Fitting Is Not Enough: Smoothness in Extremely Quantized LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) achieve strong performance but incur high deployment costs, motivating extremely low-bit but lossy quantization. Existing quantization algorithms mainly focus on improving the numerical accuracy of forward computation to eliminate performance degradation. In this paper, we show that models also suffer from systematic smoothness degradation under extreme quantization, which cannot be explained by numerical accuracy alone. We confirm that input gradients serve as an effective proxy for smoothness in transformer-based LLMs and use this metric to reveal limitations in both post-training quantization and quantization-aware training. Based on this analysis, we propose a smoothness-preserving principle to maintain gradient propagation during quantization. Experiments across multiple models and tasks demonstrate that preserving smoothness yields additional benefits beyond numerical accuracy. Our study suggests smoothness as an important design consideration for future extreme quantization methods.

1 Introduction

Large language models (LLMs) face substantial deployment costs, posing a major bottleneck for real-world adoption. To unlock model capability under constrained budgets, model quantization is widely used, replacing high-bit value representations with low-bit counterparts (Frantar et al., 2022; Liu et al., 2025). Model quantization has now been pushed to an extremely low bit-width, such as 1-bit (Xu et al., 2024; Li et al., 2025) or even sub-1-bit (Dong et al., 2025). Under such extreme bit-width compression, models often suffer from substantial performance degradation. Existing studies attribute this degradation primarily to numerical precision loss, and consequently focus on preserving the numerical precision of forward computation as much as possible (Huang et al., 2024; Li et al., 2025). A natural question then arises: *Is the collapse of model performance solely caused by numerical precision in extremely low-bit quantization?*

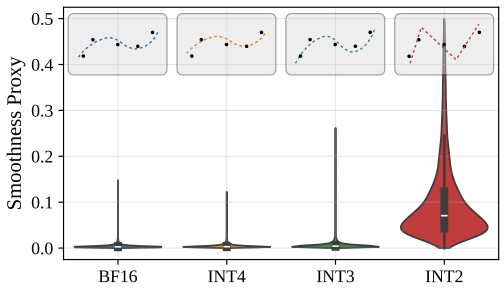


Figure 1: Smoothness score distribution of LLaMA-2-7B quantized models under different GPTQ precisions. Higher scores indicate worse smoothness.

Precision loss alone is insufficient to fully account for the behavioral changes observed in extremely quantized models. We find that such models also exhibit pronounced degradation in smoothness, as shown in Figure 1, which may be an additional source of model capability loss. From the perspective of learning theory, smoothness has long been recognized as being closely related to generalization (Bahri et al., 2022), robustness (Cohen et al., 2019), and training stability (Lyu et al., 2022). Models with poor smoothness often exhibit high sensitivity to small perturbations, leading to output fluctuations and amplified generalization errors. Extensive prior work in machine learning and vision shows that smoothness degradation is typically accompanied by declines in performance and reliability (Dwivedi et al., 2025; Lee et al., 2025; Li et al., 2024; Chan et al., 2020). Nevertheless, despite its well-established importance, smoothness remains insufficiently understood in transformer-based LLMs under extreme quantization.

To characterize smoothness in transformer-based LLMs, we first formalize an effective smoothness proxy. Inspired by vision models and tasks (Chan et al., 2020; Khromov and Singh, 2024; Lee et al., 2025), we employ the norm of *input gradients* to

measure the smoothness of LLMs. Our analysis shows that input gradients serve well as a reliable proxy for LLM smoothness, enabling systematic comparison of different quantization models in extremely low-bit settings.

Based on this metric, we observe that existing extremely low-bit post-training quantization (PTQ) algorithms consistently exhibit a clear trend of smoothness degradation, which is not an isolated phenomenon. We identify the core cause as their incomplete optimization objectives. While minimizing numerical reconstruction error is reasonable, they inevitably compromise the smoothness under extremely low-bit settings. Motivated by this analysis, we propose learnable gradient preserving (**LGP**) that supplements the missing components in prior optimization designs. Accordingly, the original gradient should be explicitly maintained to preserve the smoothness as much as possible.

We further analyze the impact of smoothness on quantization-aware training (QAT). Although QAT explicitly models quantization noise during training, it fails to eliminate smoothness degradation under extremely low-bit settings. Unlike PTQ, where instability affects end-to-end input gradients, the smoothness issue in QAT predominantly manifests as amplified intermediate gradients of hidden states. To counteract this, we introduce loss of gradient regularization (**LGR**) during training to mitigate layer-wise smoothness disruption.

Extensive experiments on both PTQ and QAT show that smoothness yields additional performance gains, independent of the specific algorithm. Rather than proposing a competing algorithm to others, we highlight a design principle for extreme quantization that is often overlooked in prior studies: smoothness preservation should be considered on par with fitting precision. We hope that this perspective will provide new insights into the design of future extreme quantization methods. Overall, this paper makes the following three contributions:

- We establish the feasibility of using input gradients as a proxy for smoothness in transformer-based LLMs, and reveal systematic smoothness degradation in existing extreme quantization methods.
- We identify the deficiencies in extreme quantization algorithms, and propose a smoothness-preserving design principle that aims to maintain model smoothness throughout the quantization process.
- Extensive experiments across models and settings show that improving smoothness in extremely quantized models yields additional benefits beyond those algorithms provided.

2 Preliminary

2.1 Network Smoothness

Lipschitzness is the most relevant metric of smoothness in neural networks. Generally, let $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be a function defined on a domain $\mathcal{D}$. The function f is said to be C-Lipschitz continuous with respect to the α-norm if there exists a constant $C > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$:

$$\|f(\mathbf{x}) - f(\mathbf{y})\|_\alpha \leq C\|\mathbf{x} - \mathbf{y}\|_\alpha. \tag{1}$$

The Lipschitz constant can also be simply given by

$$C = \sup_{\mathbf{x}\in\mathcal{D}} \|\nabla_{\mathbf{x}} f\|_{\tilde{\alpha}}, \tag{2}$$

where $\nabla_{\mathbf{x}} f$ is the Jacobian of f with respect to the input $\mathbf{x}$. Here, $\tilde{\alpha}$ denotes the dual norm of α if $m = 1$; otherwise, $\tilde{\alpha} = \alpha$. For simplicity, we fix $\alpha = \tilde{\alpha} = 2$ in the rest of this paper.

