2}\sum_{i,j} \mathrm{Cov}\big((\mathbf{H}_t)_{ij}, (\mathbf{A}_t)_{ij}\big). \tag{16}$$

Replacing $\bar{\mathbf{H}}$ by its Fisher estimate $\mathbf{F}$ gives Eq. 6. The covariance sum describes only the decoupling gap; finite-sample Fisher estimation and higher-order Taylor terms are separate approximations. □

C.2 Error Accumulation in Column-wise Quantization

Consider a single linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ and input $\mathbf{x} \in \mathbb{R}^n$. The standard layer-wise quantization objective is:

$$\min_{\hat{\mathbf{W}}} \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{W}\mathbf{x} - \hat{\mathbf{W}}\mathbf{x}\|^2\right] = \min_{\hat{\mathbf{W}}} \frac{1}{2}\mathrm{Tr}\left((\mathbf{W}-\hat{\mathbf{W}})\mathbf{H}(\mathbf{W}-\hat{\mathbf{W}})^\top\right), \tag{17}$$

where $\mathbf{H} = 2\mathbb{E}[\mathbf{x}\mathbf{x}^\top] \in \mathbb{R}^{n\times n}$ is the row-wise Hessian, using the same normalization as Section 3.2.

Theorem C.1 (Reconstruction error of GPTQ; restated from Frantar et al. (2022))**.** *Let GPTQ quantize $\mathbf{W}$ column-by-column under a fixed positive-definite Hessian $\mathbf{H}$, with $\mathbf{H}^{-1} = \mathbf{R}^\top\mathbf{R}$ and $\mathbf{R}$ upper triangular. For the step-j residual $\boldsymbol{\epsilon}_j = \mathbf{W}^{(j)}_{:,j} - Q(\mathbf{W}^{(j)}_{:,j})$, the final quadratic reconstruction loss is exactly $\mathcal{L}_{GPTQ} = \sum_{j=1}^n \|\boldsymbol{\epsilon}_j\|_2^2/(2R_{jj}^2)$. If damping is applied, this identity refers to the damped quadratic objective.*

Proof of Theorem C.1. Let $S_j = \{j,\dots,n\}$ be the remaining free columns and $\mathbf{K}^{(j)} = (\mathbf{H}_{S_j,S_j})^{-1}$, retaining the original column indices. Schur complementation gives $K^{(j)}_{jj} = R^2_{jj}$ and $K^{(j)}_{jk} = R_{jj}R_{jk}$ for $k > j$. Thus the OBS compensation is $-\boldsymbol{\epsilon}_j \mathbf{R}_{j,j+1:n}/R_{jj}$, and the constrained minimum increases by $\|\boldsymbol{\epsilon}_j\|_2^2/(2R_{jj}^2)$ (Hassibi et al., 1993; Frantar et al., 2022). Summing these increments proves the identity. □

Lemma C.2 (Static Approximation Penalty)**.** *Let $\mathbf{H}^{(j)}_{\mathrm{ideal}}$ denote the true downstream-aware Hessian at step j, and $\Delta\mathbf{H}^{(j)} = \mathbf{H}^{(j)}_{\mathrm{ideal}} - \mathbf{H}$ denote the gap of the cached static Hessian. In GPTQ, the weight update at step j spans multiple columns via compensation, forming a rank-1 perturbation $\Delta\mathbf{W}^{(j)} = \boldsymbol{\epsilon}_j\mathbf{v}_j^\top$, where $\mathbf{v}_j \in \mathbb{R}^n$ contains the static Hessian dependencies. Assuming the structural drift makes this static compensation increasingly misaligned, such that the projection gap is lower-bounded by $\mathbf{v}_j^\top \Delta\mathbf{H}^{(j)}\mathbf{v}_j \geq c\cdot(j/n)\cdot\|\mathbf{H}\|_2$ for some constant $c > 0$, the expected cumulative uncompensated penalty scales as:*

$$\mathbb{E}[\mathcal{E}] = \Omega(n\cdot\bar{\epsilon}^2\cdot\|\mathbf{H}\|_2), \tag{18}$$

where $\bar{\epsilon}^2 = \mathbb{E}[\|\boldsymbol{\epsilon}_j\|_2^2]$ is the expected squared residual norm.

