GSQ: Highly-Accurate Low-Precision Scalar Quantization for LLMs via Gumbel-Softmax Sampling

Alireza Dadgarnia[†] ISTA
Soroush Tabesh ISTA
Mahdi Nikdan ISTA
Michael Helcig ETH Zürich
Eldar Kurtić ISTA & Red Hat AI
Maximilian Kleinegger TU Wien & ISTA
Dan Alistarh[†] ISTA & Red Hat AI

Abstract

Quantization has become a standard tool for efficient LLM deployment, especially for local inference, where models are now routinely served at 2–3 bits per parameter. The state of the art is currently split into simple scalar quantization techniques, such as GPTQ or AWQ, which are widely deployed but plateau in accuracy at 3–4 bits per parameter (bpp), and "second-generation" vector- or trellis-quantized methods, such as QTIP, GPTVQ and AQLM, which push the accuracy frontier but are notoriously hard to implement and to scale. In this paper, we ask whether this gap is fundamental, or whether a carefully optimized *scalar* quantizer can recover most of it. We answer in the affirmative, by introducing GSQ (Gumbel-Softmax Quantization), a post-training scalar quantization method which jointly learns the per-coordinate grid assignments and the per-group scales using a Gumbel-Softmax relaxation of the discrete grid. GSQ matches the cardinality of the relaxation to the small number of levels available in the target bit-width regime (e.g., 3–8 levels for ternary and 3 bpp, respectively), making optimization tractable. Practically, on the standard Llama-3.1-8B/70B-Instruct models, GSQ closes most of the gap between scalar quantization and the QTIP frontier at 2 and 3 bits, while using a symmetric scalar grid with group-wise quantization, and thus remains compatible with existing scalar inference kernels. We further show that the same discrete-assignment optimization can be applied to practical GGUF K-Quant checkpoints: starting from publicly released GGUF models, GSQ improves accuracy while projecting the result back into the same deployment format. Finally, GSQ scales to trillion-scale Mixture-of-Experts models such as Kimi-K2.5, where vector-quantized methods are difficult to apply. The source code is publicly available at `https://github.com/IST-DASLab/GSQ`.

1 Introduction

The memory and bandwidth costs of LLM inference have made quantization a standard approach for efficient deployment. Among the many quantization directions studied [26, 48, 17, 18, 45, 3, 50, 61, 70, 6, 64, 65, 20, 67, 8], *weight-only* quantization has emerged as the standard for *local* deployment, where the bottleneck is memory rather than compute, and where engines such as llama.cpp [29] and Ollama [56] have made models broadly accessible. It is now common to have versions of large open models at around 2–3 bits per parameter, and a whole open-source ecosystem of "quants" has emerged around repositories such as Hugging Face [39, 2]. In practice, many of these models are distributed in deployment-oriented formats such as GGUF K-Quants, which are directly consumed by local inference engines and therefore impose strong compatibility constraints on any improved quantization method.

[†]Corresponding authors: alirezadadgarnia1378@gmail.com, dan.alistarh@ist.ac.at

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

Existing techniques. Broadly, weight quantization techniques can be viewed as two successive "waves". The *first wave* investigated scalar (1D) quantization methods, with llama.cpp [29], bitsandbytes [16, 17], GPTQ [26], and AWQ [48] being among the most popular methods. These approaches round each weight independently to a small uniform grid, are simple to implement, and benefit from highly optimized unpacking kernels; as a result, they enjoy broad adoption. Their main limitation is *accuracy*: scalar quantization has hit an "error wall" around 3–4 bits per parameter (bpp), below which output quality degrades [26, 8]. The *second wave*, by contrast, has focused on more expressive vector-quantized or trellis representations, including AQLM [20], QuIP# [64], QTIP [65] with its implementation exllamav3 [66], and GPTVQ [67]. By minimizing reconstruction MSE jointly over *groups* of weights, these methods substantially reduce accuracy loss at 2–3 bits per parameter; yet, unfortunately, the resulting representations are considerably harder to implement and scale. Specifically, Tseng et al. [65] observed that, although VQ and trellis methods yield major memory savings, they lead to only small decoding speedups vs BF16 due to format complexity.

We are thus left with a clear gap between accuracy and practicality. The question we ask is: *Can scalar quantization bridge most of the accuracy gap to complex vector- or trellis-based techniques, while remaining a drop-in replacement for existing popular formats?*

Our approach. We answer this question affirmatively, by proposing GSQ (Gumbel-Softmax Quantization), a post-training quantization method which closes most of the gap to second-wave techniques while preserving deployment-friendly scalar structure. In its main form, GSQ produces symmetric, group-wise, b-bit weights drawn from a small uniform grid, and is therefore directly compatible with existing scalar inference kernels. Yet, on the accuracy side, it substantially improves over previous scalar-focused methods such as GPTQ [26], AWQ [48], QuIP [6], and EfficientQAT [8] at 2 and 3 bits, where it recovers the majority of the gap to the strongest existing baselines. Beyond this standard scalar setting, we also show that the same local discrete-assignment view can be used to improve existing GGUF K-Quant checkpoints: GSQ starts from a GGUF model, optimizes its assignments, and projects the result back into the same K-Quant format, preserving compatibility with GGUF inference stacks. GSQ also scales to very large Mixture-of-Experts (MoE) models, including the trillion-parameter-scale Kimi-K2 [62], where second-wave methods have so far not been applied.

Method overview. The key idea is that we want to reformulate layer-wise reconstruction as a *differentiable* discrete-assignment problem. For each weight coordinate, we introduce a small set of trainable logits over the candidate grid points, and obtain a soft quantized weight via Gumbel-Softmax sampling [53, 42]. The resulting reconstruction loss is fully differentiable in both the per-group scales and the discrete assignments, and can be optimized jointly via gradient-based methods. Sampling allows competing assignments to be explored during optimization, while annealing makes each relaxed sample increasingly discrete. The final model uses deterministic assignments obtained from the learned logits. Our ablations show that stochastic assignment optimization improves hard reconstruction error over deterministic softmax under the tested schedule (Appendix L.4).

One key observation about grid size is that Gumbel-Softmax relaxation is a natural fit for low-bit scalar quantization. In the regimes we are interested in (e.g. ternary and 2-bit), the cardinality of the per-coordinate grid is small, so a Gumbel-Softmax distribution over the entire grid introduces only a few logits per weight and can be optimized end-to-end at LLM scale. At higher bit-widths (e.g. 3-4 bpp), where the grid grows exponentially, we replace the global relaxation with a *new local-shift formulation* in which only a small number of nearest grid points around the assignment are considered, keeping overheads small. This combination allows GSQ to operate uniformly across the entire low- to mid-bit range.

Accuracy results. Empirically, GSQ sets a new state of the art for scalar quantization at low bit-widths. For standard benchmark experiments on Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct, at 2 bits per parameter, GSQ improves average zero-shot accuracy by a remarkable 4.76 and 4.14 points, respectively over the best scalar baseline (EfficientQAT), and trails QTIP, the strongest and most complex prior method, by only 1.33 and 1.68 points, respectively. At 3 bits, the picture is similar: GSQ matches or surpasses all scalar baselines, and is essentially on par with QTIP on the 70B model. Notably, these results are obtained with *symmetric* group-wise quantization, without any zero-point parameters, in contrast to most baselines, which is direct evidence that the gains come from better optimization of the discrete assignments rather than from a more flexible quantizer. On the same models, ternary (1.58-bit) GSQ already exceeds all scalar baselines even if they are run at higher 2-bit precision. Furthermore, because GSQ produces standard scalar layers, it naturally supports non-uniform bit allocation across layers. Using RCO [34] to search per-layer bit assignments

on Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct, mixed 2/3-bit configurations at 2.37 and 2.62 average bits per parameter retain most of the 3-bit accuracy while substantially reducing model size. We provide speedup results via Humming kernels [40].

We further evaluate GSQ in the popular GGUF K-Quant setting. Starting from Unsloth GGUF checkpoints for Qwen3-8B, GSQ improves both Q3_K_M and Q2_K models while preserving the same output format. The gains are especially large in the aggressive Q2_K setting, where the average score across AIME25, GPQA Diamond, and MMLU-Pro improves from 50.03 to 56.28.

Since GSQ only requires per-coordinate discrete optimization and per-group scales, its memory footprint is close to that of a standard scalar PTQ approach. This allows us to directly apply GSQ to massive Mixture-of-Experts models such as Kimi-K2/2.5, where the codebook training and per-block updates become prohibitively expensive. To our knowledge, GSQ is the first method to obtain low-bit, close-to-lossless quantization of trillion-parameter MoE models using a fully scalar, kernel-compatible format.

2 Related work

Early post-training quantization (PTQ) methods such as LLM.int8() [17] showed that quantizing 99.9% of features to INT8 while keeping the outliers in 16-bit achieves significant memory and runtime improvements. Frantar et al. [26] introduced GPTQ, which uses second-order information to minimize layer-wise quantization error based on the Optimal Brain Surgeon framework [33], while AWQ [48] used activation statistics to identify and protect a small set of weights. Outlier-aware formats [18, 45] suggest keeping a small set of outlier weights in high-precision. More recently, rotation-based methods have emerged, enabling near-lossless 4-bit quantization of weight, activations, and KV cache [3, 50, 61]. As discussed, vector/codebook methods target 2-bit compression or less while retaining full-precision execution, where QuIP [6], QuIP# [64], QTIP [65], AQLM [20], and PV Tuning [54] are strong baselines.

Quantization-aware training (QAT) fine-tunes the model under simulated low-precision arithmetic. EfficientQAT [8] makes QAT practical for scalar quantization of LLMs by combining block-wise training of model parameters with a final end-to-end optimization of quantization parameters. Early work on training binary networks [14, 57] established that 1-bit weights are possible for general deep neural networks. For LLMs, BitNet [69] and follow-up work argue that training in ternary (1.58-bit) can be competitive with full precision. TernaryLLM [9] uses trainable scale and zero-point parameters along with a specialized information-theoretic knowledge distillation objective. In the post-training settings, PT^2-LLM [72] enables ternary quantization via iteratively alternating between refining the grid and rounding. Tequila [37] reactivates deadzone-trapped weights by re-introducing them as dynamic bias parameters. PTQTP [71] decomposes the weights into two trit-planes, achieving multiplication-free additive inference. PT-BitNet [32] first transforms the weights to make them quantizaition-friendly, then quantizes each weight block separately. For binary PTQ, BiLLM [38] compresses outlier weights by a binary residual approximation approach, while simply binarizing the remaining weights. DB-LLM [7] decomposes its 2-bit budget into two independent binaries. ARB-LLM [46] present alternating refined binarization to progressively update the binary parameters. PB-LLM [60] simply keeps the salient weights in high-precision, while binarizing the rest. PTQ1.61 [76] takes a similar approach, except that the salient weights are structured and are quantized to 4-bits. STBLLM [19] combines N:M sparsity with binarization of non-pruned weights and provides system support for this format.

MoE quantization must preserve both expert and router quality; QMoE [25] was the first to enable sub-1-bit compression at trillion-parameter scale with custom on-the-fly decoding kernels. MoQa [77] assigns different bit-width to each expert based on their sensitivity and distribution of tokens. MoPEQ [12] replaces the criteria with the more rigorous Hessian trace approximation. EAQuant [27] introduces expert smoothing to suppress activation outliers, aligns the logit distribution of the router to preserve expert selection, and balances calibration data across experts. Similarly, ExpertQuant [23] uses a Jaccard loss to ensure the top-k selected experts remain unchanged, while MoEQuant [36] specifically addresses the data imbalance problem. Additionally, EAC-MoE [11] not only calibrates the routers to preserve expert selection, but also suggests pruning less frequently used experts.

Because quantization and sparsity introduce discrete choices (rounding, masks), many approaches use differentiable proxies. The straight-through estimator (STE) [4] keeps a hard assignment in the forward pass and treats rounding as the identity in the backward pass; LSQ [21] uses it to learn quantizer step sizes. The rotation trick [24] propagates gradients through vector quantization using

a rotation and rescaling transformation that encodes the relative angle and magnitude between an input vector and its selected codeword. DiffQ [15] uses pseudo quantization noise to optimize bit allocation. For post-training rounding, AdaRound [55] learns whether each weight rounds up or down through a deterministic sigmoid relaxation. The Gumbel-Softmax relaxation has been applied to neural architecture search [35], and MaskLLM [22] uses it to learn semi-structured N:M masks end-to-end.