A smaller constant C implies limited output variation under small input perturbations. It is closely associated with the generalization and robustness. Unfortunately, computing the exact value of C is NP-hard (Virmaux and Scaman, 2018). Therefore, we approximately characterize the network smoothness solely by estimating the upper and lower bounds of C.

The simplest, yet significantly loose, upper bound is the product of the Lipschitz constants of each layer. Consider an L-layer network $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, defined as $f_{\boldsymbol{\theta}} = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$. An upper bound on its Lipschitz constant is given by

$$C \leq \prod_{i=1}^{L} \sup_{\mathbf{x}^{(i-1)}\in \mathrm{dom}(f^{(i)})} \|\nabla_{\mathbf{x}^{(i-1)}} f^{(i)}\|_2 = C_{\text{upper}}, \tag{3}$$

where $\mathbf{x}^{(i-1)}$ denotes the input for the i-th layer and $\mathrm{dom} f$ is the definition area of f.

The lower bound is estimated by sampling from the domain $\mathcal{D}$. Using the obtained small subset S, we determine C_{lower} as:

$$C \geq \sup_{\mathbf{x}\in S} \|\nabla_{\mathbf{x}} f_{\boldsymbol{\theta}}\|_2 = C_{\text{lower}}. \tag{4}$$

C_{lower} reduces the complexity of accurately estimating C. However, a drawback is that it merely

reflects the local gradient magnitude at specific points rather than the global landscape. An alternative is to compute the expected input gradient:

$$C_{\text{avg}} = \mathbb{E}_{\mathbf{x}\in S}\|\nabla_{\mathbf{x}} f_{\boldsymbol{\theta}}\|_2. \tag{5}$$

Although this metric does not strictly satisfy the definition of Lipschitz constant, it provides a valid and often more practical estimate of C. For more details, please refer to Khromov and Singh (2024).

2.2 Model Quantization

Model quantization converts the 16-bit floating-point formats commonly used in LLMs into low-bit representations (Frantar et al., 2022; Xiao et al., 2023). While most studies focus on low-bit integer representations, floating-point formats (Wang et al., 2025) and codebook-based encoding(Tseng et al., 2024) are also investigated. In this work, we focus on fixed-point quantization for our analysis.

Quantization converts precision through scaling and shifting. The process is formulated as

$$Q(w) = \text{clamp}(\lfloor \frac{w}{h} \rceil + z, 0, 2^N - 1), \tag{6}$$

where $\lfloor \cdot \rceil$ denotes the rounding-to-nearest, and clamp represents the clipping operation. The parameters h and z are obtained as follows:

$$\begin{aligned} h &= \frac{\max(w) - \min(w)}{2^N - 1}, \\ z &= -\lfloor \min(w)/h \rceil. \end{aligned} \tag{7}$$

Here, N denotes the bit-width of the integer. For memory efficiency, matrix rows or columns are often quantized in groups, each sharing the same h and z, typically of size 64 or 128.

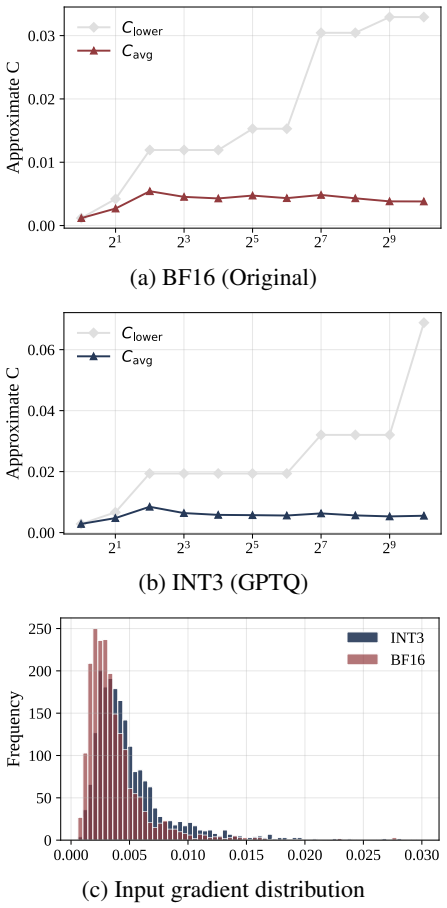
Correspondingly, the dequantization process maps the discrete integer values back to the floating-point domain to enable subsequent computation. It can be defined as

$$\hat{w} = Q^{-1}(Q(w)) = h \cdot (Q(w) - z). \tag{8}$$

The recovered value $\hat{w}$ serves as an approximation of the original weight w, with the quantization error determined by the bit-width N, the clipping range, and the group-wise statistics.

3 Smoothness Proxy

While bounds on Lipschitz constants effectively characterize model smoothness, existing conclusions are predominantly derived from simple architectures or vision tasks, such as MLPs (Shi et al., 2022), ResNets (Chan et al., 2020), or ViTs (Khromov and Singh, 2024). It remains unclear whether these findings hold for transformer-based LLMs and quantized LLMs. In this section, we discuss the applicability of approximate Lipschitz constants to LLMs, focusing on C_{upper}, C_{lower}, and C_{avg}.



(a) BF16 (Original)

(b) INT3 (GPTQ)

(c) Input gradient distribution

Figure 2: Approximate Lipschitz constant of LLaMA-2-7B on one input sequence under different precisions.

It is well-established that C_{upper} is ill-suited for characterizing smoothness across all architectures. By neglecting interlayer dependencies, it yields a vacuous bound that diverges exponentially from the true Lipschitz constant as depth increases. Furthermore, computing layer-wise Jacobians incurs prohibitive computational costs.

In fact, C_{lower} serves as a more accurate proxy, adhering closely to the definition of C via sampling estimation. However, due to the non-differentiable sup operation, it is ill-suited for differentiable contexts. Conversely, while C_{avg} offers differentiability, it deviates from the strict definition of C. We now address two pivotal questions: (a) Can C_{avg} approximate C in transformer-based LLMs? and (b) Does this approximation hold for quantized LLMs?

To facilitate measuring the impact of input per-

turbations on LLM outputs, we define f as the language modeling objective combined with the cross-entropy loss. Consequently, f takes the form $f : \mathbb{R}^n \to \mathbb{R}$. A key benefit is that $\nabla_{\mathbf{x}} f$ simplifies to a gradient vector instead of a Jacobian matrix, thereby avoiding the excessive computation and memory overhead associated with Jacobians.