Proof of Lemma C.2. At step j, quantizing column j introduces an immediate residual $\boldsymbol{\epsilon}_j = \mathbf{W}^{(j)}_{:,j} - Q(\mathbf{W}^{(j)}_{:,j})$ (defined as in Theorem C.1, with $\mathbf{W}^{(j)}$ the weight matrix entering step j). Following the OBS framework (Frantar et al., 2022), GPTQ applies a correlated update to the remaining unquantized columns to compensate. The exact step-wise weight perturbation (new minus old) is a rank-1 matrix $\Delta\mathbf{W}^{(j)} = \boldsymbol{\epsilon}_j \mathbf{v}_j^\top$, where the compensation vector $\mathbf{v}_j \in \mathbb{R}^n$ is strictly defined as:

$$[\mathbf{v}_j]_k = \begin{cases} 0 & \text{if } k < j \\ -1 & \text{if } k = j \\ -R_{jk}/R_{jj} & \text{if } k > j \end{cases} \tag{19}$$

Evaluating this exact GPTQ update under the true downstream landscape, the actual step-wise penalty gap introduced by the stale Hessian is the quadratic form:

$$\xi_j = \frac{1}{2}\mathrm{Tr}\left(\Delta\mathbf{W}^{(j)}\Delta\mathbf{H}^{(j)}(\Delta\mathbf{W}^{(j)})^\top\right) = \frac{1}{2}\left(\mathbf{v}_j^\top \Delta\mathbf{H}^{(j)}\mathbf{v}_j\right)\cdot\|\boldsymbol{\epsilon}_j\|_2^2. \tag{20}$$

In deep neural networks with non-quadratic loss landscapes, the true local Hessian $\mathbf{H}^{(j)}_{\text{ideal}}$ deviates from the initial static Hessian $\mathbf{H}$ proportional to the accumulated distance traversed in the weight space. As the column index j progresses, the accumulated weight displacement grows monotonically. It is therefore reasonable to assume that the structural drift penalty along the compensation direction, $\mathbf{v}_j^\top \Delta\mathbf{H}^{(j)}\mathbf{v}_j$, worsens at least linearly with the macroscopic quantization progress j/n.

Formalizing this motivated assumption ($\mathbf{v}_j^\top \Delta\mathbf{H}^{(j)}\mathbf{v}_j \geq c\frac{j}{n}\|\mathbf{H}\|_2$ for some $c > 0$), the step-wise penalty gap becomes $\xi_j \geq \frac{c}{2}\frac{j}{n}\|\mathbf{H}\|_2\|\boldsymbol{\epsilon}_j\|_2^2$.

Recognizing that in sequential quantization, the magnitude of the local quantization residual $\|\boldsymbol{\epsilon}_j\|_2^2$ is primarily determined by local weight variance, it can be reasonably modeled as statistically independent of the macroscopic structural drift index j. Taking the expectation over the random quantization residuals, we can decouple the summation due to linearity:

$$\mathbb{E}\left[\sum_{j=1}^{n}\xi_j\right] \geq \frac{c}{2}\cdot\|\mathbf{H}\|_2\sum_{j=1}^{n}\frac{j}{n}\mathbb{E}\left[\|\boldsymbol{\epsilon}_j\|_2^2\right]. \tag{21}$$

Letting $\bar{\epsilon}^2 = \mathbb{E}[\|\boldsymbol{\epsilon}_j\|_2^2]$ denote the expected squared residual norm, the cumulative expected penalty evaluates to:

$$\frac{c}{2}\cdot\|\mathbf{H}\|_2\cdot\bar{\epsilon}^2\sum_{j=1}^{n}\frac{j}{n} = \frac{c}{2}\cdot\|\mathbf{H}\|_2\cdot\bar{\epsilon}^2\cdot\frac{n(n+1)}{2n}. \tag{22}$$

Since $\frac{n+1}{2} \geq \frac{n}{2}$, the expected total cumulative penalty is lower-bounded by $\frac{c}{4}\cdot n\cdot\bar{\epsilon}^2\cdot\|\mathbf{H}\|_2$. This establishes the asymptotic lower bound $\mathbb{E}[\mathcal{E}] = \Omega(n\cdot\bar{\epsilon}^2\cdot\|\mathbf{H}\|_2)$, showing that under the stated assumption, static analytical solvers cannot prevent compounding cross-column errors. □