Relaxed Quantization (RQ) [51] applies Gumbel-Softmax relaxations to weight and activation quantization. RQ derives categorical probabilities by integrating a continuous logistic distribution over quantization bins, and jointly optimizes the underlying continuous values and quantizer parameters. It also introduces a local grid to reduce the cost of sampling at higher bit-widths. GSQ instead directly learns assignment logits over scalar grid points, or over a fixed set of index shifts around an initial quantized assignment, together with per-group scales.

3 The Gumbel-Softmax quantization (GSQ) method

Notation and Gumbel-Softmax Sampling. Let $f(\cdot; \mathbf{w})$ denote a function parameterized by weights $\mathbf{w} \in \mathbb{R}^d$ that we aim to compress, e.g. a single linear layer, a sub-module (such as a Transformer block), or an entire network. Given calibration input data $\mathbf{x}$, the objective is to find a parameterization $\hat{\mathbf{w}}$ that satisfies a constraint set $\mathcal{C}$ while minimizing the output reconstruction error, measured e.g. in the Frobenius norm $\|\cdot\|_F$:

$$\hat{\mathbf{w}} = \arg\min_{\bar{\mathbf{w}}} \ \|f(\mathbf{x}; \bar{\mathbf{w}}) - f(\mathbf{x}; \mathbf{w})\|_F^2 \quad \text{s.t.} \quad \bar{\mathbf{w}} \in \mathcal{C}. \tag{1}$$

Optimizing the objective in Equation 1 becomes challenging when the constraint set $\mathcal{C}$ includes discrete components, since standard gradient-based methods are not directly applicable. This difficulty arises naturally in model compression: quantization requires mapping weights to a discrete grid, while sparsity needs setting a specific subset of weights to zero. To address this challenge, our method leverages Gumbel-Softmax sampling [53, 42] to make the discrete selection process differentiable.

For candidates $\mathcal{D} = \{d_1, \ldots, d_K\}$, let ℓ_i denote the trainable assignment logits. Independent perturbations $g_i \sim \mathrm{Gumbel}(0, 1)$ provide an exact categorical sampler through the Gumbel–Max identity:

$$\Pr\left[\arg\max_j \{\kappa\ell_j + g_j\} = i\right] = \frac{\exp(\kappa\ell_i)}{\sum_j \exp(\kappa\ell_j)}.$$

Algorithm 1 replaces this hard selection with the differentiable relaxation $p_i = \mathrm{softmax}_i((\kappa\boldsymbol{\ell}+\mathbf{g})/\tau)$ and returns $\tilde{d} = \sum_i p_i d_i$. The logit scale κ controls the categorical sampling probabilities, whereas the temperature τ controls the softness of each relaxed sample. For fixed logits and κ, taking $\tau \to 0$ recovers a categorical sample. For deployment, we remove the perturbations and select the candidate with the largest learned logit.

The perturbations allow alternative assignments to be sampled throughout optimization. In contrast, a deterministic relaxation can concentrate around early choices and provide weak gradients for reconsidering them. At finite temperature, GSQ uses soft weights in the forward pass and differentiates through the relaxation, whereas the STE alternatives in our ablation use hard forward assignments (Appendix L.4). Variance-matched Gaussian, logistic, and uniform perturbations perform similarly to Gumbel perturbations in a separate study (Appendix L.5). We therefore choose Gumbel noise for its exact categorical-sampling interpretation, without claiming empirical superiority over other noise families.

This optimization introduces $|\mathcal{D}|$ learnable logits. In the special case where $\mathcal{D}$ contains only two elements, instead of introducing two separate logits, we use a single logit ℓ and assign $-\ell$ as the logit for the other element. The probability assigned to the candidate with logit $+\ell$ is $\mathrm{sigmoid}((2\kappa\ell + g_+ - g_-)/\tau)$. This halves the number of trainable parameters in the binary case and substantially lowers the memory overhead.

Algorithm 1: Gumbel-Softmax (GS) Sampling

Require: Finite set $\mathcal{D} = \{d_1, \ldots, d_n\}$
Require: Logits $\ell_1, \ldots, \ell_n$
Require: Temperature $\tau > 0$, noise scale $\kappa > 0$
Ensure: Soft sample $\tilde{d}$

for $i = 1$ **to** n **do**
  Draw $g_i \sim \mathrm{Gumbel}(0, 1)$
end for
for $i = 1$ **to** n **do**
  $p_i \leftarrow \dfrac{\exp\big((\kappa \ell_i + g_i)/\tau\big)}{\sum_{j=1}^{n} \exp\big((\kappa \ell_j + g_j)/\tau\big)}$
end for
return $\tilde{d} \leftarrow \sum_{i=1}^{n} p_i d_i$

Algorithm 2: Ternary GSQ

Require: Weights $\mathbf{w} \in \mathbb{R}^d$, calibration data $\mathbf{x}$
Require: Schedules $\{\tau_t\}_{t=1}^{T}$, $\{\kappa_t\}_{t=1}^{T}$
Ensure: Ternary weights $\hat{\mathbf{w}} \in \{-s, 0, s\}^d$

Initialize scale $s \in \mathbb{R}$, logits $\boldsymbol{\ell}^{(m)}, \boldsymbol{\ell}^{(b)} \in \mathbb{R}^d$
for $t = 1$ **to** T **do**
  for $i = 1$ **to** d **do**
    $\tilde{\mathbf{m}}_i \leftarrow \mathrm{GS}\big(\{0, 1\}, \{-\boldsymbol{\ell}_i^{(m)}, \boldsymbol{\ell}_i^{(m)}\}, \tau_t, \kappa_t\big)$
    $\tilde{\mathbf{b}}_i \leftarrow \mathrm{GS}\big(\{-1, 1\}, \{-\boldsymbol{\ell}_i^{(b)}, \boldsymbol{\ell}_i^{(b)}\}, \tau_t, \kappa_t\big)$
  end for
  $\tilde{\mathbf{w}} \leftarrow s \cdot \tilde{\mathbf{m}} \odot \tilde{\mathbf{b}}$
  $\mathcal{L} \leftarrow \|f(\mathbf{x}; \tilde{\mathbf{w}}) - f(\mathbf{x}; \mathbf{w})\|_F^2$
  Update $s, \boldsymbol{\ell}^{(m)}, \boldsymbol{\ell}^{(b)}$ using $\nabla \mathcal{L}$
end for
$\hat{\mathbf{m}}_i \leftarrow \mathbb{1}\{\boldsymbol{\ell}_i^{(m)} \geq 0\}, \quad \hat{\mathbf{b}}_i \leftarrow 2\mathbb{1}\{\boldsymbol{\ell}_i^{(b)} \geq 0\} - 1$
return $\hat{\mathbf{w}} \leftarrow s \cdot \hat{\mathbf{m}} \odot \hat{\mathbf{b}}$

Algorithms 1 and 2: The Gumbel-Softmax relaxation, denoted by GS in Algorithm 2, and its use inside ternary GSQ to learn differentiable mask and sign variables before hard thresholding.

3.1 The ternary quantization case

We begin by describing how Gumbel-Softmax sampling is used to compress the model parameters into a ternary quantization format. Specifically, we impose the following constraint in Objective 1:

$$\mathcal{C}_{\text{ternary}} = \big\{\bar{\mathbf{w}} \mid \bar{\mathbf{w}} = s \cdot \mathbf{m} \odot \mathbf{b}; s \in \mathbb{R}, \mathbf{m} \in \{0, 1\}^d, \mathbf{b} \in \{-1, 1\}^d\big\}. \tag{2}$$

Under this formulation, a ternary-quantized vector is parameterized by three components: a binary mask $\mathbf{m}$ indicating which entries are zero, a binary sign vector $\mathbf{b}$ specifying whether each nonzero entry is -1 or $+1$, and a scaling factor s. This parameterization introduces $2d$ binary decisions, which we relax using $2d$ instances of binary Gumbel-Softmax sampling. Concretely, we jointly optimize the scale s, the mask logits $\boldsymbol{\ell}^{(m)} \in \mathbb{R}^d$, and the sign logits $\boldsymbol{\ell}^{(b)} \in \mathbb{R}^d$. At each training step, $\mathbf{m}$ and $\mathbf{b}$ are obtained by applying Gumbel-Softmax sampling to $\boldsymbol{\ell}^{(m)}$ and $\boldsymbol{\ell}^{(b)}$, respectively. The full procedure is provided in Algorithm 2. Although the formulation above assumes a single shared scale value (i.e., symmetric global quantization), the same framework extends naturally to asymmetric and/or group-wise quantization with minor modifications.

Initialization. Instead of initializing the logits randomly, we warm-start from the GPTQ ternary solution $\mathbf{q}_{\text{GPTQ}} \in \{-1, 0, 1\}^d$ [26]. Recall that the mask logit $\boldsymbol{\ell}_i^{(m)}$ controls whether weight i is nonzero: a positive logit favors $\mathbf{m}_i = 1$ (active), while a negative logit favors $\mathbf{m}_i = 0$ (pruned). Similarly, the sign logit $\boldsymbol{\ell}_i^{(b)}$ controls the sign of the nonzero weight: a positive logit favors $\mathbf{b}_i = +1$ and a negative logit favors $\mathbf{b}_i = -1$. We therefore initialize each logit to reflect the corresponding GPTQ decision:

$$(\boldsymbol{\ell}_{\text{GPTQ}}^{(m)})_i = \begin{cases} +1.0, & \text{if } (\mathbf{q}_{\text{GPTQ}})_i \neq 0, \\ -1.0, & \text{if } (\mathbf{q}_{\text{GPTQ}})_i = 0, \end{cases} \qquad (\boldsymbol{\ell}_{\text{GPTQ}}^{(b)})_i = \begin{cases} +1.0, & \text{if } (\mathbf{q}_{\text{GPTQ}})_i = +1, \\ -1.0, & \text{if } (\mathbf{q}_{\text{GPTQ}})_i = -1, \\ 0.0, & \text{if } (\mathbf{q}_{\text{GPTQ}})_i = 0. \end{cases} \tag{3}$$

When $(\mathbf{q}_{\text{GPTQ}})_i = 0$, the sign logit is initialized to 0, since the sign is irrelevant at initialization but may become active during subsequent optimization if the mask flips to nonzero. To prevent getting stuck, we inject isotropic Gaussian noise into the logits before training. We initialize the mask and sign logits as

$$\boldsymbol{\ell} = \sigma_{\text{init}}\Big(\boldsymbol{\epsilon} + \alpha\, \boldsymbol{\ell}_{\text{GPTQ}}\Big), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4}$$

where $\alpha \in \mathbb{R}$ controls the strength of the GPTQ warm-start relative to the injected noise, and $\sigma_{\text{init}} \in \mathbb{R}$ sets the overall scale of the logits. We also initialize the quantization scale value s to the scale computed by GPTQ.

3.2 General scalar quantization

We now describe how GSQ extends to general scalar quantization. Suppose the goal is to quantize the model parameters to b bits using symmetric quantization with a single shared scale factor. In this

setting, the constraint set $\mathcal{C}$ in Objective 1 can be written as

$$\mathcal{C}_{b\text{-bit}} = \left\{ \bar{\mathbf{w}} \middle| \bar{\mathbf{w}} = s \cdot \mathbf{q};\ s \in \mathbb{R},\ \mathbf{q} \in \mathcal{G}_b^d \right\}, \tag{5}$$

where $\mathcal{G}_b$ denotes the ordered quantization grid, with cardinality $|\mathcal{G}_b| = 2^b$, specifying the set of values that each quantized parameter may take. This formulation imposes no structural restrictions on the grid and therefore accommodates both uniform and non-uniform quantization schemes.

To enable gradient-based optimization, we apply Gumbel–Softmax sampling independently to each of the d coordinates. Each such instance introduces 2^b trainable logits, i.e., the logits can be concatenated into $\boldsymbol{\ell} \in \mathbb{R}^{d \times 2^b}$. With this relaxation, we jointly optimize the logits $\boldsymbol{\ell}$ and the scale parameter s.

The 2-bit case. As a direct application, consider 2-bit uniform quantization with $\mathcal{G}_2 = \{-2, -1, 0, 1\}$. In this case, each coordinate is associated with 4 trainable logits for Gumbel–Softmax sampling. Together with the shared scale parameter s, this yields a total of $4d+1$ trainable parameters. Although the grid itself is skewed toward negative values, we allow the scale s to take negative values as well, thereby removing any inherent bias toward either side.

