FINE-TUNING LOW-BIT MODELS WITH GRADIENTS IN QUANTIZED CODE SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning low-bit models aims to adapt a quantized model while keeping the final deployed checkpoint in the same low-bit form. This setting is practically important as it reduces memory and inference cost for storage and deployment. Under this constraint, adaptation becomes an optimization problem over quantization codes and scales. Existing continuous low-bit training is efficient, but it can be distorted by straight through estimation error or by post-quantization gap; discrete search is deployment-faithful, but can be computationally inefficient under a finite training budget. We propose code surrogate gradient to find the steepest direction of a local continuous surrogate in code space to accelerate optimization, and perform guided search to preserve deployment faithfulness. Experiments across arithmetic reasoning, instruction following, and structured language understanding show that GradCodeS consistently improves fine-tuning low-bit models across different quantization datatypes.

1 INTRODUCTION

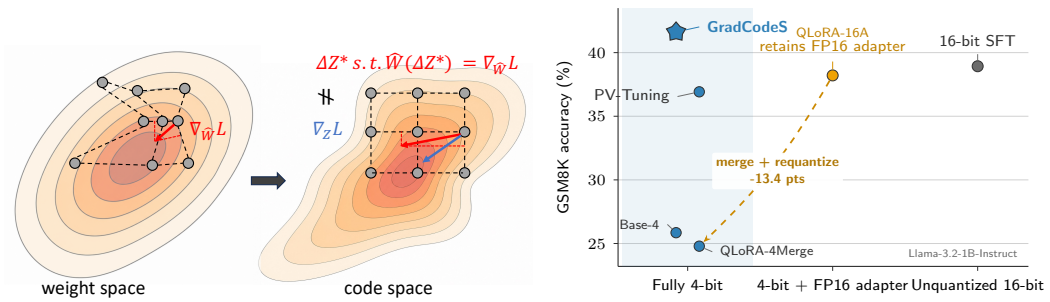
Large language models (LLMs) are increasingly deployed in low-precision formats such as INT4, NF4, and MXFP4 to reduce storage, memory traffic, and inference cost (Xiao et al., 2023; Lin et al., 2024). These savings matter not only for a shared backbone, but also when many task- or user-specific checkpoints must be stored under a fixed memory budget. We study a strict fully low-bit setting in which the targeted weights of every adapted checkpoint remain in the target quantized representation, with only datatype-permitted scale metadata and no high-precision residual adapter at inference time (Jeon et al., 2025; Malinovskii et al., 2024). Under this constraint, fine-tuning is no longer optimization in a nearly continuous floating-point space; it is optimization over quantization codes and scales.

One practical strategy retains continuous optimization. Quantized-backbone adapter methods train high-precision LoRA-style parameters and either keep them at inference time or merge and requantize them afterward (Dettmers et al., 2023; Li et al., 2024; Loeschcke et al., 2024; Xu et al., 2024). This is effective when mixed-precision deployment or a separate conversion stage is acceptable. Under the stricter setting above, however, retaining the adapter violates the deployment constraint, while merging and requantizing can make the evaluated low-bit state differ from the state optimized during training. Quantization-aware and compressed-representation methods move optimization closer to the final checkpoint (Jeon et al., 2025; Malinovskii et al., 2024), but still leave a fundamental question: how should first-order information be expressed when the actual decision variable is a quantization-code index?

Direct discrete and zeroth-order approaches instead evaluate candidates in or near the quantized model family (Zhou et al., 2025; Shang et al., 2026; Xu et al., 2026), but random or population proposals can require many forward evaluations in high-dimensional code spaces. The central challenge is to combine deployment-faithful selection with first-order directional guidance.

To be deployment-faithful, we optimize over quantized code coordinates rather than weight space coordinates. To be efficient, we seek to incorporate first-order information into the discrete search. Based on this principle, we propose GradCodeS, a gradient-guided search method. We first identify that mapping ordinary weight-space gradient to code space is generally not the steepest descent direction in code space, as illustrated in Figure 1(a). We therefore introduce a *code-coordinate*

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



(a) Geometry mismatch between weight and code spaces. (b) Accuracy under different deployment regimes.