C.3 Descent Condition Analysis under First-Order Updates

Theorem C.3 (Sufficient Condition for End-to-End Descent)**.** *Assume J is β-smooth and consider the trailing-weight update $\mathbf{W}^{(+)} = \mathbf{W} - \eta\mathbf{g}_{\text{surr}}$, where $\eta > 0$ and $\mathbf{g}_{\text{surr}} = \nabla\tilde{J}(\mathbf{W})$. Here $\tilde{J}$ denotes the selected objective $\mathcal{L}^{(b)}$ in Eq. 7, with all other weights fixed. If both $\mathbf{g}_{\text{surr}}$ and $\nabla J(\mathbf{W})$ are nonzero, then $J(\mathbf{W}^{(+)}) < J(\mathbf{W})$ is guaranteed whenever:*

$$\cos\theta > \frac{\eta\beta}{2}\cdot\frac{\|\mathbf{g}_{\text{surr}}\|}{\|\nabla J(\mathbf{W})\|}. \tag{23}$$

Proof of Theorem C.3. The descent lemma gives:

$$J(\mathbf{W} - \eta\mathbf{g}_{\text{surr}}) \leq J(\mathbf{W}) - \eta\langle\nabla J(\mathbf{W}), \mathbf{g}_{\text{surr}}\rangle + \frac{\eta^2\beta}{2}\|\mathbf{g}_{\text{surr}}\|^2. \tag{24}$$

It suffices for the upper bound to be smaller than $J(\mathbf{W})$, or equivalently:

$$\eta\langle\nabla J(\mathbf{W}), \mathbf{g}_{\text{surr}}\rangle > \frac{\eta^2\beta}{2}\|\mathbf{g}_{\text{surr}}\|^2. \tag{25}$$

Substituting $\langle \nabla J, \mathbf{g}_{\text{surr}} \rangle = \|\nabla J\| \|\mathbf{g}_{\text{surr}}\| \cos\theta$ and dividing by the positive quantity $\eta \|\nabla J\| \|\mathbf{g}_{\text{surr}}\|$ yields the stated sufficient condition. It is not necessary for actual descent, since the quadratic upper bound need not be tight. □

D Additional Experimental Details

D.1 Full Setup (continuation of Section 6.1)

Quantization and optimization. We use symmetric scalar weights with group size 128, activation ordering, Hessian damping 0.01, and four saliency groups for the analytical solver. REAL-Q uses Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) with independent parameter states, updating only the unquantized suffix of the current linear module; other linear modules remain fixed, including when the loss spans the next transformer block. Each Block-GD step uses 32 calibration sequences, cycling through the stored order on the first pass and reshuffling after each pass; both sliding-window terms use the same mini-batch. A4KV4 denotes 4-bit input activations and K/V cache, each symmetric per-token with clipping 0.9; queries are not separately quantized. For W4A4KV4, Block-GD uses A/V and K fake quantization in the forward pass and an identity straight-through estimator (STE) in the backward pass, with quantization parameters treated as constants.

Stage-0 Fisher pre-computation. Before quantization, we run the frozen full-precision model in evaluation mode on calibration mini-batches. At each output position, we sample a target from its full-vocabulary predictive distribution using fixed random seeds and backpropagate the summed NLL. Hooks at transformer-block outputs accumulate $\sum_t \mathbf{g}_t^{(\ell)} \mathbf{g}_t^{(\ell)\top}$ in fp32; after undoing loss scaling, we divide by the total token count and cache one bf16 matrix per block. If distributed, sums and counts are reduced across workers before normalization. Separate linear-output hooks collect grouped squared-gradient saliency for the analytical solver. No per-token Fisher matrices are retained.

Layer-wise schedule. For weight-only quantization, the main LR η in Table 6 is the reverse-cosine endpoint:

$$\eta_\ell = 0.01\eta + 0.99\eta \sin\left(\frac{\pi}{2} \frac{\ell - 1}{L - 1}\right). \tag{26}$$

W4A4KV4 uses a constant main LR. The final transformer block instead optimizes the true KL against the LM head, with its independent LR specified in Section 6.1.