We provide a formal definition of $\nabla_{\mathbf{x}} f$ in the context of LLMs. For an input sequence $(w_1, \ldots, w_T)$, let $\mathbf{x}_t^{(i)}$ denote the input hidden state of token w_t at layer $i = 1, \ldots, L$. The smoothness proxy, referred to as "input gradient", is defined as $\nabla_{\mathbf{x}^{(0)}} f$, where $\mathbf{x}^{(0)}$ is the token embedding of w_t.

We investigate the relationship between C_{lower} and C_{avg} on Llama-2-7B. As shown in Figure 2a, C_{lower} converges as the number of input tokens increases. Although C_{lower} exhibits a few distinct spikes, these are attributed to a very small number of tokens with large gradients. As illustrated in Figure 2c, the gradients for nearly all tokens remain below 0.02. In the absence of these outliers, the distribution of C_{lower} would be flat. Meanwhile, despite a magnitude gap between the two, C_{avg} exhibits a trend similar to that of C_{lower}. Therefore, although C_{avg} does not strictly adhere to the definition of C, it remains a valid metric for estimating trends in LLM smoothness.

This conclusion also holds for the quantized models shown in Figure 2b. Additionally, Figure 2c illustrates the shift in input gradients for quantized models. We observe that at 3-bit quantization, the input gradients exhibit a noticeable increase, even though the quantization loss remains minimal.

4 Gradient Preservation for PTQ

In this section, we first analyze why existing PTQ algorithms fail to avoid the smoothness problem. We then introduce learnable gradient preservation as an illustrative solution and empirically demonstrate its efficacy through experiments.

4.1 Incomplete Optimization Objective

PTQ compresses each module of an LLM in a layer-wise manner, aiming to align the behavior of the quantized module with that of the original. Let z denote a specific layer in the LLM, where $\mathbf{X}$ represents its input activations and $\boldsymbol{\theta}$ the layer weights. Generally, all PTQ algorithms optimize the following objective during the quantization process:

$$\min_{\hat{\boldsymbol{\theta}}} \|z_{\boldsymbol{\theta}}(\mathbf{X}) - z_{\hat{\boldsymbol{\theta}}}(\mathbf{X})\|_F^2. \tag{9}$$

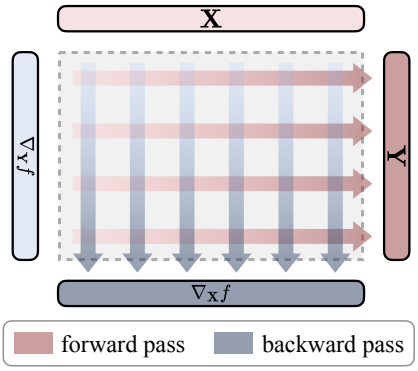


Figure 3: In a linear layer, the forward pass is computed row-wise, whereas the backward pass is column-wise.

Here, $\hat{\boldsymbol{\theta}}$ denotes the quantized weights. This optimization objective implies that, regardless of the values of quantized weights, the primary requirement is to align the output activation of the quantized model with that of the original one. However, it can not ensure that the module smoothness is not compromised after quantization. Specifically, in a linear layer $\mathbf{Y} = \mathbf{W}\mathbf{X}$, the objective for quantizing the weight can be formulated as:

$$\min_{\hat{\mathbf{W}}} \|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_F^2. \tag{10}$$

This addresses the approximation of the forward propagation. Turn to consider the backward propagation. Let the gradient with respect to $\mathbf{Y}$ be denoted as $\nabla_{\mathbf{Y}} f = \mathbf{G}$. According to the chain rule, the gradient with respect to the input $\mathbf{X}$ is given by $\nabla_{\mathbf{X}} f = \mathbf{W}^\top \mathbf{G}$. Considering that the backward propagation of the original model remains intact, we further need to optimize:

$$\min_{\hat{\mathbf{W}}} \|\mathbf{W}^\top \mathbf{G} - \hat{\mathbf{W}}^\top \mathbf{G}\|_F^2. \tag{11}$$

To maintain accuracy while preserving smoothness, we should optimize the following objective:

$$\min_{\hat{\mathbf{W}}} \underbrace{\|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_F^2}_{\text{accuracy}} + \underbrace{\|\mathbf{W}^\top \mathbf{G} - \hat{\mathbf{W}}^\top \mathbf{G}\|_F^2}_{\text{smoothness}}. \tag{12}$$

The first term of this objective aims to ensure the fitting fidelity of the quantized model to the original one, while the second term seeks to preserve the smoothness of the original model. As shown in Figure 3, the optimization objectives in Eq. 10 and 11 are mutually orthogonal.

This indicates that the classical PTQ optimization is incomplete. While this incompleteness has a negligible impact at higher bit-widths (> 3 bits), addressing it yields noticeable gains in extreme quantization algorithms.

Method	Wiki2	C4	BoolQ	OBQA	RTE	Wino.	Hella.	PIQA	MathQA	ARC-e	ARC-c	Avg.
Llama-2-7B ($\alpha_1 = 1e6$ in LGP)												
BF16	5.47	6.97	77.71	44.20	62.82	69.14	76.00	79.11	28.31	74.54	46.25	62.01
GPTQ	39.58	31.19	58.34	26.80	54.51	52.49	37.13	58.54	22.68	38.43	25.26	41.58
OmniQ	**13.76**	14.43	**63.27**	31.20	54.51	**55.88**	50.70	66.59	22.71	43.60	25.77	46.03
+LGP	13.77	**14.37**	62.35	**33.00**	**55.23**	55.25	**50.81**	**67.95**	**23.38**	**45.45**	**26.62**	**46.67**
Llama-2-13B ($\alpha_1 = 1e5$ in LGP)												
BF16	4.88	6.47	80.55	45.20	65.34	72.14	79.38	80.52	31.86	77.44	49.06	64.61
GPTQ	30.79	29.81	54.19	27.60	50.18	51.14	41.08	61.37	22.24	38.64	26.02	41.38
OmniQ	9.30	10.90	66.69	33.20	55.23	58.17	**60.38**	**71.82**	**25.90**	57.15	31.83	51.15
+LGP	**9.25**	**10.87**	**67.09**	**34.20**	**56.68**	**58.17**	60.00	71.71	25.06	**58.08**	**32.59**	**51.51**
Qwen-3-0.6B ($\alpha_1 = 1e4$ in LGP)												
BF16	20.96	25.43	63.82	31.40	53.79	56.43	47.30	67.25	31.46	55.93	33.70	49.01
GPTQ	7.1e3	2.4e4	41.99	23.00	51.62	50.83	25.59	52.56	19.39	25.84	25.00	35.09
OmniQ	310.2	141.8	61.07	25.00	52.71	48.78	28.28	**56.96**	**22.81**	31.86	22.26	38.86
+LGP	**295.0**	**140.1**	**61.50**	**25.00**	**52.71**	**51.62**	**28.44**	56.75	22.45	**31.94**	**22.53**	**39.22**
Qwen-3-8B ($\alpha_1 = 1e2$ in LGP)												
BF16	9.73	13.30	86.61	41.80	77.98	68.11	74.90	77.86	49.45	80.81	56.74	68.25
GPTQ	52.34	43.40	44.95	27.00	51.62	50.98	35.44	53.97	19.16	27.36	23.81	37.14
OmniQ	**37.47**	**30.36**	73.70	**33.00**	**54.15**	57.62	**48.90**	68.72	24.46	55.72	32.42	49.85
+LGP	37.84	30.79	**73.94**	30.80	52.71	**59.12**	47.89	**68.72**	**25.06**	**57.32**	**33.19**	**49.86**