Figure 1: Motivation for deployment-faithful code-space optimization. (a) Group scales and non-uniform codebook gaps distort local geometry, so a weight gradient does not identify the best discrete code move. (b) In the Llama-3.2-1B GSM8K example, continuous adaptation either retains an FP16 adapter or loses 13.4 points after QLoRA requantization; GradCodeS improves the fully 4-bit result while retaining a deployable checkpoint.

surrogate gradient by codebook continuation, serving as first-order information in code space with theoretical support. GradCodeS uses code-coordinate surrogate gradient descent to provide a reference point and then samples multiple nearby code candidates biased toward that reference, and selects an update only according to its realized deployed loss. Thus every selected iterate is a valid low-bit checkpoint. This procedure is parameterization-agnostic and datatype-agnostic: it can be instantiated on full weights, integrated with structured parameterizations such as LoRA, combined with joint scale optimization, and paired with different quantization datatypes.

Our contributions can be summarized as follows:

- We derive a *code-coordinate surrogate gradient* and establish its local steepest-descent interpretation under codebook continuation.
- We develop a guided candidate-sampling algorithm that combines first-order guidance with loss-based selection over deployable low-bit candidates.
- We evaluate GradCodeS across three task families, multiple quantization datatypes, and both full and structured code parameterizations, observing consistent gains in the fully low-bit setting.

2 Related Work

Continuous low-bit adaptation. Low-bit adaptation commonly retains continuous optimization. Quantized-backbone parameter-efficient methods train high-precision adapters, improve their quantized initialization, or design updates that can later be merged (Dettmers et al., 2023; Li et al., 2024; Xu et al., 2024; Loeschcke et al., 2024). They are effective when mixed-precision inference or a separate conversion stage is acceptable, but under strict fully low-bit deployment the optimized and deployed states may differ because an adapter remains at inference time or a merged update must be requantized. Quantization-aware and compressed-representation methods, including L4Q and PV-Tuning, optimize closer to the final quantized checkpoint (Jeon et al., 2025; Malinovskii et al., 2024). Their focus is efficient optimization of differentiable or structured compressed representations. We instead study how a weight gradient should be expressed in ordered code coordinates without treating the resulting continuous reference as an accepted update.

Direct low-bit optimization. This approach avoids a separate deployment conversion. QuZO and QZO use zeroth-order estimates for low-precision or quantization-aware fine-tuning, while QES applies evolution strategies to quantized parameters (Zhou et al., 2025; Shang et al., 2026; Xu et al., 2026). These methods reduce or avoid backpropagation and evaluate search information through forward perturbations or population fitness. Although candidate quality can therefore be measured in or near the quantized model family, efficient proposal generation remains challenging. in a high-dimensional space under a finite evaluation budget. GradCodeS instead spends one backward pass to

obtain an analytical local signal, uses it only to guide where discrete search looks, and still selects updates by the realized loss of deployable low-bit candidates.

Gradient-informed discrete proposals. A derivative can shape a discrete proposal without being followed deterministically. Representative samplers construct local, Langevin-style, Newton-style, or importance proposals from gradients with respect to discrete inputs (Grathwohl et al., 2021; Zhang et al., 2022; Sun et al., 2023a; Xiang et al., 2023; Liu et al., 2023); related optimization methods bias random search with surrogate gradients or adapt discrete sampling to combinatorial objectives (Maheswaranathan et al., 2019; Sun et al., 2023b; Li & Zhang, 2025). These works motivate separating proposal construction from state selection, but they do not address two challenges specific to low-bit fine-tuning. First, the available derivative is with respect to recovered weights, whereas the decision variable is a code index whose geometry is distorted by group scales and non-uniform codebook gaps. Second, our goal is loss minimization over a finite candidate set rather than preservation of a stationary distribution. GradCodeS addresses these differences through a quantization-specific code-coordinate signal, proposal-only gradient guidance, and loss-based selection of deployable states.