Baseline configurations. EfficientQAT runs both of its stages. Block-AP trains all block parameters for 2 epochs with batch size 2 under fp16 AMP, using a cosine schedule with minimum-LR ratio 0.05, quantizer LR 10^{-4}, and weight LR 10^{-5} for W4/W3 and 2×10^{-5} for W2, optimizing block-output MSE reconstruction. E2E-QP then trains only the quantized-linear scales for 1 epoch in bf16 with AdamW, micro-batch 4 and 8 gradient-accumulation steps (effective batch 32), cosine schedule with warmup ratio 0.03, gradient clipping 0.3, and scale LR 10^{-5} for W4/W3 and 2×10^{-5} for W2; scales are projected to stay positive with minimum 10^{-4}. Its weight quantizer is symmetric max-abs-initialized uniform scalar with group size 128 and learnable scale but fixed zero point. Because our shared calibration budget is far smaller than YAQA's official configuration, we follow the claim in their paper and adopt the Sketch-A estimator. YAQA uses the Sketch-A Kronecker Hessian with 6 power iterations, 32 collection iterations, batch size 8 per rank at world size 1, and fp32 collection without fp64 accumulation; its solver is anti-diagonal LDLQ with $\sigma_{\text{reg}} = 0.01$, tile shape $(t_x, t_y) = (1, 128)$, fp64 Hessian solve, fp32 weight work dtype, native two-sided RHT, and no fine-tuning epochs. TurboBoA uses the grouped block-V solver with 16 quantization rows per group, $\alpha = 0.25$, ΔX compensation enabled, adaptive and refined quantization parameters, a single solver iteration, column-wise activation ordering with row ordering disabled, and the mean-Jacobian K-FAC mode for the Qwen3 Q/K RMSNorm Hessian.

Calibration and evaluation. All methods read the same saved WikiText-2 training-token tensor for each model; calibration sampling and rotation seeds are fixed at 1 and 0. KL/PPL use the WikiText-2 test split. We use EleutherAI's LM Evaluation Harness (Gao et al., 2023) for QA and reasoning evaluation. QA is reported only as the arithmetic mean over ARC-C/E (Clark et al., 2018), BoolQ (Clark et al., 2019), C-Eval (Huang et al., 2023), HellaSwag (Zellers et al., 2019), LAMBADA

(Paperno et al., 2016), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2020), with ARC-C and ARC-E counted separately. Per-task reasoning results cover GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), and HumanEval+ (Chen et al., 2021; Liu et al., 2023). We evaluate their full test sets with zero-shot chat prompts, greedy decoding with one completion, evaluation seed 1234, and generation limits of 1024/2048/2048 tokens, respectively; Qwen3 thinking is enabled. HumanEval+ reports official plus-test pass@1, not base-test accuracy.

D.2 Experimental Details for Loss Approximation Quality

This section provides the detailed setup for the gradient alignment evaluation presented in Section 6.4 (Figure 3).

We evaluate on Qwen3-0.6B at RTN W4A16. The model is processed transformer block by transformer block in the standard sequential quantization order. Immediately after the internal linear modules of block ℓ are quantized, we freeze the current model state to compute the exact analytical gradients. Specifically, for each linear module within block ℓ, we calculate:

(i) The **true end-to-end KL gradient** with respect to the module's weights, obtained via a full backward pass through all subsequent layers and the language modeling head over the calibration mini-batch.

(ii) The **surrogate gradients** produced by the respective proxy losses: **REAL-Q Fisher MSE** (our aggregated full Fisher), **Saliency-guided MSE** (the saliency-weighted per-module MSE optimized by GuidedQuant), and **Module MSE** (the uniform per-module reconstruction MSE optimized by GPTQ / GPTAQ).

We then compute the cosine similarity between the surrogate gradients and the true KL gradient. As observed in the per-module breakdown in the main text (Figure 3), while REAL-Q maintains strong overall alignment, its fidelity degrades slightly on FFN modules (particularly `up_proj` and `down_proj`) compared to attention modules. This is primarily due to the SwiGLU non-linearity, which makes the block-wise truncated second-order expansion a locally poorer approximation of the true downstream loss. This architectural heterogeneity directly motivates Future-work (1) discussed in Section 7.

D.3 Quantization Cost

Table 5 reports the end-to-end quantization time for models of different sizes under different quantization settings, including both pre-computation and quantization, measured on a single NVIDIA B200 GPU.

GPU-hours in Table 6 use the authors' released implementations for baselines and a system-optimized implementation for REAL-Q. They reflect implementation costs, including kernel and memory-management differences, rather than theoretical computational complexity alone.

Table 5: End-to-end REAL-Q quantization cost on a single NVIDIA B200 GPU (GPU-hours).