Table 1: Main W2A16 quantization results of the evaluation experiment. The best scores are in bold.

4.2 Learnable Gradient Preservation

The objective in Eq. 10 has a high-precision approximate closed-form solution. Leveraging second-order information (i.e., the Hessian matrix $\mathbf{H} = \mathbf{X}\mathbf{X}^\top$), GPTQ iteratively quantizes weights while adjusting the remaining ones to minimize error. Due to its effectiveness, it serves as the backbone for numerous methods. Unfortunately, the joint smoothness objective (Eq. 12) is incompatible with GPTQ-like solvers due to mutually orthogonal optimization directions: Eq. 10 proceeds row-wise, while Eq. 11 demands column-wise iteration.

As an example solution, we focus on addressing the problem itself, leaving the derivation of a closed-form solution for the joint optimization to future work. We choose to optimize the smoothness of the OmniQuant (Shao et al., 2024). Unlike GPTQ, which quantizes weights iteratively, OmniQuant introduces learnable parameters during the quantization process and enhances performance via layer-wise distillation. Specifically, the core of OmniQuant lies in learnable weight clipping, which introduces parameters γ and β:

$$h = \frac{\gamma \max(w) - \beta \min(w)}{2^N - 1}, \qquad z = -\lfloor \beta \min(w)/h \rceil. \tag{13}$$

They are then updated using the layer-wise distillation loss defined in Eq. 9. Although OmniQuant is no longer the state-of-the-art baseline for extreme quantization, this does not undermine the validity of our demonstration regarding the benefits of introducing smoothness.

To preserve the smoothness of the original model, we propose learnable gradient preservation (LGP) based on OmniQuant and our analysis. Consequently, the layer-wise distillation objective incorporated with LGP is formulated as:

$$\min_{\hat{\boldsymbol{\theta}}} \|z_{\boldsymbol{\theta}} - z_{\hat{\boldsymbol{\theta}}}\|_F^2 + \alpha_1 \|\nabla_{\mathbf{X}} f_{z_{\boldsymbol{\theta}}} - \nabla_{\mathbf{X}} f_{z_{\hat{\boldsymbol{\theta}}}}\|_F^2, \tag{14}$$

where the coefficient α_1 controls the strength of the smoothness objective.

4.3 Experimental Setup

We evaluate the effectiveness of introducing LGP on two mainstream models, Llama-2 and Qwen-3. The model sizes range from 0.6B to 13B.

Baselines We select the original model, 2-bit GPTQ, and 2-bit OmniQuant as our baselines. The original model serves as a reference to quantify the performance gap induced by quantization. By comparing OmniQuant with GPTQ, we highlight the accuracy gains achieved by the learnable weight clipping (i.e., optimizing Eq. 9). Our method is

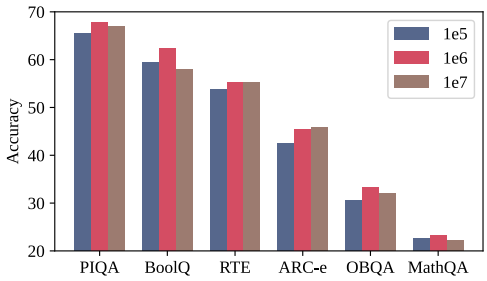


Figure 4: Comparison of zero-shot accuracy under different values of the coefficient α_1.

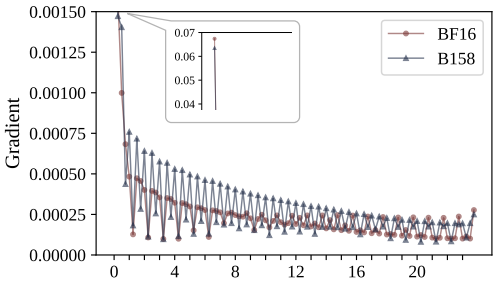


Figure 5: Layer-wise gradient of BF16 and B158 models. The left side corresponds to shallow layers.

designed to demonstrate the effectiveness of introducing LGP for joint optimization (Eq. 14). For calibration, all methods utilize 128 sequences from the C4 dataset, each with a length of 2048. The quantization group size is set to 128.

LGP For all models, regardless of whether LGP is incorporated, the layer-wise distillation process is conducted for 40 epochs. The coefficient α_1 is typically set to be $1e^K$ where K is a positive integer, as listed in Table 1.

Evaluation To assess the performance of the baselines and LGP, we calculate perplexity using sequences randomly sampled from Wikitext2 (Merity et al., 2017) and C4 (Raffel et al., 2020). Additionally, we report zero-shot accuracy across a range of tasks, including Winogrande (Sakaguchi et al., 2021), Hellaswag (Zellers et al., 2019), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), ARC (Clark et al., 2018), OBQA (Mihaylov et al., 2018), MathQA (Amini et al., 2019), and RTE (Wang et al., 2018).