Higher bit-widths. As the bit-width b increases, the number of trainable logits, and consequently the required memory, grows exponentially in bpp. To address this, for $b > 2$, we use a local shift-based formulation, explained below and summarized in Figure 1. The key observation is that, each coordinate typically remains close to its initialized quantized value, and large jumps across the quantization grid rarely happen (see Appendix L.3). Instead of assigning a logit to every value in $\mathcal{G}_b$, we only learn a small relative shifts. Specifically, suppose we are given an initialized quantized vector $\mathbf{q}^0 \in \mathcal{G}_b^d$. For each coordinate $i \in \{1, \ldots, d\}$, let $j_i^0 \in \{1, \ldots, 2^b\}$ denote the index of the initialized grid point, i.e., $q_i^0 = (\mathcal{G}_b)_{j_i^0}$.

Instead of introducing 2^b logits for coordinate i, we introduce only 5 logits corresponding to a discrete shift $\delta_i \in \{-2, -1, 0, 1, 2\}$. Let $\boldsymbol{\ell}_\delta \in \mathbb{R}^{d \times 5}$ denote the corresponding trainable logits. At each training step, δ_i is obtained by applying Gumbel-Softmax sampling to the i-th row of $\boldsymbol{\ell}_\delta$. The resulting grid index is then $j_i = \mathrm{clip}(j_i^0 + \delta_i, 1, 2^b)$, where $\mathrm{clip}(x, a, b) = \min\{\max\{x, a\}, b\}$ clips the value into the valid range. The final quantized value is $q_i = (\mathcal{G}_b)_{j_i}$.

Equivalently, the constraint set becomes

$$\mathcal{C}_{b\text{-bit}}^{\text{shift}} = \left\{ \bar{\mathbf{w}} \middle| \bar{\mathbf{w}} = s \cdot \mathbf{q},\ s \in \mathbb{R},\ q_i = (\mathcal{G}_b)_{\mathrm{clip}(j_i^0 + \delta_i,\, 1,\, 2^b)},\ \delta_i \in \{-2, -1, 0, 1, 2\} \right\}. \tag{6}$$

Under this parameterization, each coordinate requires only 5 trainable logits rather than 2^b. Therefore, the total number of trainable parameters is reduced from $d \times 2^b + 1$ to $5d + 1$, making higher-bit optimization practical while still allowing each coordinate to move to nearby grid values.

Initialization. We distinguish the initial quantized assignment from its representation in the relaxed optimization. GPTQ supplies the initial assignments and scales. We initialize the assignment logits with a Gaussian-like prior over candidate distance, centered on the selected grid value for 2-bit quantization and on zero shift for higher bit-widths (Appendix A). This prior favors the initial assignment while assigning greater probability to nearby alternatives than to distant ones.

RTN prior assignments perform nearly as well as GPTQ, whereas random assignments fail (Appendix L.2). With GPTQ assignments fixed, the Gaussian-like prior substantially outperforms an initialization that boosts only the selected candidate's logit. Thus, GSQ needs a meaningful initial assignment, though not GPTQ specifically, and benefits from a distance-aware logit initialization.

3.3 Implementation details

Objective. Unlike most PTQ methods, GSQ is not tied to a layerwise quadratic objective. In principle, it can optimize richer objectives directly over the quantized parameters, such as block-level reconstruction losses or model-level task-aware losses. This flexibility, however, comes at the cost of additional memory, since GSQ introduces auxiliary trainable logits whose footprint is typically 2-5$\times$ that of the weights being quantized. As a result, jointly optimizing the entire model is prohibitively expensive for large Transformers and MoEs.

In practice, we adopt a combination of objectives during optimization. For example, in the ternary quantization setting, where the GPTQ initialization is particularly weak, we first warm up the logits (initialized from GPTQ) using a cheaper layerwise quadratic objective applied independently to

each linear layer. This is then followed by a blockwise or expertwise optimization stage. The exact procedure depends on the model and setting, and is described in Section 4.

We note that this differs from approaches that first quantize each layer independently, and afterwards perform a limited block-level tuning over a small subset of continuous parameters such as the quantization scales [64, 20]. In GSQ, the discrete assignments and their associated continuous parameters are optimized jointly.

Optimizer. The Gumbel-Softmax relaxation can enter a saturated regime in which the relaxed categorical distribution becomes nearly one-hot, for example due to temperature annealing or growing logit gaps; see Appendix B. [22] mitigate this through problem-specific regularization and by increasing the ϵ hyperparameter in AdamW (e.g., to 10^{-5}). When gradients vanish, AdamW effectively stalls. We address this differently, by using Lion [10], which is much less sensitive to vanishing gradients.

Gradient accumulation. In our setting, gradient accumulation plays an additional role, beyond memory savings: because we *resample* the Gumbel noise independently for each forward pass, averaging gradients across micro-batches reduces sampling variance. This leads to noticeably more stable optimization.

4 Experiments

4.1 Experimental setup

Models. We evaluate on two standard dense models, Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct [30], as well as the MoE model Kimi-K2.5 [63]. For the Llama models, we quantize all non-embedding and non-head linear layers, with one exception: in the 8B model, we find the `down_proj` of the second block to be unstable under high compression and leave it in full precision. For Kimi-K2.5, we quantize only the non-shared expert weights while leaving the shared experts untouched; we also skip the vision-related components and only evaluate the language model. Finally, to evaluate compatibility with existing local-deployment formats, we also apply GSQ to publicly available GGUF K-Quant checkpoints of Qwen3-8B, using the released quantized model as initialization and preserving the same GGUF format after optimization.

Quantization configuration. For scalar experiments, we focus on 2-bit and 3-bit weight-only quantization with a group size of 128. GSQ uses a symmetric scalar quantizer, where each group shares a single scale value. Groups are formed row-wise over consecutive entries, following the standard packing layout used in prior work. We also include brief ternary quantization experiments (i.e., 1.58-bit). In addition, we evaluate *non-uniform* bit allocation, in which different layers are assigned different bit-widths so as to achieve a fractional average rate such as 2.37 or 2.62 bits per parameter.

Training details. We perform block-wise optimization with a Gumbel-Softmax relaxation over the discrete assignments, followed by scale-only fine-tuning only for 2-bit Llama experiments. Full hyperparameters and sensitivity results appear in Appendices G.1 and L.6.

Within-block staging. For dense models, we found that optimizing all quantized layers in a Transformer block jointly under a block reconstruction loss was suboptimal, despite being the natural formulation used in prior work [20, 8]. We instead use a staged schedule: query and key projections are first optimized independently with linear reconstruction losses; value and output projections are then optimized jointly under the self-attention output reconstruction loss; and the MLP projections are optimized last under the full block reconstruction loss. After a block is quantized, it is frozen, and subsequent blocks are optimized using inputs from the already-quantized prefix, making the procedure aware of accumulated quantization error. We provide further motivation and details in Appendix C.

Calibration data and training budget. For calibration data, we use FineWeb-Edu [52] in Llama experiments, and OpenThoughts [31] for Kimi K2.5 experiments. Unless otherwise stated, we use 4096 sequences of length 4096. For block-wise training, we run 20 epochs for the Llama models and 10 epochs for Kimi-K2.5. For end-to-end scale-only fine-tuning on the Llama models, we run a single epoch over the same 4096 sequences.

Scalar quantization baselines. We include GPTQ [26], QuIP [6], and EfficientQAT [8] as baselines; these methods are allowed to use asymmetric quantization with per-group zero-points, giving them strictly more representational freedom than GSQ. Whenever a released quantized checkpoint is available, we re-evaluate it directly under our evaluation pipeline; otherwise, we run the official

Table 1: Zero-shot accuracy on Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct under ternary, 2-bit, 3-bit, and non-uniform quantization. Bit/param excludes non-quantized tensors.

	Llama-3.1-8B-Instruct							Llama-3.1-70B-Instruct						
Method	bit/param	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.	bit/param	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
FP16/BF16	16	55.12	79.63	79.16	80.85	73.80	73.71	16	63.48	83.92	84.58	83.95	79.01	78.99
QTIP	3.00	53.92	79.42	78.30	80.25	72.14	72.81	3.00	61.77	82.79	84.21	83.79	78.30	78.17
GPTQ	3.25	39.68	58.67	64.24	67.03	66.30	59.18	3.25	60.75	80.64	82.37	**83.30**	77.98	77.01
QuIP	3.25	52.30	76.68	75.37	78.84	71.90	71.02	3.25	**62.12**	82.45	82.83	82.10	78.06	77.51
EfficientQAT	3.25	**52.99**	**78.91**	**76.85**	79.76	71.59	72.02	3.25	61.86	**83.88**	82.79	82.48	76.01	77.40
GSQ (ours)	3.13	**52.99**	78.37	76.66	**80.03**	**73.56**	**72.32**	3.13	**62.12**	82.83	**83.30**	82.75	**78.93**	**77.99**
GSQ (ours)	2.62^{NU}	49.23	74.79	74.83	79.00	70.24	69.62	2.62^{NU}	59.60	81.40	82.90	82.90	80.40	77.50
GSQ (ours)	2.37^{NU}	50.26	74.41	74.81	78.13	69.61	69.44	2.37^{NU}	60.20	81.40	82.60	82.40	80.60	77.40
QTIP	2.00	50.68	75.42	75.02	78.18	70.09	69.88	2.00	61.69	81.69	82.95	82.43	77.51	77.25
PV-Tuning	2.27	50.26	73.91	75.28	79.16	70.56	69.83	2.07	58.62	80.72	82.72	81.56	77.74	76.27
GPTQ	2.25	24.91	27.61	30.50	52.83	51.78	37.53	2.25	38.23	58.63	60.11	72.80	57.14	57.38
QuIP	2.25	23.72	28.96	38.65	53.86	50.83	39.20	2.25	39.51	60.44	68.95	73.61	65.35	61.57
EfficientQAT	2.25	43.77	67.55	68.65	74.65	64.33	63.79	2.25	54.86	77.27	79.01	80.36	65.67	71.43
GSQ (ours)	2.13	**48.12**	**72.35**	**73.42**	**78.07**	**70.80**	**68.55**	2.13	**58.87**	**79.55**	**82.11**	**81.07**	**76.24**	**75.57**
GSQ (ours)	1.71	**42.83**	**67.13**	**67.91**	**73.50**	**65.82**	**63.44**	–	–	–	–	–	–	–

codebase with the hyperparameters recommended by the original authors. For GPTQ and QuIP, we use 512 calibration samples as is standard, and for EfficientQAT we follow the authors' suggested setup.

Vector quantization baselines. We compare against QTIP [65] and PV-Tuning [54], which optimizes over an AQLM vector quantized representation, as these are two state-of-the-art methods in the low-bit regime. Since VQ methods are not restricted to a small scalar grid, they are generally more expressive than scalar quantizers at the same bit-width; we therefore view the comparison to VQ as a particularly hard test for GSQ.

4.2 Llama-3.1 results

Table 1 reports zero-shot accuracy on Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct. Across both model sizes, GSQ is the strongest scalar quantization method in our comparison and remains competitive with recent VQ methods.

At 2 bits, GSQ substantially outperforms GPTQ, QuIP, and EfficientQAT despite using symmetric quantization without zero-points. On Llama-3.1-70B, it improves over the best scalar baseline by 4.14 average points and narrows the gap to VQ methods to 1.68 points vs. QTIP and 0.70 points vs. PV-Tuning. At 3 bits, GSQ again gives the best scalar results and approaches the VQ frontier, showing that the benefit is not limited to the most extreme 2-bit setting. Finally, ternary GSQ on Llama-3.1-8B is competitive with, and often exceeds, scalar baselines evaluated at higher 2-bit precision.

Non-uniform results. Because GSQ produces standard scalar quantized layers, it naturally supports *non-uniform* bit allocation, in which different layers are assigned different bit-widths (e.g., some layers at 3 bits and others at 2 bits) to achieve a target average rate. This is motivated by the observation that not all layers are equally sensitive to quantization: allocating more bits to sensitive layers and fewer to robust ones can yield a better accuracy-compression trade-off than using a single uniform bit-width everywhere. Per-layer bit assignments are obtained via non-uniform search [34]. Table 1 includes non-uniform GSQ results on Llama-3.1-70B-Instruct at 2.62 and 2.37 average bits per parameter. These configurations interpolate between the uniform 3-bit and 2-bit operating points. Llama-3.1-8B-Instruct is also evaluated at 2.37 average bits per parameter.

Speedup. A key advantage of scalar quantization over vector- or trellis-based methods is that it can directly leverage highly optimized low-precision GEMM kernels, translating memory savings into proportional throughput gains. Table 2 reports end-to-end throughput for GSQ-quantized Llama-3.1-70B-Instruct using vLLM [43] and Humming kernels [40] on

Table 2: End-to-end inference speedup for GSQ-quantized Llama-3.1-70B-Instruct on NVIDIA L40s GPUs (vLLM + Humming kernels). TPS/GPU = output tokens per second per GPU.