3 Problem Setup

Fix a target quantization datatype q, specified by its bitwidth b_q, numerical codebook $\mathcal{C}_q$, grouping structure $\mathcal{G}_q$, and admissible scale space $\mathcal{S}_q$. Its ordered codebook is

$$\mathcal{C}_q = \{c_1^{(q)}, \dots, c_{K_q}^{(q)}\}, \qquad c_1^{(q)} \le \cdots \le c_{K_q}^{(q)}, \qquad K_q \le 2^{b_q}.$$

The indices follow numerical order and need not match hardware bit-pattern order. For a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, let $[K_q] := \{1, \dots, K_q\}$ and $\mathcal{Z}_q := [K_q]^{d_{\text{out}} \times d_{\text{in}}}$. A code matrix $Z \in \mathcal{Z}_q$ selects values entrywise as $\mathcal{C}_q(Z) = [c_{Z_{ij}}^{(q)}]_{ij}$, where $\left[\psi(i,j)\right]_{ij}$ denotes the matrix whose (i,j)-th element is $\psi(i,j)$. Each entry belongs to a group $g_q(i,j) \in \mathcal{G}_q$ with scale metadata $s = \{s_g\}_{g \in \mathcal{G}_q} \in \mathcal{S}_q$. Defining $S_q(s) = [s_{g_q(i,j)}]_{ij}$, the deployed weight is

$$\widehat{W}_q(Z, s) = S_q(s) \odot \mathcal{C}_q(Z), \tag{1}$$

where $\odot$ is element-wise multiplication. This representation covers the evaluated datatypes NF4, INT4, and MXFP4; their datatype-specific components are given in Appendix B.

For a model, let $\mathcal{L}$ be the set of targeted quantized linear layers. We use Z and s for their layer-wise collections and let $\mathcal{Q}_q$ denote the product of the corresponding layer-wise feasible code and scale sets. We use $\widehat{\mathbf{W}}_q(Z, s)$ for the recovered weights. Parameters outside $\mathcal{L}$ are left unquantized.

Given an initial feasible state (Z_0, s_0), a target distribution $\mathcal{D}$, a model f, and a loss function ℓ, we formulate fully low-bit fine-tuning as

$$\min_{(Z,s) \in \mathcal{Q}_q} L(Z, s) := \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\ell\Big(f\Big(x; \widehat{\mathbf{W}}_q(Z, s) \Big), y \Big) \right]. \tag{2}$$

Unlike approaches that optimize a continuous auxiliary state and quantize only after training, (2) is defined directly over deployable states. Here Z is discrete, while s is continuous in the default setting but constrained to its admissible space. Since q is fixed, we henceforth omit datatype annotations from K_q, $c_k^{(q)}$, $\mathcal{C}_q$, $\mathcal{Z}_q$, $\mathcal{S}_q$, and S_q. Thus, for a single matrix, $\widehat{W}(Z, s) = S(s) \odot \mathcal{C}(Z)$. Bold $\widehat{\mathbf{W}}$ is reserved for the full collection of targeted weights, while L denotes the corresponding objective.

4 Method

To optimize (2), GradCodeS separates gradient guidance from discrete state selection. Section 4.1 constructs a geometry-aware code surrogate gradient, and Section 4.2 establishes its local and discrete first-order properties. Section 4.3 then incorporates this signal into a *guide–sample–evaluate–select* procedure that accepts updates according to the realized loss of deployable low-bit states.

4.1 Constructing the Code Surrogate Gradient

The ordinary weight gradient $\nabla_{\widehat{W}} L$ does not directly provide a suitable optimization direction for the discrete code indices, because group scales and local codebook gaps rescale a unit code move.