4.4 Main Results

Table 1 presents the evaluation results of incorporating LGP into PTQ. Compared to GPTQ, OmniQuant achieves substantial accuracy gains in 2-bit weight quantization. However, a noticeable performance gap remains relative to the FP16 baseline. Upon integrating LGP, we first observe that the language modeling capability is maintained or even slightly enhanced. Furthermore, LGP demonstrates improvements across the majority of accuracy tasks for each model. Notably, although LGP does not explicitly optimize for fitting accuracy, it yields further improvements in both PPL and Accuracy. We attribute this success to the inherent advantages of smoothness, which mitigates erratic output shifts.

4.5 Ablation

LGP incorporates a parameter α_1 to balance the objectives of fitting fidelity and smoothness. An insufficiently small α_1 renders LGP ineffective, while an overly large value disrupts the fitting process, leading to accuracy degradation. To avoid these issues, we employ a heuristic method for selecting α_1. Given that the norm of the backward gradients is smaller than the fitting loss by multiple orders of magnitude, we choose a sufficiently large α_1 to bring both terms to a comparable scale. For example, for Llama-2-7B, we perform a search across magnitudes ranging from $1e4$ to $1e8$. The difference of varying α_1 is shown in Figure 4.

5 Gradient Regularization for QAT

QAT typically employs the language modeling loss, thereby also neglecting the smoothness issue. Intuitively, $\nabla_{\mathbf{x}^{(0)}} f$ appears suitable for constraining model smoothness. However, the existence of *gradient ridge* at the 0-th layer makes $\nabla_{\mathbf{x}^{(0)}} f$ invalid for this purpose.

5.1 Gradient Ridge

We begin by examining the layer-wise gradient discrepancies between our trained high-precision BF16 model and its 1.58-bit counterpart (Ma et al., 2024). Specifically, we monitor the input and output gradients of the `input_layernorm` and `post_attention_layernorm` for each layer, as illustrated in Figure 5. Both models utilize the same training setting.

Initially, we observe that the intermediate gradients in the B158 model are noticeably larger than those in the BF16 model, implying that the intermediate states of the former are less smooth. This is especially significant in the early and middle layers. Furthermore, we notice that the input gradients at the 0-th layer for both models spike to a similarly high magnitude (≈6e-2). Why do the input

Size	Method	Wiki2	C4	Wino.	Hella.	PIQA	BoolQ	OBQA	ARC-e	ARC-c	Avg.
0.4B	BF16	58.85	67.76	50.75	26.85	53.97	61.77	26.40	31.48	23.04	39.18
	B158	62.86	**68.11**	49.80	25.51	53.21	62.05	**25.00**	**30.64**	23.46	38.52
	+LGR	**62.28**	69.44	**50.04**	**26.22**	**53.54**	**62.05**	24.40	29.76	**24.49**	**38.64**
1.0B	BF16	60.29	72.43	46.88	27.13	55.28	42.29	24.00	30.93	23.21	35.67
	B158	**65.52**	77.12	48.30	25.70	49.62	37.83	26.20	27.15	24.57	34.20
	+LGR	66.97	**76.15**	**50.28**	**25.75**	**50.54**	**37.83**	**28.20**	**27.61**	**25.43**	**35.09**
1.7B	BF16	57.76	65.87	49.41	27.07	56.04	39.36	24.20	32.24	23.81	36.02
	B158	**55.61**	70.87	**51.22**	26.07	49.62	37.83	26.40	26.30	**25.77**	34.74
	+LGR	55.80	**68.22**	50.51	**26.17**	**51.03**	**38.41**	**26.80**	**27.02**	24.32	**34.89**

Table 2: Main quantization results on our trained FP16 and B158 models. The best scores are in bold.

size	n_{layer}	n_{head}	d_{hidden}	d_{inter}	steps	lr
0.4B	16	16	1024	4096	300k	2e-4
1.0B	24	24	1536	6144	400k	2e-4
1.7B	24	32	2048	8192	400k	2e-4

Table 3: Configurations of different model sizes.

LGR	C4	Wino	PIQA	ARC-e	ARC-c
$\nabla_{\mathbf{x}^{(1)}} f$	68.11	50.83	54.13	29.42	23.29
$\nabla_{\mathbf{x}^{(0)}} f$	70.29	48.22	53.97	29.00	22.78

Table 4: Ablation of different input gradients. We report the 0.4B model training with 400k steps, and $\alpha_2 = 0.1$.

gradients at the 0-th layer remain nearly identical? We *hypothesize* that it stems from the nature of the input: word embeddings that lack semantic integration. The extreme sparsity of such inputs may drive the model into a state of overfitting. As a result, input points in the latent space tend to be situated on "ridges" rather than in "valleys". We term it the "*Gradient Ridge*", an intrinsic being that exists independent of weight compression. For more comparisons, please refer to Section A.2.

5.2 Loss of Gradient Regularization

To enable extremely low-bit models to automatically acquire smoothness during training, we propose the loss of gradient regularization (LGR). Given the aforementioned gradient ridge, we utilize the input hidden states of the 1-st layer instead of the 0-th layer inputs. Specifically:

$$\mathcal{L}_{\text{smooth}} = \frac{1}{N} \sum_{i}^{N} \|\nabla_{\mathbf{x}_i^{(1)}} f\|_F^2, \tag{15}$$

assuming a sequence length of N tokens. Therefore, the overall loss becomes $\mathcal{L} = \mathcal{L}_{\text{lm}} + \alpha_2 \mathcal{L}_{\text{smooth}}$, where α_2 controls the regularization intensity. We will demonstrate the drawbacks of relying on 0-th layer input gradients subsequently.

5.3 Experimental Setup

We evaluate the FP16 baseline against the standard B158 model and its smoothed counterpart. We train models of three different sizes on the OpenWebText2 dataset, and their configurations are listed in Table 3, with detailed training settings provided in the Appendix. We fix $\alpha_2 = 0.01$ for all runs. The evaluation protocol aligns with that of Section 4.3.

5.4 Main Results

Table 2 presents the performance of B158 models and their LGR-enhanced counterparts. For reference, the results of the FP16 models are also included. It is evident that the models trained with LGR exhibit better performance on the majority of benchmarks. We also observe that the 0.4B model outperforms the 1.7B model on certain tasks. This is likely attributable to the sufficiency of training.

5.5 Ablation

Table 4 shows the superiority of utilizing input gradients from the 1-st layer. In contrast, relying on the 0-th layer may have a detrimental impact on training results. We attribute this potential failure to the corruption of embedding representations.