Method	**Avg bit/param**	**ShareGPT TPS/GPU**	**Speedup**
BF16 (4 GPU)	16.00	60.3	1.00×
Uniform 3-bit	3.13	289.6	4.80×
Non-uniform 2.62	2.62	301.1	4.99×
Non-uniform 2.37	2.37	329.2	5.46×
Uniform 2-bit	2.13	374.0	6.20×

Table 3: Results on Kimi-K2.5. We compare the original model and its 2-bit GSQ version on mathematical reasoning, scientific QA, coding benchmarks, and long-context OpenAI-MRCR across different context lengths. OpenAI-MRCR uses 3 repeats and 4 needles.

Method	bpp	Reasoning and Coding				OpenAI-MRCR					
		AIME 25 ($n = 10$)	GPQA Diamond ($n = 5$)	LiveCodeBench ($n = 10$)	MATH 500 ($n = 5$)	0–8k	8k–16k	16k–32k	32k–64k	64k–128k	128k–256k
Base	4.5	**95.33**	**89.29**	61.37	96.68	95.37	77.81	69.19	**57.10**	**59.50**	**46.04**
GSQ (Ours)	2.13	93.00	76.57	**69.37**	**97.32**	**97.81**	**85.75**	**74.41**	53.73	59.03	44.91

NVIDIA L40s GPUs. We report output tokens per second per GPU (TPS/GPU) to normalize across tensor-parallelism configurations, using a representative ShareGPT (short, conversational) workload.[3] Uniform 2-bit quantization achieves up to 6.2× speedup over BF16, while the non-uniform configurations at 2.37 and 2.62 average bits provide 4.99–5.46× speedup, demonstrating that non-uniform bit allocation offers a practical accuracy–throughput trade-off within a single, kernel-compatible scalar format.

4.3 Kimi-K2.5 results

We further evaluate GSQ on Kimi-K2.5, a 1T-parameter MoE model natively stored in 4-bit format. We quantize only the non-shared expert weights to 2 bits, leaving shared experts and vision components unchanged, and compare against the original model under the same evaluation pipeline.

Table 3 shows that 2-bit GSQ largely preserves Kimi-K2.5's reasoning and coding performance. It remains close on AIME25, improves on LiveCodeBench-v6 and MATH500, but drops on GPQA Diamond. We attribute this pattern partly to calibration data: OpenThoughts is skewed toward mathematics and code, whereas GPQA Diamond emphasizes science-heavy question answering. To further examine this hypothesis, Appendix D reports an additional Kimi-K2.6 experiment using a more diverse calibration mixture. Although this is not a controlled ablation since the base model also changes, the GPQA gap is substantially smaller, indicating the importance of calibration diversity.

Table 3 additionally reports OpenAI-MRCR results across context lengths up to 256k. GSQ improves over the base model up to 32k tokens and remains close at longer ranges, with modest degradation beyond 32k. This suggests that aggressive 2-bit quantization preserves much of the model's long-context capability, although the longest contexts are more sensitive.

Our LiveCodeBench score for the original Kimi-K2.5 model is lower than the model-card number, likely due to differences in evaluation protocol. Since the exact protocol is not publicly specified, we report both base and quantized results under our common pipeline.

Lastly, we evaluate Kimi-K2.5 on agentic benchmarks. Results can be found in Appendix K.

4.4 Extension to GGUF K-Quant formats

We finally evaluate whether GSQ can improve already-available GGUF K-Quant checkpoints. This setting is practically important because GGUF quantization formats are widely used for local and edge deployment, and many high-quality quantized checkpoints are released publicly. In this setting, GSQ treats the GGUF checkpoint as the initialization point, improves its discrete assignments, and projects the result back into the same format. Thus, the optimized model remains compatible with standard GGUF inference stacks. Additional details on the K-Quant parameterization and optimization procedure are provided in Appendix E.

Setup. We use publicly available Unsloth GGUF checkpoints as initialization and compare them against the corresponding GSQ-optimized checkpoints. We evaluate a reasoning-oriented Qwen3-8B model under two common K-Quant formats: Q3_K_M

Table 4: GGUF K-Quant results on Qwen3-8B.

Method	AIME25		GPQA Diamond		MMLU-Pro		Avg.
	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	
BF16	64.67	0.52M	58.38	1.48M	72.42	32.07M	65.16
Unsloth-Q3_K_M	56.67	0.58M	54.65	1.67M	70.23	35.11M	60.52
GSQ-Q3_K_M	**59.00**	**0.55M**	**55.15**	**1.53M**	**70.67**	**31.57M**	**61.61**
Unsloth-Q2_K	38.67	0.64M	46.46	2.46M	64.97	48.78M	50.03
GSQ-Q2_K	**52.00**	**0.58M**	**50.91**	**1.80M**	**65.94**	**39.69M**	**56.28**

[3] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

and Q2_K. The evaluation includes AIME25, GPQA Diamond, and MMLU-Pro. For AIME25 and GPQA Diamond, we report the average `pass@1` score over 10 and 5 repeated evaluation runs, respectively. In addition to task accuracy, we report the average number of generated tokens, including both visible text tokens and reasoning tokens, since low-bit quantization is known affect not only correctness but also the efficiency of the model's reasoning process; specifically, quantized models tend to produce more tokens for the final answer.

Table 4 shows that GSQ consistently improves over the Unsloth [1] initialization. Gains are largest in the more aggressive Q2_K format: the average score improves from 50.03 to 56.28 on Qwen3-8B. The Q3_K_M setting shows smaller but consistent gains. GSQ also often reduces total generated tokens, suggesting that better discrete assignments can improve not only accuracy but also reasoning efficiency. Overall, these results show that GSQ extends beyond standard scalar quantization and can improve practical GGUF K-Quant checkpoints without changing their deployment format. We provide additional GGUF K-Quant results on Qwen3-4B in Appendix J, where GSQ shows similar gains over the corresponding Unsloth checkpoints.

5 Conclusion

We presented a sampling-based quantization method suggesting that the apparent divide between "simple" scalar quantization and more expressive vector- or trellis-based methods is smaller than previously believed. In the low-bit regime, much of the gap appears to be an *optimization* gap rather than a fundamental limitation of deployment-friendly quantization formats. By turning per-weight grid assignment into a differentiable discrete optimization problem, GSQ substantially improves the accuracy of standard symmetric group-wise scalar quantization at 2 and 3 bits, while preserving full compatibility with existing scalar inference kernels and deployment stacks. We further showed that the same idea extends to GGUF K-Quant checkpoints, improving public local-deployment models.

Although the area of weight quantization is by now very well studied, our work presents a new combination of accuracy, simplicity, and kernel compatibility. On dense Llama models, GSQ consistently outperforms prior scalar baselines and closes most of the gap to state-of-the-art VQ methods; at ternary precision, it is already competitive with or better than several stronger-bit scalar alternatives. At the same time, because GSQ does not rely on learned codebooks or specialized decoding schemes, it remains practical at the scale of modern MoE models, where more expressive quantizers are difficult to train and deploy.

More broadly, these results indicate that there is still substantial headroom in hardware-friendly scalar quantization, provided that the discrete optimization problem is treated seriously. An important direction for future work is to extend this idea beyond weight-only PTQ, for example to activation and KV-cache quantization, richer blockwise or task-aware objectives, and more efficient relaxations for even lower-bit or jointly quantized settings.

Acknowledgements

The authors would like to thank Verda Cloud for computational support, and in particular Paul Chang for his consistent, prompt and generous help throughout the project. We acknowledge the use of Humming kernels developed by Jinzhen Lin and the Venus Team, Ant Group. The ISTA team was supported in part by the FWF Bilateral AI Center of Excellence, as well as a generous grant from the NVIDIA corporation. This work was also supported under project ID 40 as part of the Swiss AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure.

References

[1] Unsloth AI, Daniel Han-Chen, and Michael Han-Chen. Qwen3-8b-gguf. `https://huggingface.co/unsloth/Qwen3-8B-GGUF`, 2025.

[2] Unsloth AI, Daniel Han-Chen, and Michael Han-Chen. Kimi-k2.5-gguf. `https://huggingface.co/unsloth/Kimi-K2.5`, 2026.

[3] 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. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[6] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

[7] Hong Chen, Chengtao Lv, Liang Ding, Haotong Qin, Xiabin Zhou, Yifu Ding, Xuebo Liu, Min Zhang, Jinyang Guo, Xianglong Liu, et al. Db-llm: Accurate dual-binarization for efficient llms. *arXiv preprint arXiv:2402.11960*, 2024.

[8] 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)*, pages 10081–10100, 2025.

[9] Tianqi Chen, Zhe Li, Weixiang Xu, Zeyu Zhu, Dong Li, Lu Tian, Emad Barsoum, Peisong Wang, and Jian Cheng. Ternaryllm: Ternarized large language model. *arXiv preprint arXiv:2406.07177*, 2024.

[10] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023. URL `https://arxiv.org/abs/2302.06675`.

[11] Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12942–12963, 2025.

[12] Krishna Teja Chitty-Venkata, Jie Ye, and Murali Emani. Mopeq: Mixture of mixed precision quantized experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4023–4032, 2025.

[13] 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.

[14] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

[15] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987*, 2021.

[16] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

[17] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[18] 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. *arXiv preprint arXiv:2306.03078*, 2023.

[19] Peijie Dong, Lujun Li, Yuedong Zhong, Dayou Du, Ruibo Fan, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Yike Guo, et al. Stbllm: Breaking the 1-bit barrier with structured binary llms. *arXiv preprint arXiv:2408.01803*, 2024.

[20] Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

[21] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[22] Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. *Advances in Neural Information Processing Systems*, 37:7736–7758, 2024.

[23] Yi-Zeng Fang and Juinn-Dar Huang. Router choice matters: Rank-aware post-training quantization for moe models.

[24] Christopher Fifty, Ronald G. Junkins, Dennis Duan, Aniketh Iyengar, Jerry W. Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. In *International Conference on Learning Representations*, 2025.

[25] Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023.

[26] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[27] Zhongqian Fu, Ning Ding, Kai Han, Xianzhi Yu, Xiaosong Li, Xinghao Chen, Yehui Tang, and Yunhe Wang. Eaquant: Enhancing post-training quantization for moe models via expert-aware optimization. *arXiv preprint arXiv:2506.13329*, 2025.

[28] 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, 12 2023. URL `https://zenodo.org/records/10256836`.

[29] Georgi Gerganov and contributors. llama.cpp: Inference of LLaMA models in pure C/C++. `https://github.com/ggerganov/llama.cpp`, 2023.

[30] 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.

[31] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL `https://arxiv.org/abs/2506.04178`.

[32] Yufei Guo, Zecheng Hao, Jiahang Shao, Jie Zhou, Xiaode Liu, Xin Tong, Yuhan Zhang, Yuanpei Chen, Weihang Peng, and Zhe Ma. Pt-bitnet: Scaling up the 1-bit large language model with post-training quantization. *Neural Networks*, page 107855, 2025.

[33] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[34] Michael Helcig and Dan Alistarh. Model compression with exact budget constraints via Riemannian manifolds. *arXiv preprint arXiv:2605.00649*, 2026. URL `https://arxiv.org/abs/2605.00649`.

[35] Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. Channel selection using gumbel softmax. In *European conference on computer vision*, pages 241–257. Springer, 2020.

[36] Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. *arXiv preprint arXiv:2505.03804*, 2025.

[37] Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models. *arXiv preprint arXiv:2509.23809*, 2025.

[38] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*, 2024.

[39] Hugging Face. The Hugging Face model hub. https://huggingface.co/models, 2024.

[40] InclusionAI. Humming: An open-source toolkit for efficient LLM inference with mixed-precision quantization, 2025. URL https://github.com/inclusionAI/humming. Open-source library for vLLM-integrated weight-only quantization kernels supporting integer bitwidths 4–8.

[41] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[42] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[43] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[44] Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[45] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364, 2024.

[46] Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Linghe Kong, Yulun Zhang, Xiaokang Yang, et al. Arb-llm: Alternating refined binarizations for large language models. *arXiv preprint arXiv:2410.03129*, 2024.

[47] 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 *The twelfth international conference on learning representations*, 2023.

[48] 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. *Proceedings of machine learning and systems*, 6:87–100, 2024.

[49] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025. URL https://arxiv.org/abs/2308.03688.

[50] 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*, 2024.

[51] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations*, 2019.