Thus in general, the direction in code space corresponding to the steepest descent direction in weight space is **not** the steepest descent direction in code space, as illustrated in Figure 2.

For a matrix A, write $A_+ = [\max\{A_{ij}, 0\}]_{ij}$ and $A_- = [\max\{-A_{ij}, 0\}]_{ij}$. Let $Z^{\pm} = \mathrm{clip}(Z \pm \mathbb{1}, 1, K)$ and define the forward and backward gaps

$$G^+(Z) = \left[c_{Z^+_{ij}} - c_{Z_{ij}}\right]_{ij}, \quad G^-(Z) \quad = \left[c_{Z_{ij}} - c_{Z^-_{ij}}\right]_{ij}.$$

The quantities above characterize the local geometry of the two neighboring code moves. To use a weight gradient for discrete search, we need a code-coordinate signal that reflects this geometry; Definition 1 constructs such a signal.

Definition 1 (Local effective step and code surrogate gradient)**.** *Given a quantized state (Z, s) with fixed codebook $\mathcal{C}$, let $\sigma = \mathrm{sign}(\nabla_{\widehat{W}} L(Z, s))$ and define the descent-aligned effective step by $D\big(Z, s, \nabla_{\widehat{W}} L(Z, s)\big) = S(s) \odot \big(G^+(Z) \odot \sigma_- + G^-(Z) \odot \sigma_+\big)$. The directional code surrogate gradient is then*

$$\nabla_Z L(Z, s) = \nabla_{\widehat{W}} L(Z, s) \odot D\big(Z, s, \nabla_{\widehat{W}} L(Z, s)\big). \tag{3}$$

Figure 2 gives a simplified illustration of Definition 1. It uses an identity codebook, whose adjacent gaps are all one, so the effective step reduces to $D = S(s)$. The example therefore shows that even without nonuniform codebook gaps, accounting for the deployed-weight change of each code coordinate yields a substantially better descent direction. Appendix C derives this construction from the local piecewise-linear continuation.

Let $\mathcal{C}(Z) = Z$, $S(s) = (0.1, 10)$, $L(\hat{W}) = \hat{W}_1 + 0.6\hat{W}_2$. $\nabla_{\widehat{W}} L = (1, 0.6)$, so mapped code move (to achieve $\nabla_{\widehat{W}} L$) is $(-10, -0.06)$; while $D = S(s)$ and $\nabla_Z L = (0.1, 6)$. For an equal code-space step $\Delta Z = \epsilon d$ with $\|d\|_2 = 1$:

Direction	Unit direction d	Loss change ΔL
Mapped weight gradient	$(-10, -0.06)/\sqrt{100.0036}$	-0.136ϵ
Definition 1 surrogate	$(-0.1, -6)/\sqrt{36.01}$	-6.001ϵ

Result. The effective-step correction yields about $44\times$ more loss decrease for the same code-space step.

Figure 2: A two-coordinate illustration of the effective-step correction in Definition 1.

4.2 First-Order Properties

Lemma 1 gives the surrogate a local continuous interpretation. Proposition 1 establishes exact first-order consistency for feasible one-level code moves. Corollary 1 then converts this identity into a realized-loss bound used by the candidate-search analysis in Proposition 2.

Lemma 1 (Local steepest-descent interpretation)**.** *Let $D_{Z,s} := D\big(Z, s, \nabla_{\widehat{W}} L(Z, s)\big)$. Define the local directional surrogate function of $L(Z, s)$ by extending it to real-valued interpolation $\Delta Z \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ by*

$$\bar{L}_{Z,s}(\Delta Z) = L\Big(\widehat{W}(Z, s) + D_{Z,s} \odot \Delta Z\Big).$$

Then $\nabla_{\Delta Z} \bar{L}_{Z,s}(\Delta Z)\big|_{\Delta Z = 0} = \nabla_Z L(Z, s)$. Consequently, whenever $\nabla_Z L(Z, s) \neq 0$, the direction $-\nabla_Z L(Z, s)$ is the steepest descent direction of $\bar{L}_{Z,s}$ at $\Delta Z = 0$ under the Frobenius norm.