6 Discussion

6.1 Quantization and Weight Space

Weight quantization is equivalent to shifting the original weights along a specific direction. Eq. 12 and our analysis show that there exist two orthogonal optimization directions that jointly influence quantization performance, as illustrated in Figure 6. In high-precision regimes, both accuracy and smoothness can simultaneously achieve optimality. However, when a substantial distance exists between $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$, the quantization algorithm should

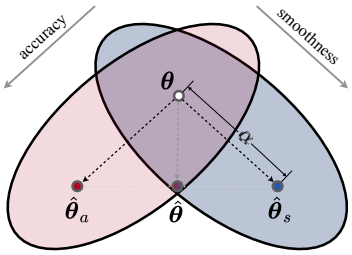


Figure 6: Anisotropy of the weight space. Deviating from $\boldsymbol{\theta}$ reveals two distinct directions where accuracy and smoothness degrade, respectively.

consider smoothness as much as possible, subject to the constraint of ensuring accuracy.

6.2 Smooth Training

Section 5 presents a preliminary exploration of smooth training. Emerging evidence suggests that it is particularly important in extremely low-bit QAT scenarios. Due to space constraints, we detail its effectiveness and provide deeper insights and ablations in Appendix A.1.

6.3 Accuracy + Smoothness = ?

Let $\Delta\mathbf{W} = \mathbf{W} - \hat{\mathbf{W}}$, and Eq. 10 can be simplified to $\min \|\Delta\mathbf{W}\mathbf{X}\|_F^2$. Given that $\mathbf{X}$ adheres to a certain distribution, a $\Delta\mathbf{W}$ exists that minimizes this term, which partially explains the viability of quantization. If we subsequently impose the stronger constraint of Eq. 11 (i.e., $\min \|\Delta\mathbf{W}^\top\mathbf{G}\|_F^2$), does a valid $\Delta\mathbf{W}$ still persist? Does this imply that the original model is the sole candidate capable of simultaneously achieving both accuracy and smoothness? The answer seems to be *no*. This stems from the fact that the two optimization processes essentially involve finding the rows of $\Delta\mathbf{W}$ in the null spaces of $\mathbf{X}$ and the columns must lie in the null space of $\mathbf{G}$. A joint solution exists provided that $\mathcal{N}(\mathbf{X}^\top) \cap \mathcal{N}(\mathbf{G}^\top) \neq \{\mathbf{0}\}$. Indeed, the extreme sparsity of LLMs ensures that $\mathbf{X}$ and $\mathbf{G}$ are low-rank, making the condition:

$$\operatorname{rank}(\mathbf{X}) + \operatorname{rank}(\mathbf{G}) < \min(d_{\text{in}}, d_{\text{out}})$$

generally valid, though the feasible solution space becomes much narrower. This observation highlights the challenges that dual constraints impose on developing algorithms for extreme quantization.

7 Related Work

7.1 Model Quantization

Model quantization compresses high-precision weights into low-bit representations. Post-training quantization (PTQ) converts a trained model to low-bit precision using optimization solvers (Frantar et al., 2022; Dettmers et al., 2024). Quantization-aware training (QAT) integrates quantization into training process to mitigate the adverse effects of reduced precision (Liu et al., 2024; Xu et al., 2024). Some studies also explore simultaneous weight and activation quantization (Liu et al., 2025; Xiao et al., 2023). Currently, the widely used lossless quantization level is INT4. Quantization below this level is considered extreme, but it often leads to degraded model performance (Tseng et al., 2024; Huang et al., 2024; Ma et al., 2024; Li et al., 2025; Xu et al., 2025). Nearly all existing extreme quantization focuses primarily on improving representation accuracy to reduce forward computation loss.

7.2 Network Smoothness

Lipschitzness is the most direct smoothness metric in neural networks (Bartlett et al., 2017). However, despite its concise definition in machine learning, accurately estimating the Lipschitz constant for neural networks is highly challenging (Virmaux and Scaman, 2018). Numerous studies attempt to approximate network Lipschitz constants, yet they typically suffer from two limitations. First, they fail to effectively address the exponential growth of estimated bounds with network depth (Virmaux and Scaman, 2018; Wang et al., 2022). Second, most prior work relies on early architectures (Gouk et al., 2021; Singla and Feizi, 2021; Erichson et al., 2021; Zhou et al., 2019; Khromov and Singh, 2024), such as ResNet. In contrast, the Lipschitzness of transformer (Kim et al., 2021; Qi et al., 2023) remains relatively underexplored. Although early studies in vision (Gouk et al., 2021) and robustness (Pauli et al., 2021) investigate the impact of smoothness on model performance via proxies, conclusions on smoothness in LLMs remain scarce.

8 Conclusion

In this work, we highlight a critical yet often overlooked objective in extreme quantization: the smoothness of quantized models. Building upon the viability of input gradients in LLMs, we propose LGP for PTQ and LGR for QAT as preliminary solutions, respectively. We experimentally validate their effectiveness and provide a detailed analysis. We advocate for future research in extreme quantization to explicitly incorporate smoothness into algorithm design.

Limitations

Although we propose a novel smoothness-aware perspective to complete the optimization objectives for extreme compression, we acknowledge three main limitations that point toward future directions.

Firstly, our proposed methods (LGP and LGR) function as foundational baselines. We prioritized simplicity and versatility to demonstrate the validity of the smoothness hypothesis. Consequently, these methods may not represent the optimal mathematical solution for the joint optimization problem we formulated. We anticipate that future work can build upon our dual-constraint analysis to design more advanced solvers.

Secondly, while we provide intuitive explanations like the "Gradient Ridge", its underlying causes are not yet fully understood. We regard this work as a stepping stone that invites the community to delve deeper into its underpinnings, which may contribute to the explainability of LLMs.

Lastly, constrained by available computational resources, our experiments are confined to models of moderate scale and trained with a limited number of steps on a restricted dataset. While we observe consistent improvements within this scope, the behavior of smoothness constraints under the regime of massive-scale pre-training remains an open question. We hope that future work can further scale these experiments.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2357–2367.

Dara Bahri, Hossein Mobahi, and Yi Tay. 2022. Sharpness-aware minimization improves language model generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7360–7371.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. 2017. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems (NeurIPS)*, 30:6240–6249.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 7432–7439.