[52] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

[53] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[54] 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. *Advances in Neural Information Processing Systems*, 37:5074–5121, 2024.

[55] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, volume 119, pages 7197–7206, 2020.

[56] Ollama contributors. Ollama: Get up and running with large language models locally. https://github.com/ollama/ollama, 2023.

[57] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[58] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First conference on language modeling*, 2024.

[59] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[60] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. Pb-llm: Partially binarized large language models. *arXiv preprint arXiv:2310.00034*, 2023.

[61] Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

[62] Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[63] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[64] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

[65] Albert Tseng, Qingyao Sun, David Hou, and Christopher M De Sa. Qtip: Quantization with trellises and incoherence processing. *Advances in Neural Information Processing Systems*, 37: 59597–59620, 2024.

[66] Turboderp. exllamav3: An optimized quantization and inference library for local LLMs. `https://github.com/turboderp-org/exllamav3`, 2025.

[67] Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cédric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. GPTVQ: The blessing of dimensionality for LLM quantization. In *International Conference on Machine Learning (ICML)*, 2024.

[68] Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, et al. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*, 2024.

[69] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

[70] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

[71] He Xiao, Runming Yang, Qingyao Yang, Wendong Xu, Zhen Li, Yupeng Su, Zhengwu Liu, Hongxia Yang, and Ngai Wong. Ptqtp: Post-training quantization to trit-planes for large language models. *arXiv preprint arXiv:2509.16989*, 2025.

[72] Xianglong Yan, Chengzhu Bao, Zhiteng Li, Tianao Zhang, Kaicheng Yang, Haotong Qin, Ruobing Xie, Xingwu Sun, and Yulun Zhang. Pt ^{2}-llm: Post-training ternarization for large language models. *arXiv preprint arXiv:2510.03267*, 2025.

[73] 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*, pages 4791–4800, 2019.

[74] Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025. URL `https://arxiv.org/abs/2510.04206`.

[75] Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

[76] Jiaqi Zhao, Miao Zhang, Ming Wang, Yuzhang Shang, Kaihao Zhang, Weili Guan, Yaowei Wang, and Min Zhang. Ptq1. 61: Push the real limit of extremely low-bit post-training quantization methods for large language models. *arXiv preprint arXiv:2502.13179*, 2025.

[77] Zihao Zheng, Xiuping Cui, Size Zheng, Maoliang Li, Jiayu Chen, Yun Liang, and Xiang Chen. Moqa: Rethinking moe quantization with multi-stage data-model distribution awareness. *arXiv preprint arXiv:2503.21135*, 2025.

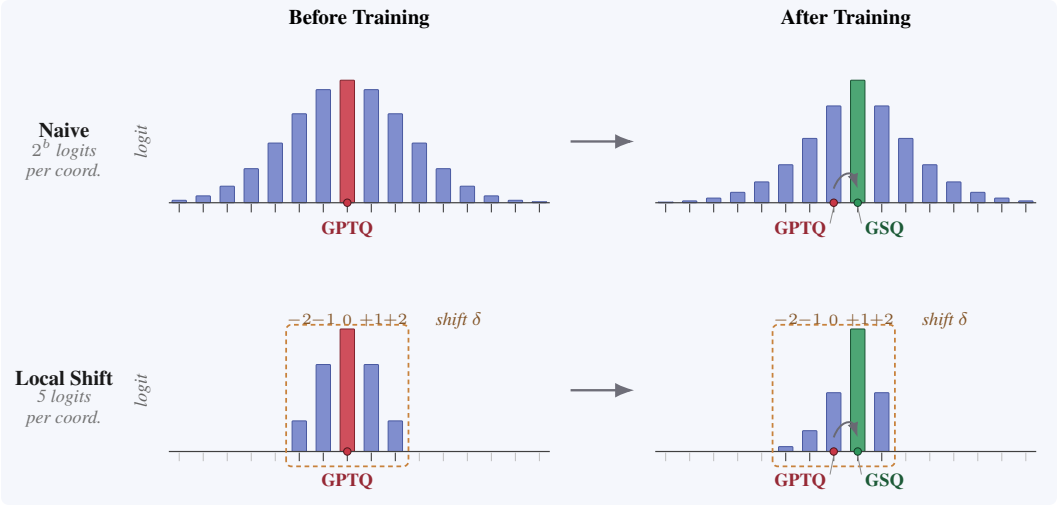


Figure 1: **Local-shift parameterization at higher bit-widths.** Each row shows, for a single weight coordinate, the logit distribution over candidate grid points before and after training. The red bar and dot mark the GPTQ-initialized grid point used to warm-start the logits; the green bar and dot mark the grid point selected by GSQ after training. *Top (naive):* placing one trainable logit on every grid point costs 2^b logits per coordinate, which quickly becomes prohibitive as b grows. *Bottom (local shift):* we instead assign logits only to a discrete shift $\delta_i \in \{-2, -1, 0, +1, +2\}$ relative to the GPTQ-initialized grid index (dashed window), reducing the per-coordinate parameter count from 2^b to 5. In both cases the distribution is initialized as a Gaussian centered at the GPTQ solution.

A Initialization details

In the main text, we state that for the 2-bit case and higher bitwidths we initialize the logits using a Gaussian-like prior around the GPTQ solution. Here we give the details.

For each coordinate i and candidate k, we set

$$(\boldsymbol{\ell}_{\text{GPTQ}})_{i,k} \propto -\frac{(c_k - \mu_i)^2}{2}, \qquad \mu_i = \begin{cases} (\mathbf{q}_{\text{GPTQ}})_i, & b = 2, \\ 0, & b > 2. \end{cases}$$

Equivalently, candidates closer to μ_i receive larger initial logits. Here c_k denotes the k-th candidate value. For $b = 2$, the candidates are the quantization grid points,

$$c_k \in \mathcal{G}_2,$$

so centering at $\mu_i = (\mathbf{q}_{\text{GPTQ}})_i$ directly favors the GPTQ-selected value. For $b > 2$, the candidates are discrete index shifts,

$$c_k \in \{-2, -1, 0, 1, 2\}.$$

In this case, the GPTQ solution is already encoded by the initial grid index j_i^0, and the candidate $c_k = 0$ corresponds to keeping the GPTQ-assigned grid point. Thus centering at $\mu_i = 0$ favors staying at the GPTQ solution while still assigning nonzero probability to nearby shifts.

Finally, for each coordinate, we subtract the mean logit and inject noise as in Equation 4 for the ternary case. This yields an initialization concentrated near the GPTQ solution while still allowing exploration of alternative candidates.

B Vanishing gradients in saturated Gumbel-Softmax relaxations

We explain why gradients through a low-temperature softmax relaxation can become exponentially small when the logits are well separated. Let $a \in \mathbb{R}^K$ denote the logits and let $y = \mathrm{softmax}(a/\tau)$, where $\tau > 0$ is the temperature. Suppose that a has a unique maximizer $i^\star = \arg\max_i a_i$, and define the logit gap $\Delta := a_{i^\star} - \max_{j \neq i^\star} a_j > 0$. As τ decreases, or as Δ increases, the relaxed categorical vector y becomes increasingly close to the one-hot vector $e_{i^\star}$. This saturation causes the Jacobian of the softmax to shrink.

Lemma B.1. *Let $a \in \mathbb{R}^K$ have a unique maximizer $i^\star = \arg\max_i a_i$, and define $\Delta := a_{i^\star} - \max_{j \neq i^\star} a_j > 0$. For $\tau > 0$, let $y = \mathrm{softmax}(a/\tau)$. Suppose ℓ is differentiable at y and $\|\nabla_y \ell(y)\|_1 \leq G$. Then*

$$\|\nabla_a \ell(y)\|_\infty \leq \frac{G}{\tau}\big(1 - y_{i^\star}\big) \leq \frac{(K-1)G}{\tau} e^{-\Delta/\tau}.$$

Consequently, for fixed $\Delta > 0$, $\|\nabla_a \ell(y)\|_\infty \to 0$ as $\tau \to 0$, and for fixed $\tau > 0$, $\|\nabla_a \ell(y)\|_\infty \to 0$ as $\Delta \to \infty$.

Proof. Let $g = \nabla_y \ell(y)$. The softmax Jacobian satisfies $\partial y_k / \partial a_i = \tau^{-1} y_k (\mathbf{1}_{k=i} - y_i)$, so

$$\frac{\partial \ell}{\partial a_i} = \frac{1}{\tau} \sum_{k=1}^{K} g_k y_k (\mathbf{1}_{k=i} - y_i).$$

Taking absolute values gives

$$\left| \frac{\partial \ell}{\partial a_i} \right| \leq \frac{\|g\|_1}{\tau} \max_k y_k |\mathbf{1}_{k=i} - y_i|.$$

We now bound the final factor. If $i = i^\star$, then $y_{i^\star}(1 - y_{i^\star}) \leq 1 - y_{i^\star}$. If $i \neq i^\star$, then $y_i(1 - y_i) \leq y_i \leq 1 - y_{i^\star}$, and for $k \neq i$, $y_k y_i \leq y_i \leq 1 - y_{i^\star}$. Thus, for every i,

$$\left| \frac{\partial \ell}{\partial a_i} \right| \leq \frac{G}{\tau}(1 - y_{i^\star}),$$

which implies $\|\nabla_a \ell(y)\|_\infty \leq \frac{G}{\tau}(1 - y_{i^\star})$. Finally,

$$1 - y_{i^\star} = \frac{\sum_{j \neq i^\star} \exp((a_j - a_{i^\star})/\tau)}{1 + \sum_{j \neq i^\star} \exp((a_j - a_{i^\star})/\tau)} \leq \sum_{j \neq i^\star} \exp((a_j - a_{i^\star})/\tau) \leq (K-1) e^{-\Delta/\tau}.$$

The limiting statements follow immediately. □

The lemma shows that the gradient with respect to the logits is controlled by the probability mass assigned to non-maximal categories. Once the relaxation is nearly one-hot, this tail mass is exponentially small in Δ/τ. Thus, even if the downstream loss has a nonzero gradient with respect to the relaxed variable y, the gradient propagated back to the logits can become negligible.

This effect is relevant for optimizer choice. AdamW rescales gradients using an estimate of their second moment and includes an additive numerical constant ϵ in the denominator. In the saturated regime, the raw gradients passed to the optimizer may be extremely small, so the effective update can become highly sensitive to ϵ and may shrink enough to stall progress. Increasing ϵ can partially counteract this behavior, but it introduces an additional tuning choice and may be problem dependent.

Lion instead uses a sign-based update computed from momentum. As a result, its update direction is less directly tied to the absolute magnitude of the instantaneous gradient. This does not remove the saturation phenomenon itself: if the gradient is exactly zero, no first-order optimizer receives useful local information. However, when gradients are nonzero but exponentially small, Lion is less sensitive to their magnitude collapse than AdamW, which makes it a natural choice for training with saturated Gumbel-Softmax relaxations.

Ablation: AdamW vs. Lion We have theoretically shown that AdamW is not well suited to our setting, as it can effectively stall due to the vanishing gradient magnitudes of the logit parameters, i.e., the parameters associated with discrete assignments. In this subsection, we empirically validate this observation.

Tuning the learning rate for AdamW in this setting is particularly challenging. With a small learning rate, similar to the one used for Lion, the logit parameters initially make progress while the temperature is still sufficiently high. However, after a few epochs, as the temperature decreases and the gradient magnitudes shrink, the logits tend to stall and fail to converge to a clear discrete choice. As a result, the final assignment can become effectively random, since the probabilities of different choices remain close to one another. To mitigate this issue, AdamW requires a much larger learning rate for the logits so that they can converge early in training, before the gradients become too small. This, however, makes the optimization less stable and harder to tune.

Table 5 compares AdamW and Lion on Llama-3.1-8B-Instruct under 2-bit weight-only quantization. Lion consistently outperforms AdamW across all evaluated benchmarks, improving the average accuracy from 61.84 to 65.54, a gain of 3.70 points. The largest improvements are observed on WinoGrande and HellaSwag, where Lion improves over AdamW by 4.73 and 4.55 points, respectively. These results support our theoretical analysis: Lion provides a more reliable optimization behavior for the logit parameters and leads to better discrete assignment decisions.

Table 5: Ablation of the optimizer choice on Llama-3.1-8B-Instruct at 2-bit weight-only quantization.

Optimizer	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
AdamW	40.61	69.49	62.15	73.50	63.46	61.84
Lion	**44.20**	**72.18**	**66.70**	**76.44**	**68.19**	**65.54**

C Details of within-block staging

For Llama and Qwen models, we optimize the quantized layers within each Transformer block using a staged procedure rather than optimizing all layers jointly under a single block reconstruction loss. Joint optimization is the most direct formulation and has been used in prior work [20, 8]. However, we found it to be suboptimal in our setting.