Model	W3A16	W2A16	W4A4KV4
LLaMA-3.1-8B-Instruct	0.594	0.595	0.682
Qwen3-4B	0.371	0.375	0.368
Qwen3-8B	0.660	0.663	0.710
Qwen3-32B	3.314	3.300	3.339

D.4 Memory for Fisher Aggregation

Caching one full $d \times d$ aggregated Fisher matrix per transformer block requires $O(Ld^2)$ storage, independent of sequence length. For bf16 storage, the analytical budget is $2Ld^2$ bytes: approximately 450 MiB for Qwen3-4B, 1 GiB for LLaMA-3.1-8B, 1.13 GiB for Qwen3-8B, and 3.13 GiB for Qwen3-32B. These are matrix-storage budgets, not measured peak GPU memory; end-to-end pre-computation additionally requires model and activation memory. However, this additional footprint

can be mitigated through offloading, activation recomputation, or fully sharded data parallelism (FSDP) (Zhao et al., 2023).

E Complete Results for All Twelve Settings

Table 6: Full results on twelve settings. KL ($\times 10^{-2}$) and PPL: WikiText-2; QA: ten-task mean; MATH: MATH-500; HE+: HumanEval+; Mean: the three reasoning accuracies (%). Bold marks the best quantized value per setting (including displayed ties); bf16 is a reference. REAL-Q LRs follow §6.1; its GPU-hours use the end-to-end B200 measurements (§D.3).