Lemma 1 gives $-\nabla_Z L$ a precise but deliberately local interpretation: it is the steepest direction of the active continuous surrogate, not an automatically valid code update. We next connect its score to an actual one-level code move.

Proposition 1 (One-step consistency)**.** *Fix the scales s. Let $\Delta Z \in \{-1, 0, 1\}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ be a feasible one-step code move such that $Z + \Delta Z \in \mathcal{Z}$, $\Delta Z \odot \nabla_{\widehat{W}} L(Z, s) \leq 0$ entrywise. Then the surrogate linear term matches the exact first-order deployed-weight change:*

$$\langle \nabla_Z L(Z, s), \Delta Z \rangle_F = \Big\langle \nabla_{\widehat{W}} L(Z, s), \widehat{W}(Z + \Delta Z, s) - \widehat{W}(Z, s) \Big\rangle_F. \tag{4}$$

Algorithm 1 GradCodeS: Guide–Sample–Evaluate–Select in Quantized Code Space

Input: state (Z, s); codebook $\mathcal{C}$; data $\mathcal{D}$; feasible sets $(\mathcal{Z}, \mathcal{S})$; step sizes (η_Z, η_s); proposal p; candidates M; iterations T.
for $t = 1, \ldots, T$ **do**
  Sample mini-batch $\mathcal{B}_t \sim \mathcal{D}$
  Update scales by $s \leftarrow \Pi_{\mathcal{S}}(s - \eta_s \nabla_s \widehat{L}_{\mathcal{B}_t}(Z, s))$
  Guide: compute $\nabla_Z \widehat{L}_{\mathcal{B}_t}(Z, s)$ using (3), then Z^{ref} using (5)
  Sample: draw M code candidates $\{Z^m\}_{m=1}^M$ from $p(\cdot \mid Z^{\mathrm{ref}})$
  Evaluate: compute $\widehat{L}_{\mathcal{B}_t}(Z^m, s)$ for each candidate
  Select: $Z \leftarrow \arg\min_{\widetilde{Z} \in \{Z, Z^1, \ldots, Z^M\}} \widehat{L}_{\mathcal{B}_t}(\widetilde{Z}, s)$
end for
Return: $\widehat{W}(Z, s) = S(s) \odot \mathcal{C}(Z)$

Proposition 1 provides the exact first-order consistency: by selecting the adjacent gap in the descent direction, the surrogate score exactly matches the first-order deployed-weight change of every feasible gradient-aligned one-level move. This identity motivates the guidance used to construct proposals; sampled candidates need not be one-level or gradient-aligned, and are selected by their evaluated quantized objective. The next result bounds the discrepancy between the first-order prediction and the realized loss for moves satisfying the proposition.

Corollary 1 (One-step smooth-loss upper bound)**.** *Under the assumptions of Proposition 1, if L is β-smooth as a function of the deployed weight in a neighborhood of $\widehat{W}(Z, s)$, then*

$$L(Z + \Delta Z, s) \le L(Z, s) + \langle \nabla_Z L(Z, s), \Delta Z \rangle_F + \frac{\beta}{2} \left\| \widehat{W}(Z + \Delta Z, s) - \widehat{W}(Z, s) \right\|_F^2 .$$

Corollary 1 bounds the realized loss change by the surrogate prediction plus a curvature penalty; a move is guaranteed to decrease the loss when its negative surrogate term dominates this penalty. Thus the signal is first-order faithful for feasible local moves, while candidate evaluation accounts for nonlinear and higher-order effects. This bound underlies Proposition 2; proofs are provided in Appendix D.