One reason is that block reconstruction is only a surrogate for final quantized-model quality. The loss measures the discrepancy between the output of a floating-point block and the output of its quantized counterpart, but this signal is not equally informative for all layers in the block. For layers that appear early in the computation graph, such as the query, key, and value projections, their effect on the block output is mediated by subsequent attention operations, the residual pathway, and the MLP. Consequently, the block-level loss provides a relatively indirect optimization signal for these layers. Later layers, such as the MLP projections, are more directly exposed to the block output loss. This imbalance can make fully joint optimization less effective than optimizing smaller groups of layers under reconstruction losses that are closer to their immediate outputs. This intuition is consistent with the staged strategies used by Tseng et al. [64, 65], which also partition the block and optimize one group at a time rather than all layers jointly.

Our staging strategy therefore partitions the block according to the structure of the Transformer computation. Ideally, the query and key projections would be optimized jointly, since the relevant quantity for attention is their interaction through the attention logits rather than the reconstruction of either projection in isolation. In larger models, however, this joint optimization is expensive. As a cheaper approximation, we first optimize the query and key projections independently, each under its own linear reconstruction loss. We then freeze these projections and optimize the value and output projections jointly under a self-attention output reconstruction loss. Finally, after freezing the attention projections, we optimize the MLP projections under the full block reconstruction loss.

D Discussion on the effect of calibration data

The choice of calibration data can substantially affect quantization quality, particularly for Mixture-of-Experts (MoE) models, where sparse routing may leave some experts insufficiently represented during calibration. Moreover, calibration sets concentrated on specific domains may not adequately preserve performance on tasks requiring broader knowledge. We investigate this effect through additional experiments on Kimi-K2.6 and a controlled calibration-data ablation on Qwen3.6-35B-A3B.

Calibration data for Kimi-K2.6. The Kimi-K2.5 results in Table 3 show a noticeably larger drop on GPQA Diamond than on the mathematics and coding benchmarks. One plausible explanation is the calibration data: Kimi-K2.5 was calibrated only on OpenThoughts [31], which is concentrated on mathematics and code. This may help preserve performance on MATH500 and LiveCodeBench while providing weaker coverage of science-heavy domains such as GPQA Diamond.

To examine this effect further, we evaluate GSQ on Kimi-K2.6 using a more diverse calibration mixture consisting of OpenThoughts, OpenPlatypus [44], and FineWeb-Edu [52]. The goal is to retain mathematical reasoning coverage while incorporating broader knowledge and science-oriented text. We observed evaluation instabilities on some tasks that we could not conclusively attribute to

Table 6: Additional Kimi-K2.6 results using a diverse calibration mixture.

Method	bit/param	AIME25	GPQA Diamond	MATH500
Base	4.5	**90.00**	**87.21**	92.67
GSQ (Ours)	2.13	87.78	83.16	**93.67**

Table 7: Effect of calibration data on 2-bit GSQ quantization of Qwen3.6-35B-A3B. The mixed calibration set combines OpenThoughts, OpenPlatypus, and FineWeb-Edu. Average denotes the mean accuracy across the three benchmarks.

Calibration	AIME25	GPQA Diamond	MMLU-Pro	Avg.
Full precision	93.33	83.84	84.87	87.35
OpenThoughts	**86.67**	70.71	79.76	79.05
Mixed	**86.67**	**76.77**	**81.37**	**81.60**

either the checkpoint or the vLLM execution stack. We therefore report only the tasks for which evaluation was stable: AIME25, GPQA Diamond, and MATH500.

As shown in Table 6, the quantized Kimi-K2.6 model remains close to the base model on AIME25 and GPQA Diamond, with drops of 2.22 and 4.05 points, respectively, while slightly improving MATH500 from 92.67 to 93.67. Compared with the Kimi-K2.5 results, the GPQA Diamond degradation is substantially smaller. However, since both the base model and calibration data differ, this comparison does not isolate the contribution of calibration composition.

Calibration data ablation on Qwen3.6-35B-A3B. To isolate the effect of calibration data more directly, we perform a controlled experiment on Qwen3.6-35B-A3B at 2-bit precision. We compare two calibration configurations: (i) OpenThoughts alone, and (ii) a mixed dataset consisting of OpenThoughts, OpenPlatypus, and FineWeb-Edu. We evaluate both quantized models on AIME25, GPQA Diamond, and MMLU-Pro, and additionally report the full-precision model as a reference.

Table 7 shows that calibration composition has a meaningful impact on downstream accuracy. The mixed calibration set improves average performance by 2.55 percentage points over OpenThoughts alone. The largest gain occurs on GPQA Diamond, where accuracy increases from 70.71 to 76.77 (+6.06 points), followed by a 1.61-point improvement on MMLU-Pro. Meanwhile, AIME25 accuracy remains unchanged at 86.67, indicating that the additional calibration diversity improves performance on broader knowledge-intensive tasks without sacrificing mathematical reasoning accuracy in this experiment.

These findings complement the Kimi-K2.6 results and provide direct evidence that calibration-data composition can materially affect low-bit quantization performance. Importantly, calibration selection is orthogonal to the optimization mechanism of GSQ: more representative calibration sets, including expert-balanced or self-generated data, can be incorporated without modifying the underlying quantization algorithm. Developing such calibration strategies remains a promising direction for further improving extremely low-bit MoE quantization.

E Details of GGUF K-Quant extension

The key idea is to treat an existing GGUF checkpoint as the initialization point for discrete optimization. Given a pretrained K-Quant model, each quantized weight is first associated with its current GGUF code or grid index. When the corresponding discrete choice has at most four possible values, as in ternary or 2-bit quantization, we optimize over all possible outcomes directly using the full Gumbel-Softmax relaxation. For larger discrete sets, we instead use the local-shift formulation introduced earlier: rather than assigning logits to all possible code values, we introduce a small discrete shift around the initialized GGUF index. During optimization, Gumbel-Softmax sampling selects either a relaxed code value or a relaxed local shift, depending on the size of the discrete set, while the final model is projected back into the same GGUF K-Quant format. Thus, GSQ does not require changing the deployment representation: the output remains a standard K-Quant model and can be served using the same GGUF-compatible inference stack. In this sense, GSQ acts as a

Table 8: Training hyperparameters for the Llama experiments.

Bit-width	Logits lr	Group scales lr	Weight decay	Betas	Epochs	# Seqs.	Seq. len.	Batch size	Group size	τ schedule	κ schedule	α	std
1.58-bit	1e-4	5e-5	1.0	$(0.9, 0.95)$	20	4096	4096	64	128	linear: $2 \to 0.05$	linear: $100 \to 500$	3	0.01
2/3-bit	1e-4	5e-5	1.0	$(0.9, 0.95)$	20	4096	4096	64	128	linear: $2 \to 0.05$	linear: $100 \to 500$	6	0.01

Table 9: Training hyperparameters for Kimi experiments.

Bit-width	Logits lr	Group scales lr	Weight decay	Betas	Epochs	# Seqs.	Seq. len.	Batch size	Group size	τ schedule	κ schedule	α	std
2-bit	2e-4	1e-5	1.0	$(0.9, 0.95)$	10	4096	4096	64	128	linear: $2 \to 0.05$	linear: $100 \to 500$	6	0.01

post-processing optimizer for existing GGUF checkpoints, improving the discrete assignments while preserving the original storage and execution format.

F Limitations

While GSQ improves the accuracy of low-bit scalar quantization and preserves deployment-friendly formats, it has several limitations. First, the method introduces auxiliary logits during optimization, with memory overhead proportional to the number of candidate grid values considered per weight. Although this overhead is temporary and does not affect the final deployed model, it makes full-model end-to-end optimization impractical for large LLMs and motivates our block-wise, expert-wise, and local-shift training procedures.

Second, GSQ depends on a good initialization. In our experiments we warm-start from GPTQ or from an existing GGUF checkpoint, and the local-shift formulation assumes that most useful assignment changes remain near the initialized grid point. This assumption works well empirically, but it may restrict recovery when the initialization is poor or when globally different assignments are needed. More adaptive shift ranges or multi-stage re-initialization strategies may improve robustness.

Finally, our results do not imply that scalar quantization universally matches vector- or trellis-based methods. These methods remain more expressive at a fixed bit-width and may retain an advantage in regimes where codebook lookup or specialized kernels are acceptable. Rather, our results show that a carefully optimized scalar quantizer can close much of the gap while preserving a simpler and more portable deployment format.

G Additional experimental details

G.1 Full training hyperparameters

Tables 8, 9, and 10 summarize the training hyperparameters used in our block-wise optimization experiments.

For the dense Llama and Qwen models, we use 20 epochs of block-wise optimization. For Kimi models, we use only 10 epochs. Although Kimi models are much larger overall, the optimization problem is decomposed across 384 experts and solved independently for each expert. As a result, each individual optimization problem is substantially smaller than in the dense Llama models, which makes fewer epochs sufficient in practice. Additionally, based on the discussion in the previous subsection, for Qwen models we use a calibration mixture consisting of OpenThoughts, OpenPlatypus, and FineWeb-Edu.

There is one additional stabilization detail for Llama-3.1-70B. For this model only, we apply gradient clipping during training, and only to the logits of the discrete parameters, not to the group scales. The clipping threshold is set to 10^{-6} for the 2-bit setting and 10^{-8} for the 3-bit setting.

G.2 Evaluation

We evaluate all models with `lm-eval-harness` [28] in the zero-shot setting, using a maximum sequence length of 4096. For the Llama models, we report accuracy on ARC-Easy, ARC-Challenge, HellaSwag, WinoGrande, and PIQA [13, 73, 59, 5]. These are standard zero-shot reasoning and commonsense benchmarks widely used in the quantization literature, and together they cover multi-choice scientific reasoning, commonsense completion, and physical reasoning.

Table 10: Training hyperparameters for the Qwen experiments.

Bit-width	Logits lr	Group scales lr	Weight decay	Betas	Epochs	# Seqs.	Seq. len.	Batch size	Group size	τ schedule	κ schedule	α	std
Q2_K	1e-4	2e-5	1.0	$(0.9, 0.95)$	20	4096	4096	64	128	linear: $2 \rightarrow 0.05$	linear: $100 \rightarrow 500$	6	0.01
Q3/4/5_K	1e-4	0	1.0	$(0.9, 0.95)$	20	4096	4096	64	128	linear: $2 \rightarrow 0.05$	linear: $100 \rightarrow 500$	6	0.01

Model	Runtime
Llama-3.1-8B-Instruct	10 hours
Llama-3.1-70B-Instruct	68 hours
Kimi-K2.5	24 hours

Table 11: Total runtime of GSQ for different models on $8\times$H200 GPUs.

For Kimi-K2.5 model, we additionally focus on long-context and reasoning evaluations. More specifically, for long-context, we evaluate models on OpenAI-MRCR [68] benchmark across all sequence-length buckets from 0 to 256k, which is the model's maximum supported sequence length. For each sequence length bucket, we report average `pass@1` score over 5 repetitions. As for reasoning evaluations, we focus on AIME25 [75], GPQA:Diamond [58], MATH500 [47], and LiveCodeBench-v6 [41]. We report the average `pass@1` score: over 10 repetitions for AIME25 and LiveCodeBench-v6, and over 5 repetitions for GPQA:Diamond and MATH500. For both, long-context and reasoning evaluations, we follow Kimi's suggested sampling parameters: `temperature=1.0` and `top_p=0.95`.

G.3 Randomness and reproducibility

GSQ is stochastic due to Gumbel-Softmax sampling. However, repeated runs on early-layer block-wise reconstruction show low variance in the optimization loss, hence given the cost of 70B- and MoE-scale quantization, we report a single run per configuration. All experiments were conducted primarily on a node with $8\times$H200.

G.4 End-to-end compression runtime

Table 11 summarizes the end-to-end runtime of GSQ across different models using $8\times$H200 GPUs. As expected, the total runtime increases with model size. The 8B model completes in 10 hours, while Llama-3.1-70B-Instruct requires 68 hours. Interestingly, although Kimi-K2.5 is substantially larger than Llama-3.1-70B-Instruct, it completes in only 24 hours. This is mainly due to two factors. First, we do not quantize the attention layers for Kimi-K2.5, which decreases its computational cost. Second, we train Kimi-K2.5 for fewer epochs, which further reduces its total runtime. Overall, these results show that GSQ remains practical at different model scales, although larger models introduce a substantially higher runtime overhead.