Method	KL↓	PPL↓	QA↑	GSM8K↑	MATH↑	HE+↑	Mean↑	GPU-h↓	LR
LLaMA-3.1-8B-Instruct — W3A16									
bf16	—	7.21	67.21	85.67	40.60	53.05	59.77	—	—
GPTAQ	11.86	8.12	64.17	70.28	23.60	29.27	41.05	1.561	—
GuidedQuant	11.68	8.11	64.49	72.10	**29.00**	35.37	45.49	3.951	—
ResComp	12.40	8.17	64.23	73.31	24.40	30.49	42.73	0.330	—
TurboBoA	10.95	8.05	64.08	**75.82**	**29.00**	35.98	46.93	0.726	—
EfficientQAT	21.65	**7.92**	62.19	66.26	21.20	32.93	40.13	**0.197**	—
YAQA	193.54	50.43	50.67	1.06	5.80	0.00	2.29	6.468	—
REAL-Q (ours)	**9.71**	7.94	**64.78**	73.77	26.40	**42.68**	**47.62**	0.594	5×10^{-6}
LLaMA-3.1-8B-Instruct — W2A16									
bf16	—	7.21	67.21	85.67	40.60	53.05	59.77	—	—
GPTAQ	71.93	14.56	42.31	1.52	1.60	**0.00**	1.04	1.568	—
GuidedQuant	85.18	16.76	41.80	1.06	2.00	**0.00**	1.02	4.458	—
ResComp	75.78	15.19	40.92	0.91	2.00	**0.00**	0.97	0.337	—
TurboBoA	93.61	18.15	39.96	0.00	0.00	**0.00**	0.00	0.730	—
EfficientQAT	366.74	257.20	30.26	0.00	0.00	**0.00**	0.00	**0.201**	—
YAQA	1077.40	3.48×10^{5}	33.06	0.00	0.00	**0.00**	0.00	6.493	—
REAL-Q (ours)	**54.51**	**12.32**	**48.66**	**2.88**	**3.60**	**0.00**	**2.16**	0.595	3.5×10^{-5}
LLaMA-3.1-8B-Instruct — W4A4KV4									
bf16	—	7.21	67.21	85.67	40.60	53.05	59.77	—	—
GPTAQ	21.08	8.85	61.46	**69.60**	20.80	**39.02**	43.14	0.912	—
GuidedQuant	25.72	9.38	61.24	68.92	22.00	30.49	40.47	2.120	—
ResComp	21.64	8.93	61.59	68.76	22.80	35.98	42.51	0.346	—
TurboBoA	21.39	8.87	61.97	68.61	22.60	35.98	42.40	0.656	—
EfficientQAT	151.13	29.12	40.55	1.21	3.60	0.00	1.60	**0.288**	—
YAQA	365.06	275.24	33.25	0.53	0.40	0.00	0.31	6.463	—
REAL-Q (ours)	**19.75**	**8.72**	**62.22**	68.39	**23.40**	37.80	**43.20**	0.682	5×10^{-7}
Qwen3-4B — W3A16									
bf16	—	13.66	64.52	87.19	60.20	57.93	68.44	—	—
GPTAQ	19.18	15.61	59.81	81.96	51.00	31.71	54.89	1.110	—
GuidedQuant	17.28	15.05	60.74	82.64	**54.80**	34.76	57.40	2.887	—
ResComp	19.81	15.53	60.56	78.32	44.40	27.44	50.05	0.245	—
TurboBoA	28.32	16.56	60.76	80.06	48.20	18.90	49.05	0.495	—
EfficientQAT	33.35	**13.18**	57.39	56.33	27.80	0.00	28.04	**0.135**	—
YAQA	16.97	15.22	59.75	83.24	51.60	30.49	55.11	4.562	—
REAL-Q (ours)	**14.98**	14.08	**60.86**	**84.84**	51.40	**37.20**	**57.81**	0.371	4×10^{-6}
Qwen3-4B — W2A16									
bf16	—	13.66	64.52	87.19	60.20	57.93	68.44	—	—
GPTAQ	101.51	28.80	36.64	0.38	0.40	**0.00**	0.26	1.204	—
GuidedQuant	107.10	26.06	35.88	1.06	1.60	**0.00**	0.89	3.414	—
ResComp	107.11	30.51	36.37	0.23	0.80	**0.00**	0.34	0.248	—
TurboBoA	184.26	65.97	37.02	0.00	0.00	**0.00**	0.00	0.514	—
EfficientQAT	451.55	666.11	29.98	0.08	0.20	**0.00**	0.09	**0.139**	—
YAQA	161.95	41.73	32.49	**1.74**	**4.00**	**0.00**	**1.91**	4.573	—
REAL-Q (ours)	**65.99**	**19.10**	**41.48**	1.44	2.00	**0.00**	1.15	0.375	3.5×10^{-5}
Qwen3-4B — W4A4KV4									
bf16	—	13.66	64.52	87.19	60.20	57.93	68.44	—	—
GPTAQ	45.56	18.93	53.53	60.35	35.20	4.88	33.48	0.780	—
GuidedQuant	43.84	17.71	54.17	56.48	41.00	17.68	38.39	1.884	—
ResComp	41.67	18.05	55.02	66.41	40.60	9.15	38.72	0.261	—
TurboBoA	59.51	20.96	51.66	37.91	19.80	3.66	20.46	0.404	—
EfficientQAT	594.45	2059.36	30.12	0.15	0.20	0.00	0.12	**0.191**	—
YAQA	809.10	1.87×10^{4}	29.60	1.36	3.00	0.00	1.45	4.576	—
REAL-Q (ours)	**33.01**	**15.16**	**57.73**	**78.70**	**48.60**	**28.66**	**51.98**	0.368	4×10^{-6}
Qwen3-8B — W3A16									
bf16	—	9.72	68.20	85.37	56.80	50.61	64.26	—	—
GPTAQ	13.20	10.74	64.47	79.00	49.80	22.56	50.45	1.522	—
GuidedQuant	12.49	10.40	65.17	82.18	51.00	26.22	53.13	4.169	—
ResComp	14.46	10.80	64.09	82.56	46.80	23.17	50.84	0.338	—
TurboBoA	16.29	10.91	63.98	**89.31**	**55.00**	27.44	57.25	0.782	—
EfficientQAT	25.44	**10.13**	63.12	74.98	48.80	14.02	45.94	**0.213**	—
YAQA	14.03	10.53	64.99	82.11	51.20	35.37	56.22	6.906	—
REAL-Q (ours)	**11.57**	10.35	**65.51**	87.04	52.00	**38.41**	**59.15**	0.660	6×10^{-6}

Continued on next page

Table 6 (continued)