4.3 Gradient-Guided Discrete Optimization

The results in Section 4.2 show how to rank feasible local code moves. We now turn this signal into a complete procedure that alternates differentiable scale updates with deployment-faithful code search. The surrogate gradient guides where to search, while the deployed loss determines what to accept.

Performing gradient descent with the code surrogate gradient (3) gives a real-valued reference point in code space:

$$Z^{\mathrm{ref}} = Z - \eta_Z \nabla_Z L(Z, s), \tag{5}$$

where $\eta_Z > 0$ is the step size. Because Z^{ref} is generally non-integer, it only centers a proposal over valid codes. The *guide–sample–evaluate–select* pipeline draws $M \ll |\mathcal{Z}|$ candidates with greater mass near Z^{ref} and retains the lowest-loss state. Appendix E gives an efficiently sampled coordinate-factorized proposal.

In practice, $\widehat{L}_{\mathcal{B}}(Z, s)$ denotes the average loss over mini-batch $\mathcal{B}$. For a fixed state independent of $\mathcal{B}$, it is an unbiased estimator of $L(Z, s)$. Each iteration first updates the differentiable scales by projected gradient descent with Z fixed and then searches the codes at the updated scales. Algorithm 1 summarizes the procedure; $\Pi_{\mathcal{S}}$ projects onto the admissible scale space, and selection ties retain the current Z.

Proposition 2 (Guided best-of-M descent)**.** *Fix the scales s, let q_t be the proposal mass of gradient-aligned one-step feasible updates Δ satisfying $\|\Delta + \eta_Z \nabla_Z L(Z_t, s)\|_F \le c\eta_Z \|\nabla_Z L(Z_t, s)\|_F$, and suppose that Corollary 1 applies with $\|\widehat{W}(Z_t + \Delta, s) - \widehat{W}(Z_t, s)\|_F \le \kappa \|\Delta\|_F$. If $c \in [0, 1)$ and $\alpha := 1 - c - \frac{\beta \kappa^2 \eta_Z}{2}(1 + c)^2 > 0$, then selecting Z_{t+1} to minimize the full objective $L(\cdot, s)$ over M independent proposal draws and the current state gives $\mathbb{E}[L(Z_{t+1}, s) \mid Z_t] \le L(Z_t, s) - \alpha \eta_Z \big[1 - (1 - q_t)^M\big] \|\nabla_Z L(Z_t, s)\|_F^2$.*

Proposition 2 analyzes the descent speed, conditional on the proposal mass placed near the reference step. It concerns one code-search substep at fixed scales with full-batch. In training, guidance and selection use the same mini-batch, and scales are updated separately. Keeping the current code state among the candidates ensures that the evaluated mini-batch selection objective does not increase during code search, relative to its value after the scale update. The factor $1 - (1 - q_t)^M$ is the probability of sampling at least one update in the specified set. A positive descent term requires $q_t > 0$ and a nonzero surrogate gradient; the proposal construction alone does not guarantee $q_t > 0$. Appendix F gives the proof and discusses this condition.

Operational distinction. Continuous or STE-style approaches use a gradient to produce an update that is retained, projected, or requantized, whereas zeroth-order and evolutionary approaches evaluate discrete candidates directly but rely on sampled losses to determine where to search. GradCodeS separates these roles: a quantization-aware backward signal shapes a proposal over valid codes, while realized discrete losses select the accepted deployable state. Section 5 evaluates the two practical consequences of this separation: deployment-faithful performance and more effective candidate search.

5 Experiments

We evaluate GradCodeS to address three questions: whether GradCodeS improves performance while remaining in the fully 4-bit deployment space; whether the results are robust across data regimes, code parameterizations, and quantization datatypes; and whether gradient guidance improves candidate quality and end-to-end search efficiency.