H Effect of end-to-end scale-only fine-tuning

Table 12 isolates the contribution of the end-to-end scale-only fine-tuning stage for the 2-bit Llama models. In this stage, the discrete assignments found by block-wise optimization are kept fixed, and only the per-group scales are updated using the distillation objective described in Section 4.2. This ablation shows how much additional performance can be recovered by a lightweight global refinement after the block-wise discrete search.

The results in Table 12 show that the block-wise stage already captures the main benefit of the discrete optimization, while a single end-to-end pass that updates only the scales can provide an additional refinement without revisiting the discrete assignments.

I Connection to MaskLLM: a 2:4 sparsity comparison

Our method is directly inspired by MaskLLM [22]. Conceptually, GSQ extends the same discrete optimization viewpoint from structured sparsity to low-bit quantization, while also changing the optimization granularity from end-to-end training to block-wise training. Since the latter is substan-

Table 12: Effect of end-to-end scale-only fine-tuning on the 2-bit GSQ models.

Setting	Llama-3.1-8B-Instruct						Llama-3.1-70B-Instruct					
	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
GSQ, block-wise only	44.20	72.18	66.70	76.44	68.19	65.54	57.25	79.71	78.08	80.09	77.11	74.45
GSQ, + scale fine-tuning	48.12	72.35	73.42	78.07	70.80	68.55	58.87	79.55	82.11	81.07	76.24	75.57
Δ	+3.92	+0.17	+6.72	+1.63	+2.61	+3.01	+1.62	-0.16	+4.03	+0.98	-0.87	+1.12

Table 13: Comparison to MaskLLM on the original 2:4 structured sparsity task of Fang et al. [22], evaluated on Llama-2-7B.

Method	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
FP16/BF16	46.25	74.58	75.98	79.11	69.14	69.01
MaskLLM (end-to-end, mask-only)	37.97	64.98	68.27	76.22	65.19	62.53
GSQ (block-wise, mask + weights)	40.02	68.18	65.31	75.35	65.43	62.86

tially cheaper, it is natural to ask whether the block-wise formulation remains competitive even in the original setting for which MaskLLM was designed.

To answer this, we perform an additional experiment on Llama-2-7B in the original 2:4 structured sparsity setting of MaskLLM. This experiment isolates the optimization strategy from the representation format: rather than comparing sparsity to quantization, we compare end-to-end MaskLLM training to our block-wise formulation on the same sparsity task. Importantly, the two methods do not optimize exactly the same variables. MaskLLM learns only the binary sparsity mask while keeping the underlying dense weights fixed at their original pretrained values. In contrast, in our formulation we jointly optimize both the mask and the weight values. Therefore, this comparison should not be interpreted as a strictly matched ablation, but rather as evidence that the proposed optimization framework remains effective, and in practice stronger, even when applied back to the structured sparsity setting that originally motivated MaskLLM. As shown in Table 13, the block-wise variant yields a better average zero-shot score across the same five tasks used in the main text.

J Qwen3-4B GGUF results

Table 14 reports GGUF K-Quant results for Qwen3-4B. Although these results are included mainly for completeness, they show a consistent advantage for GSQ over the Unsloth baselines at both Q3_K_M and Q2_K quantization levels. The improvement is especially pronounced under the more aggressive Q2_K setting, where GSQ substantially recovers accuracy across all evaluated benchmarks while using comparable or fewer tokens on GPQA Diamond and MMLU-Pro.

K Kimi-K2.5 additional results

Since Kimi-K2.5 is a "native multimodal agentic model" according to their model card [63], we wanted to check whether our quantization method preserves its ability to perform agentic tasks. For this, we evaluate Kimi-K2.5, our 2-bit version, and a similarly sized and bpw-comparable Unsloth model on AgentBench [49]. We used the most current version as introduced in [74], which consists of 5 agentic tasks spanning Operating System (OS) tasks, Database (DB) tasks, Knowledge Graph (KG) tasks, Web Shopping (WS) tasks, and ALFWorld, a text adventure game (Text Game / House-Holding, HH). We evaluate all models with 3 runs on each benchmark and report mean. As shown in Table 15, we remain competitive compared to the baseline on database, operating system, and web shopping tasks, which are among the most widely adopted settings. For tasks involving knowledge graphs and textual games, we perform worse, although this could heavily depend on the calibration data. Additionally, we outperform Unsloth's model on every task while maintaining a comparable overall model size. However, we note that Unsloth's model uses non-uniform quantization and quantizes every layer, whereas our approach only quantizes the experts. Therefore, although the total model size is roughly equivalent, our method effectively operates at a lower average bitwidth within the expert layers.

Table 14: GGUF K-Quant results on Qwen3-4B. Tokens include text and reasoning tokens.

Method	AIME25		GPQA Diamond		MMLU-Pro		Avg.
	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	
BF16	62.00	0.48M	53.74	1.29M	67.69	30.02M	61.14
Unsloth-Q3_K_M	55.00	0.60M	47.98	**1.32M**	64.11	34.00M	55.70
GSQ-Q3_K_M	**55.67**	**0.51M**	**52.53**	**1.32M**	**65.28**	**31.69M**	**57.83**
Unsloth-Q2_K	23.00	**0.51M**	28.59	2.17M	53.43	45.97M	35.01
GSQ-Q2_K	**45.33**	0.60M	**40.10**	**1.93M**	**59.60**	**44.69M**	**48.34**

Table 15: Results for Kimi-K2.5 for average performance (↑) across agentic benchmarks, spanning: Operating System (OS) Tasks, Database (DB) Task, Knowledge Graph (KG) Tasks, Web Shopping (WS) Tasks and ALFWorld, an Text Adventure Game (Text Game / House-Holding, HH). Reported values show mean score of 3 runs.

Method	OS	DB	KG	AlfWorld	WS
Base	50	69	57	53	52
Unsloth_UD-IQ1_M (2.4bit)	40	61	39	41	47
GSQ (2.37bit)	**43**	**63**	**45**	**42**	**49**

Limitations. Furthermore, we want to highlight that many issues occurred while evaluating GGUF models, making it infeasible to run more challenging evaluations with them. Templating often failed, and evaluating Kimi-K2.5 became extremely slow and impractical for evaluation purposes. Therefore, we focused on the most practical evaluations and skipped the others.

L Ablations

L.1 Naive block optimization

In the main experiments, we do not optimize all quantized tensors in a Transformer block jointly under a single block reconstruction loss. Instead, we use the staged within-block schedule described in Section 4.1.

To validate this choice, we compare against two simpler alternatives on Llama-3.1-8B-Instruct at 2-bit weight-only quantization. The first baseline, *Only-linear*, skips block-level optimization entirely and optimizes each linear layer only with its own layerwise reconstruction loss. The second baseline, *Block-all*, performs the most direct form of block optimization: all quantized tensors inside a Transformer block are optimized jointly using the block output reconstruction loss. Since these ablations are designed to isolate the effect of the block-wise optimization schedule, all numbers in this table are reported *before* the final end-to-end scale-only fine-tuning stage. For the staged GSQ result, we therefore use the "GSQ, block-wise only" setting from Table 12, rather than the final scale-tuned result used in the main comparison.

The results show that block-level information is useful, but that using it naively is insufficient. Moving from *Only-linear* to *Block-all* improves the average score from 57.68 to 62.02, confirming that reconstruction objectives beyond independent linear layers help reduce accumulated quantization error. However, *Block-all* is still 3.52 points below the staged GSQ schedule before any end-to-end scale-only fine-tuning. The gap is consistent across all tasks: staged GSQ improves over *Block-all* by 2.56 points on ARC-Challenge, 5.30 on ARC-Easy, 2.01 on HellaSwag, 3.97 on PIQA, and 3.79 on WinoGrande.

L.2 Robustness to initialization

We investigate the sensitivity of GSQ to its initialization through two controlled experiments on Llama-3.1-8B-Instruct at 2-bit precision. All results in this subsection are obtained after block-wise optimization, before the end-to-end scale-only fine-tuning stage.

Table 16: Ablation of the optimization objective and within-block schedule on Llama-3.1-8B-Instruct at 2-bit weight-only quantization.

Method	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
Only-linear	37.46	63.97	56.54	70.13	60.30	57.68
Block-all	41.64	66.88	64.69	72.47	64.40	62.02
GSQ, block-wise only	**44.20**	**72.18**	**66.70**	**76.44**	**68.19**	**65.54**

Table 17: Effect of initial quantization assignments on 2-bit GSQ for Llama-3.1-8B-Instruct, before end-to-end fine-tuning.

Initialization	ARC-C	ARC-E	Hella.	PIQA	Wino.	Avg.
Random	22.95	28.91	26.37	52.01	51.22	36.29
RTN	**44.71**	**72.18**	65.73	75.30	**68.67**	65.32
GPTQ	44.20	**72.18**	**66.70**	**76.44**	68.19	**65.54**

Initial quantization assignments. We first examine the effect of the initial quantization assignments by replacing GPTQ with round-to-nearest (RTN) and random Gaussian initialization, while retaining the same Gaussian-like logit prior centered on the selected assignments. Table 17 reports the downstream zero-shot accuracy for each initialization.

Random initialization substantially degrades performance, indicating that a meaningful initial assignment is important for effective optimization. In contrast, RTN and GPTQ achieve comparable downstream accuracy, with GPTQ providing a slightly higher average score (65.54 versus 65.32). These results suggest that GSQ is relatively robust to the choice of a structured initial assignment, rather than relying specifically on GPTQ.

Logit initialization. We next isolate the effect of the geometry-aware Gaussian-like logit prior described in Appendix A. Keeping the GPTQ assignments fixed, we compare our initialization against a simpler alternative that increases only the logit corresponding to the GPTQ-selected grid point, leaving all other logits unchanged. This alternative does not explicitly encode distances between candidate grid points.

Table 18 reports the hard validation reconstruction losses during optimization of the first transformer block.

The simple logit initialization quickly stagnates, whereas the Gaussian-like prior continues improving throughout optimization. By epoch 10, the geometry-aware initialization reduces the self-attention and full-block reconstruction losses by 54.5% and 60.4%, respectively, relative to the simple shift.

This improvement also translates into substantially higher downstream accuracy, as shown in Table 19.

Overall, these experiments distinguish two aspects of initialization. While GSQ is relatively insensitive to the choice between reasonable initial assignments such as RTN and GPTQ, the representation of these assignments in the relaxed optimization is considerably more important. Encoding the geometry of the scalar quantization grid through the Gaussian-like logit prior leads to substantially better reconstruction and downstream performance.

L.3 Validating the local-shift parameterization

For bit-widths above 2, GSQ uses a local-shift parameterization rather than optimizing over all grid values. In particular, for 3-bit quantization the full Gumbel-Softmax relaxation would assign one logit to each of the 8 scalar grid values, whereas our local-shift formulation only considers shifts in $\{-2, -1, 0, 1, 2\}$ around the initialized GPTQ assignment. This reduces the number of logits per weight from 8 to 5 while preserving the most relevant local moves.

To test whether this restriction excludes important assignments, we run an ablation on Llama-3.1-8B-Instruct in which the 3-bit quantizer is optimized using the *full* 8-way Gumbel-Softmax relaxation. After training, we compare the final hard grid index of each weight to its initialized GPTQ grid index and count the absolute index displacement. Table 20 reports the resulting distribution across several projection types. The results strongly support the local-shift design. Across the optimized

Table 18: Hard validation reconstruction losses for different logit initialization strategies on the first block of Llama-3.1-8B-Instruct at 2-bit precision.

Initialization	Target	E2	E4	E6	E8	E10
Gaussian-like	Attention	7.68e-6	2.67e-6	2.17e-6	1.99e-6	**1.94e-6**
Simple shift	Attention	4.36e-6	4.31e-6	4.28e-6	4.27e-6	4.26e-6
Gaussian-like	Full block	1.42e-5	1.11e-5	9.92e-6	9.42e-6	**9.27e-6**
Simple shift	Full block	2.41e-5	2.36e-5	2.35e-5	2.35e-5	2.34e-5

Table 19: Downstream zero-shot accuracy of different logit initialization strategies for 2-bit Llama-3.1-8B-Instruct, before end-to-end fine-tuning.

Initialization	ARC-C	ARC-E	Hella.	PIQA	Wino.
Simple shift	26.02	45.37	46.79	62.68	53.35
Gaussian-like	**44.20**	**72.18**	**66.70**	**76.44**	**68.19**

weights, 84.64% remain exactly at their initialized grid index and 15.36% move by only one grid position. Only 0.008% move by two positions, and 0.000004% move by three positions. Equivalently, 99.999996% of all observed full-relaxation assignments lie within the $\{-2, -1, 0, 1, 2\}$ neighborhood used by our local-shift formulation.