Alvin Chan, Yi Tay, and Yew-Soon Ong. 2020. What it thinks is important is important: Robustness transfers through input gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 332–341.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457v1*.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1310–1320.

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

Peijie Dong, Lujun Li, Yuedong Zhong, DaYou Du, Ruibo FAN, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Yike Guo, and 1 others. 2025. STBLLM: Breaking the 1-bit barrier with structured binary llms. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Pulkit Dwivedi, Benazir Islam, and Mansi Kajal. 2025. Smooth gradient loss: a loss function for gradient regularization in deep learning optimization. *The Journal of Supercomputing*, 81:1–45.

N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. 2021. Lipschitz recurrent neural networks. In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323v2*.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. 2021. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and

Xiaojuan Qi. 2024. BiLLM: Pushing the limit of post-training quantization for LLMs. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 20023–20042.

Grigory Khromov and Sidak Pal Singh. 2024. Some fundamental aspects about lipschitz continuity of neural networks. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. 2021. The lipschitz constant of self-attention. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 5562–5571.

Hongsin Lee, Seungju Cho, and Changick Kim. 2025. Indirect gradient matching for adversarial robust distillation. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Tao Li, Pan Zhou, Zhengbao He, Xinwen Cheng, and Xiaolin Huang. 2024. Friendly sharpness-aware minimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5631–5640.

Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Linghe Kong, Yulun Zhang, Xiaokang Yang, and 1 others. 2025. ARB-LLM: Alternating refined binarizations for large language models. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, pages 467–484.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025. SpinQuant: LLM quantization with learned rotations. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Kaifeng Lyu, Zhiyuan Li, and Sanjeev Arora. 2022. Understanding the generalization benefit of normalization layers: Sharpness reduction. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:34689–34708.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2381–2391.

Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. 2021. Training robust neural networks using lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126.

Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. 2023. LipsFormer: Introducing lipschitz continuity to vision transformers. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21:5485–5551.

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

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

Zhouxing Shi, Yihan Wang, Huan Zhang, J Zico Kolter, and Cho-Jui Hsieh. 2022. Efficiently computing local lipschitz constants of neural networks via bound propagation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:2350–2364.

Sahil Singla and Soheil Feizi. 2021. Skew orthogonal convolutions. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 9756–9766.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 48630–48656.

Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:3835–3844.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Ruizhe Wang, Yeyun Gong, Xiao Liu, Guoshuai Zhao, Ziyue Yang, Baining Guo, Zheng-Jun Zha, and Peng CHENG. 2025. Optimizing large language model training using FP4 quantization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 62937–62957.

Zi Wang, Gautam Prakriya, and Somesh Jha. 2022. A quantitative geometric approach to neural-network smoothness. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:34201–34215.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 38087–38099.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. OneBit: Towards extremely low-bit large language models. *Advances in Neural Information Processing System (NeurIPS)*, 37:66357–66382.

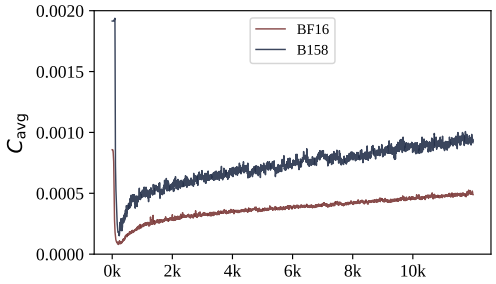
Yuzhuang Xu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. 2025. CRVQ: Channel-relaxed vector quantization for extreme compression of LLMs. *Transactions of the Association for Computational Linguistics (TACL)*, 13:1488–1506.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4791–4800.

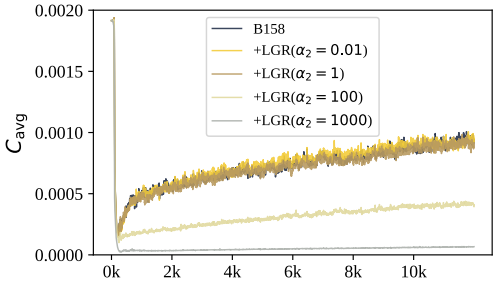
Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. 2019. Lipschitz generative adversarial nets. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 7584–7593.

A Appendix

A.1 Smoothness in Training Process



(a) BF16 model and B158 model



(b) B158 model and different LGR settings

Figure 7: Evolution of C_{avg} during training across different models and settings. Here, we use 0.4B model.

In this section, we discuss several topics related to smoothness-aware training, reflecting on insights and conjectures derived from our experiments. We primarily address the following three questions:

a) Why is RMSNorm excluded from the BitNet b1.58 model in our QAT experiments?

In the standard BitNet b1.58 architecture (Ma et al., 2024), the activations entering the linear layer are first normalized via *RMSNorm* before performing matrix multiplication with the ternary weights. This process is defined as follows:

$$\begin{aligned}\mathbf{X}' &= \text{RMSNorm}(\mathbf{X}) \\ \mathbf{W}' &= \text{Ternarize}(\mathbf{W}) \\ \mathbf{Y} &= \mathbf{W}'\mathbf{X}'.\end{aligned}$$

In our LGR implementation, we deliberately excluded *RMSNorm* as a controlled variable strategy. It is well established that BitNet models incorporating *RMSNorm* achieve performance comparable to full-precision baselines. This effectiveness stems from two factors: first, *RMSNorm* normalizes activations during the forward pass, enhancing computation accuracy; second, it significantly flattens backward gradients, substantially improving ternary model smoothness.

However, this dual impact makes it difficult to isolate the specific benefits of smoothness. By removing *RMSNorm*, we align the forward computation process with that of the full-precision model. Consequently, any observed gains can be attributed solely to smoothness. Furthermore, the success of *RMSNorm* itself implicitly validates the accuracy-smoothness trade-off idea proposed in this paper.

b) Is smoother training always better?

In QAT, maximizing smoothness is not unconditionally beneficial. Moreover, it is unrealistic to expect a ternary model to achieve parity with a full-precision model in both smoothness and performance simultaneously. The core idea of this work is to identify the smoothest candidate among the manifold of quantized solutions that have equivalent fitting accuracy.