Method	KL↓	PPL↓	QA↑	GSM8K↑	MATH↑	HE+↑	Mean↑	GPU-h↓	LR
Qwen3-8B — W2A16									
bf16	—	9.72	68.20	85.37	56.80	50.61	64.26	—	—
GPTAQ	70.34	16.70	43.43	1.36	4.40	**0.00**	1.92	1.703	—
GuidedQuant	73.89	15.61	41.20	1.14	3.60	**0.00**	1.58	4.747	—
ResComp	68.28	16.11	41.94	1.36	2.00	**0.00**	1.12	0.351	—
TurboBoA	507.65	1013.21	30.54	0.91	4.20	**0.00**	1.70	0.766	—
EfficientQAT	401.43	353.35	29.77	1.14	4.00	**0.00**	1.71	**0.218**	—
YAQA	1941.40	9.23×10^{8}	32.13	0.00	0.00	**0.00**	0.00	6.866	—
REAL-Q (ours)	**56.45**	**14.99**	**48.07**	**2.58**	**4.80**	**0.00**	**2.46**	0.663	4×10^{-5}
Qwen3-8B — W4A4KV4									
bf16	—	9.72	68.20	85.37	56.80	50.61	64.26	—	—
GPTAQ	30.57	12.18	60.49	82.94	50.40	23.17	52.17	0.978	—
GuidedQuant	30.93	11.74	60.99	72.63	47.40	17.07	45.70	2.575	—
ResComp	29.62	12.00	60.68	80.59	47.60	22.56	50.25	0.355	—
TurboBoA	36.67	12.63	60.73	76.42	45.60	20.73	47.58	0.710	—
EfficientQAT	616.60	2573.78	30.53	0.08	1.40	0.00	0.49	**0.318**	—
YAQA	797.29	1.57×10^{4}	30.99	1.59	5.40	0.00	2.33	6.754	—
REAL-Q (ours)	**25.42**	**11.35**	**63.29**	**87.95**	**52.60**	**37.20**	**59.25**	0.710	10^{-6}
Qwen3-32B — W3A16									
bf16	—	7.61	71.96	93.10	61.40	48.17	67.56	—	—
GPTAQ	15.58	8.03	70.97	93.48	59.00	43.90	65.46	4.517	—
GuidedQuant	14.96	8.03	70.66	92.87	58.20	36.59	62.55	14.329	—
ResComp	15.25	8.00	71.40	91.51	57.20	34.76	61.15	1.358	—
TurboBoA	428.87	385.14	34.36	1.14	3.60	0.00	1.58	2.906	—
EfficientQAT	25.86	**7.65**	68.73	84.91	53.60	33.54	57.35	**0.668**	—
YAQA	14.15	8.02	70.48	**93.86**	**61.00**	32.93	62.60	34.012	—
REAL-Q (ours)	**12.94**	7.76	**71.41**	92.65	57.20	**50.61**	**66.82**	3.314	1.4×10^{-5}
Qwen3-32B — W2A16									
bf16	—	7.61	71.96	93.10	61.40	48.17	67.56	—	—
GPTAQ	71.57	12.16	53.02	5.61	2.00	0.00	2.54	5.098	—
GuidedQuant	66.48	11.43	51.37	12.66	4.00	0.61	5.76	17.569	—
ResComp	64.12	11.39	50.95	4.09	3.20	0.00	2.43	1.377	—
TurboBoA	98.73	15.83	55.46	13.57	4.60	0.61	6.26	3.001	—
EfficientQAT	273.53	82.73	31.94	0.45	0.40	0.00	0.28	**0.630**	—
YAQA	1590.48	3.49×10^{7}	32.27	0.00	0.00	0.00	0.00	34.176	—
REAL-Q (ours)	**48.25**	**9.83**	**60.14**	**34.57**	**6.80**	**4.88**	**15.42**	3.300	3.5×10^{-5}
Qwen3-32B — W4A4KV4									
bf16	—	7.61	71.96	93.10	61.40	48.17	67.56	—	—
GPTAQ	41.08	9.44	64.87	88.25	54.40	15.24	52.63	3.098	—
GuidedQuant	44.82	9.81	63.79	88.70	54.40	17.07	53.39	7.467	—
ResComp	40.13	9.41	65.20	90.07	55.40	15.85	53.77	1.392	—
TurboBoA	40.40	9.35	66.76	91.21	56.20	23.78	57.06	2.804	—
EfficientQAT	871.47	2.91×10^{4}	30.75	0.53	0.00	0.00	0.18	**0.998**	—
YAQA	1249.63	1.41×10^{6}	30.97	1.14	1.00	0.00	0.71	33.646	—
REAL-Q (ours)	**25.60**	**8.28**	**68.70**	**93.56**	**61.20**	**39.63**	**64.80**	3.339	4×10^{-6}