L.4 Gumbel–Softmax versus deterministic softmax and straight-through estimators

The key role of Gumbel–Softmax in GSQ is to learn a distribution over candidate grid assignments while allowing gradients to propagate through relaxed assignments. Unlike a straight-through estimator (STE), the forward pass uses soft values at finite temperatures rather than always selecting a discrete candidate. In the absence of perturbations, deterministic softmax may increasingly concentrate around early choices, limiting subsequent exploration of alternative assignments.

To examine the importance of stochasticity and relaxation separately, we repeat the paper's 2-bit experiment on the first transformer block of Llama-3.1-8B-Instruct, comparing (i) the proposed Gumbel–Softmax relaxation, (ii) deterministic softmax without noise, (iii) hard argmax with STE, and (iv) Gumbel perturbations followed by hard argmax with STE. All other experimental settings are unchanged. Table 21 presents the hard validation losses over ten epochs.

Gumbel–Softmax achieves the lowest final hard reconstruction error on both targets: 1.94×10^{-6} for self-attention and 9.27×10^{-6} for the full block. Gumbel+STE is the next-best alternative, reaching 2.33×10^{-6} and 1.03×10^{-5}, respectively. Both stochastic approaches improve steadily toward the end of training, whereas the deterministic alternatives make smaller late-stage gains, particularly on the full block. At low temperatures, nearly discrete assignments may leave scale refinement as the main source of further improvement. These results indicate that stochastic exploration of the categorical assignment space is beneficial, and that the differentiable relaxation provides additional gains over hard Gumbel+STE under the tested optimization schedule.

L.5 Noise ablation

Motivation. GSQ optimizes a categorical assignment over scalar quantization grid points. For assignment logits $\ell_1, \ldots, \ell_K$, the Gumbel–Max trick yields

$$\Pr\left[\arg\max_k \{\ell_k + g_k\} = i\right] = \frac{\exp(\ell_i)}{\sum_{k=1}^{K} \exp(\ell_k)}, \qquad g_k \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0, 1). \tag{7}$$

Thus, additive Gumbel perturbations have a principled connection to the categorical distribution defined by the logits. The temperature-controlled softmax relaxation of the perturbed logits retains differentiability and approaches a categorical sample as the temperature tends to zero [53, 42]. Gumbel noise is therefore a canonical choice for sampling from learned categorical probabilities; this observation does not establish that Gumbel noise is empirically superior to every other perturbation distribution.

Table 20: Ablation validating the local-shift parameterization.

Tensor	$\Delta = 0$	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$
`q_proj`	86.261%	13.662%	0.076%	0.00004%
`k_proj`	84.933%	14.957%	0.110%	0.00005%
`v_proj`	78.711%	21.289%	0.000%	0.00000%
`o_proj`	85.660%	14.340%	0.000%	0.00000%
`gate_proj`	82.671%	17.329%	0.000%	0.00000%
`up_proj`	83.925%	16.075%	0.000%	0.00000%
`down_proj`	86.954%	13.046%	0.000%	0.00000%
Total	84.635%	15.357%	0.008%	0.000004%

Table 21: Ablation of categorical assignment mechanisms on the first transformer block of Llama-3.1-8B-Instruct (2-bit). The only changed component is the assignment relaxation/estimator. Entries are hard validation reconstruction losses (lower is better).

Method / noise	**Target**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**
Gumbel-Softmax	Self-attention	7.76e-6	7.68e-6	4.13e-6	2.67e-6	2.35e-6	2.17e-6	2.06e-6	1.99e-6	1.96e-6	1.94e-6
Gumbel-Softmax	Full block	1.87e-5	1.42e-5	1.22e-5	1.11e-5	1.04e-5	9.92e-6	9.61e-6	9.42e-6	9.33e-6	9.27e-6
Deterministic softmax	Self-attention	7.17e-6	5.83e-6	5.14e-6	4.95e-6	4.62e-6	4.45e-6	4.40e-6	4.40e-6	4.14e-6	3.51e-6
Deterministic softmax	Full block	2.18e-5	1.82e-5	1.81e-5	1.83e-5	1.84e-5	1.83e-5	1.81e-5	1.78e-5	1.74e-5	1.52e-5
Argmax + STE	Self-attention	5.35e-6	4.56e-6	4.51e-6	4.37e-6	4.19e-6	4.08e-6	3.93e-6	3.66e-6	3.53e-6	3.40e-6
Argmax + STE	Full block	1.88e-5	1.78e-5	1.63e-5	1.65e-5	1.61e-5	1.55e-5	1.55e-5	1.51e-5	1.49e-5	1.42e-5
Gumbel + STE	Self-attention	8.24e-6	5.25e-6	3.90e-6	3.39e-6	3.01e-6	2.73e-6	2.55e-6	2.44e-6	2.37e-6	2.33e-6
Gumbel + STE	Full block	1.88e-5	1.44e-5	1.32e-5	1.24e-5	1.18e-5	1.13e-5	1.09e-5	1.06e-5	1.04e-5	1.03e-5

Choice of perturbation distribution. To isolate the role of the noise family, we repeat the 2-bit experiment on the first transformer block of Llama-3.1-8B-Instruct and replace the injected Gumbel noise with Gaussian, logistic, or uniform noise. We use four quantization candidates per coordinate. The perturbations are mean-centered and matched to the Gumbel(0,1) variance of $\pi^2/6$; initialization, optimizer, calibration data, and annealing schedules are fixed, without tuning separately for each family. Table 22 reports hard validation reconstruction losses throughout training.

At epoch 10, self-attention losses are between 2.74×10^{-6} and 2.77×10^{-6}, while full-block losses are between 1.07×10^{-5} and 1.08×10^{-5}. In the three replicate runs per family, the reported run-to-run half-range was at most 5×10^{-8}. Consequently, these experiments do not reveal a meaningful advantage for a particular variance-matched noise family. They support the use of stochastic exploration, but not a claim that the Gumbel distribution minimizes the reconstruction loss among all perturbations.

Sensitivity to noise magnitude. We also multiply each perturbation by $c \in \{0.5, 1, 2\}$, keeping the other settings fixed. Table 23 gives the final hard losses for each setting. These are single-run results per configuration, and should not be interpreted as a statistically powered comparison of scale values.

The magnitude has a substantially larger effect than the family: reducing c from 2 to 0.5 improves the final loss consistently for all four distributions. Indeed, for probabilities of the form $p_i \propto \exp((\kappa\ell_i + cg_i)/\tau)$, replacing (κ, τ, c) with $(\kappa/c, \tau/c, 1)$ leaves the probabilities unchanged when $c > 0$. Hence, the noise-magnitude study also probes the effective logit and temperature schedule. The consistent advantage of smaller perturbations in this controlled setting suggests possible schedule improvements, rather than an advantage of switching noise distributions. We retain Gumbel noise for its exact categorical-sampling interpretation.

L.6 Hyperparameter sensitivity and optimization trajectory

We conduct a one-at-a-time sensitivity study on the first transformer block of Llama-3.1-8B-Instruct at 2 bits, varying the temperature schedule, logit scale schedule, and initialization strength while keeping all remaining settings fixed. The default temperature decreases linearly from 2 to 0.05, the logit scale increases linearly from 100 to 500, and the initialization strength is $\alpha = 6$, consistent with Table 8. Table 24 shows the final hard validation losses; the default optimization trajectory across epochs appears in the Gumbel–Softmax rows of Table 21.

Table 22: Noise-family ablation on the first transformer block of Llama-3.1-8B-Instruct at 2-bit precision. All perturbations are mean-centered and variance-matched to Gumbel(0,1), with otherwise identical training settings. Entries are hard validation reconstruction losses (lower is better).

Method / noise	**Target**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**
Gumbel	Self-attention	1.78e-5	9.00e-6	5.96e-6	4.86e-6	3.96e-6	3.58e-6	3.10e-6	2.92e-6	2.83e-6	2.77e-6
Gaussian	Self-attention	1.67e-5	9.54e-6	6.79e-6	4.74e-6	3.87e-6	3.40e-6	3.07e-6	2.89e-6	2.79e-6	2.74e-6
Logistic	Self-attention	1.60e-5	8.88e-6	6.17e-6	4.92e-6	4.05e-6	3.38e-6	3.10e-6	2.92e-6	2.80e-6	2.74e-6
Uniform	Self-attention	1.96e-5	8.34e-6	6.08e-6	4.74e-6	3.84e-6	3.38e-6	3.04e-6	2.88e-6	2.77e-6	2.74e-6
Gumbel	Full block	3.36e-5	1.94e-5	1.53e-5	1.35e-5	1.25e-5	1.18e-5	1.13e-5	1.10e-5	1.08e-5	1.08e-5
Gaussian	Full block	3.39e-5	1.93e-5	1.53e-5	1.34e-5	1.23e-5	1.16e-5	1.12e-5	1.10e-5	1.08e-5	1.07e-5
Logistic	Full block	3.36e-5	1.94e-5	1.53e-5	1.35e-5	1.25e-5	1.17e-5	1.13e-5	1.10e-5	1.08e-5	1.07e-5
Uniform	Full block	3.39e-5	1.94e-5	1.53e-5	1.34e-5	1.23e-5	1.16e-5	1.11e-5	1.09e-5	1.08e-5	1.07e-5

Table 23: Noise-magnitude ablation. Each entry reports final self-attention / full-block hard validation loss. All noise families have variance $\pi^2/6$ before multiplication by c.

Noise family	$c = 0.5$	$c = 1$	$c = 2$
Gumbel	2.29e-6 / 1.03e-5	2.77e-6 / 1.08e-5	3.96e-6 / 1.31e-5
Gaussian	2.25e-6 / 1.02e-5	2.74e-6 / 1.07e-5	3.90e-6 / 1.29e-5
Logistic	2.26e-6 / 1.02e-5	2.74e-6 / 1.07e-5	3.87e-6 / 1.29e-5
Uniform	2.25e-6 / 1.01e-5	2.74e-6 / 1.07e-5	3.96e-6 / 1.31e-5

The method is reasonably stable to the tested changes in temperature and initialization strength, but more sensitive to the logit scale schedule. Increasing α from 6 to 9 or increasing the initial temperature from 2 to 4 slightly improves self-attention reconstruction, while the default gives the best full-block result. By contrast, reducing the logit scale schedule to $10 \to 50$ raises the full-block loss to 2.24×10^{-5}. Overall, the default schedule offers a balanced choice and is used unchanged in the main experiments. These experiments concern a single block and do not establish optimality across all model sizes or training budgets.

Table 24: One-at-a-time sensitivity study for 2-bit GSQ on the first Llama-3.1-8B-Instruct block. Entries are final hard validation losses (lower is better). The default schedule uses $\tau : 2 \to 0.05$, $\kappa : 100 \to 500$, and $\alpha = 6$.

Varied parameter	**Setting**	**Self-attention**	**Full block**
—	Default	1.94e-6	**9.27e-6**
Temperature	$4 \to 0.05$	**1.92e-6**	9.46e-6
Temperature	$1 \to 0.05$	2.31e-6	1.03e-5
Logit scale	$10 \to 50$	6.00e-6	2.24e-5
Logit scale	$1000 \to 5000$	2.96e-6	1.45e-5
Initialization strength α	3	2.23e-6	9.58e-6
Initialization strength α	9	**1.92e-6**	9.39e-6

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: All claims are justified in the paper. The practical claims are shown in the "Experiments" Section.

   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: We discuss the limitations in a Section with the same name in the Appendix.

   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: The paper has one theoretical lemma in the Appendix with its proof.

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: We explained all details for our experiments in "Experiment" section and "Additional Experimental Details" in the Appendix.

   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: We provided an anonymous code.

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: All details are presented in "Experiments" and "Additional Experimental Details" Sections.

   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: [Yes]

   Justification: For the benchmarks with a high variance we repeat the experiments multiple times and report the mean of results.

   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: Our accuracy results are independent of the GPU type but for speedup results we provide details about GPU type and configurations in "Experiments" Section.

   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: We preserve anonymity in both text and codebase and follow the 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: [N/A]

    Justification: No societal impact of the work performed.

    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 paper poses no such risks.

    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: [Yes]

    Justification: We use open-source models and datasets in the paper and cited all of them.

    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: [N/A]

    Justification: The paper does not release new assets.

    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 paper does not involve crowdsourcing nor 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 paper does not involve research with human subjects.

    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 core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.