The guiding principle is **moderate smoothing**: enhancing smoothness without compromising accuracy. We selected the hyperparameter $\alpha_2 = 0.01$. As shown in Figure 7, increasing α_2 significantly improves model smoothness; at $\alpha_2 = 100$, the ternary model exhibits smoothness comparable to the full-precision baseline. However, this comes at a cost: due to over-smoothing, the accuracy degrades, causing perplexity to deteriorate from 97.4 to 130.6, similar to randomly regularization. Consequently, excessive smoothing must be avoided.

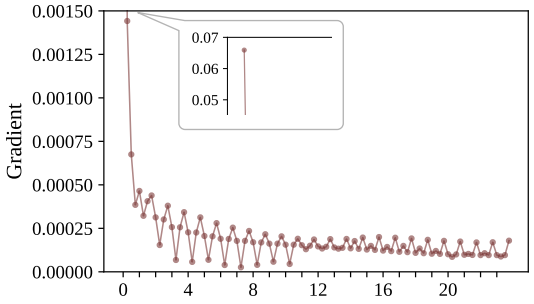
c) How can smoothness be incorporated into industrial training practices?

Since the norm of input gradients is negligible relative to the training loss, introducing smoothness constraints in the early stages of training proves ineffective. Furthermore, adhering to the principle of moderate smoothing, LGR should not drastically alter the intrinsic smoothness of quantized model. Consequently, we recommend incorporating smoothness training only during the mid-to-late stages of the training process. This strategy may preserves pre-trained knowledge while conserving computational resources.

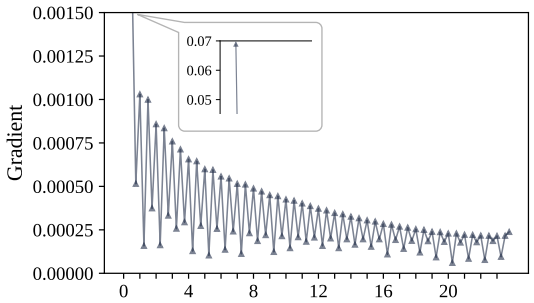
A.2 Gradient Ridge

We observe that the "Gradient Ridge" is not sporadic. It is independent of subtle architectural modifications or quantization settings. In Section 5.1, we initially demonstrate its presence in both FP16 and B158 models. To reinforce the universality of this observation, we conducted similar tests on the BitNet b1.58 architecture with *RMSNorm*.

As illustrated in Figure 8a, even though the *RMSNorm*-equipped model exhibits smoothness comparable to the FP16 baseline, the input gradient at 0-th layer still spikes to a very high magnitude. Note that all results reported thus far, including those in Section 5.1, is derived under full-weight training settings. Furthermore, we test with using frozen FP16 embeddings within the B158 model. As depicted in Figure 8b, the conclusion holds firm.



(a) BitNet b1.58 model



(b) B158 model with frozen embedding

Figure 8: Evolution of C_{avg} during training when integrate *RMSNorm* or freeze word embedding.

Based on this evidence, we conclude that the *Gradient Ridge* is by no means coincidental. Although its root cause remains elusive, we believe that exploring it could offer significant benefits for the interpretability of LLMs.

A.3 Residual Quantization and Smoothness

Section 4.1 points out the challenge of optimizing the complete objective (Eq. 12), as the two sub-objectives are orthogonal. Yet, Section 6.3 clarifies that this goal is not impossible. It merely constrains the solution space.

While deriving a closed-form solution for Eq. 12 is challenging, it remains feasible to partially fulfill the smoothness objective while prioritizing accuracy. The extreme sparsity of LLM gradients underpins this feasibility. As illustrated in the Figure 9, input and output gradients across linear layers remain negligible in the vast majority of channels. This implies that by prioritizing the precision of a select few columns, we can ensure that gradient propagation remains unimpaired. As a result,

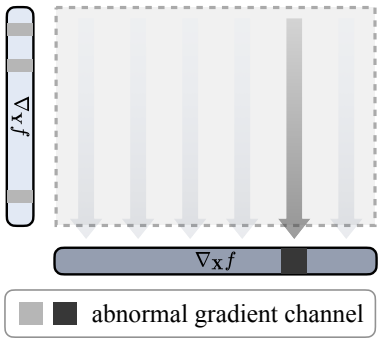


Figure 9: Important/abnormal gradient channels and relaxed optimization.

selecting key column vectors and applying high precision presents an intuitively viable strategy. Here, the objective in Eq. 11 is relaxed, effectively giving an approximate solution.

In fact, similar ideas have already emerged in the field of extreme quantization (Huang et al., 2024; Xu et al., 2025; Li et al., 2025). Methods like *residual quantization* select a subset of pivotal column vectors to fit their quantization errors, significantly enhancing extreme compression performance. The distinction between our proposal and such methods lies in the selection criteria. Residual quantization evaluates column importance based on activation error, defined as $I_i = (w_i - \hat{w}_i)x_i$. In contrast, our relaxed formulation utilizes the magnitude of backward gradients $\nabla_{x_i} f$ as the importance metric. Combining these two indicators may pave the way for designing even more potent extreme compression algorithms, which is a direction we leave for future research.

A.4 Details on Baselines

In this section, we provide supplementary implementation details for the evaluated methods. All code will be released upon de-anonymization of the paper.

GPTQ We adhere to the default configurations provided in the official open-source repositories. Specifically, we set the random seed for data sampling to 0 and employ a Hessian damping factor of 0.01. Moreover, we utilize asymmetric quantization, activation-aware channel reordering, and true sequential quantization.

OmniQuant We set the random seed for data sampling to 0. The learning rate for LWC is configured at 0.01, and the model is trained for alignment for 40 epochs using AdamW optimizer. Additionally, we employ asymmetric quantization and an auxiliary loss in their official code.

LGP Following OmniQuant, we quantize layers sequentially in a shallow-to-deep manner. For each layer, we first perform a full forward and backward pass through the entire model to compute the input and output gradients of this layer. Subsequently, we calculate the smoothness loss of the quantized model and add it to the original OmniQuant loss. Other configurations remain identical to those in OmniQuant.

LGR We conduct the training on 8 NVIDIA A800 GPUs. The setup utilizes a per-device batch size of 1 with 16 gradient accumulation steps. We employ the AdamW optimizer with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.95$, and a weight decay of 0.1. The learning rate follows a cosine schedule with a 1000-step warmup. To help for reproducibility, random seeds for Python, NumPy, PyTorch, and CUDA are all fixed at 1234.