5.1 Experimental Settings

We evaluate Qwen3-0.6B, Llama-3.2-1B and 3B-Instruct on GSM8K, AlpacaEval, and MASSIVE en-US, covering arithmetic reasoning, instruction following, and structured semantic parsing. We quantize all transformer-block linear layers while leaving the token embedding and LM head unquantized. Under matched data and evaluation protocols, we compare unquantized references (Base-16 and 16-bit SFT), mixed-precision adapter methods (QLoRA-16A and QA-LoRA), and fully 4-bit baselines (Base-4, QLoRA-4Merge, LoQT, L4Q, PV-Tuning, QuZO, QZO, and QES). We evaluate both full-matrix and low-rank code parameterizations of GradCodeS; complete dataset, metric, quantization, and baseline details are provided in Appendix A. All results are obtained with three independent runs with different random seeds, reported in mean_{std}.

5.2 Main Results

Tables 1 and 2 report results across arithmetic reasoning, instruction following, and structured semantic parsing. Across all backbones and all three tasks, GradCodeS is the strongest fully 4-bit method. Direct 4-bit quantization causes a substantial drop, and the competing fully 4-bit methods recover this loss to varying degrees; GradCodeS provides the most consistent recovery across the reported metrics. We omit several baselines on the 3B model because of computational constraints.

The comparison between QLoRA-16A and QLoRA-4Merge isolates the deployment-state mismatch. QLoRA-16A retains a high-precision adapter at inference time, whereas QLoRA-4Merge collapses that adapter into the backbone and requantizes the resulting weights. The degradation after this conversion, also highlighted in Figure 1(b), supports deployment-faithful optimization rather than relying on an optimize-then-quantize pipeline. Within the fully 4-bit regime, both GradCodeS parameterizations are effective: the LoRA variant and the full-matrix variant remain competitive. We examine their dependence on training-set size in Section 5.3.

Compared with 16-bit SFT, the best GradCodeS variant obtains higher GSM8K accuracy and stronger MASSIVE results, while the 16-bit reference remains strongest on Qwen AlpacaEval and is marginally higher on Llama AlpacaEval. We hypothesize that the effect may arise from low-bit codes as regularization in Section 5.3, not as a general claim that lower precision is universally superior.

Deployment cost view. Figure 1(b) complements the task tables by plotting Llama-3.2-1B-Instruct GSM8K accuracy across deployment regimes. The bit accounting applies to the targeted quantized

modules and excludes modules intentionally left unquantized. GradCodeS attains the best accuracy on the fully 4-bit regime, whereas mixed-precision methods retain additional high-precision adapter parameters.

Table 1: Results on GSM8K and AlpacaEval. The second column reports the final deployment regime: 16 denotes unquantized 16-bit deployment, Mixed denotes a 4-bit backbone with high-precision adapter parameters, and 4 denotes fully 4-bit deployment for the quantized modules.

Method	Deployment bits	Qwen3-0.6B		Llama-3.2-1B-Inst.		Llama-3.2-3B-Inst.	
		GSM8K Acc.	Alpaca WR	GSM8K Acc.	Alpaca WR	GSM8K Acc.	Alpaca WR
Base-16	16	$37.45_{0.00}$	*base*	$36.32_{0.00}$	*base*	$56.94_{0.00}$	*base*
16-bit SFT	16	$39.65_{0.54}$	$\mathbf{88.26}_{2.16}$	$38.93_{0.63}$	$\mathbf{56.64}_{2.36}$	$\mathbf{67.10}_{0.28}$	$\mathbf{78.57}_{2.86}$
QLoRA-16A	Mixed (4.08)	$39.57_{0.47}$	$54.90_{2.87}$	$38.21_{0.51}$	$53.89_{2.74}$	$65.05_{0.39}$	$68.26_{3.12}$
QA-LoRA	Mixed (4.05)	$36.13_{0.62}$	$53.34_{3.12}$	$34.15_{0.74}$	$49.23_{3.18}$	-	-
Base-4	**4**	$27.14_{0.00}$	$26.58_{1.94}$	$25.85_{0.00}$	$46.16_{2.42}$	$49.13_{0.00}$	$35.94_{2.68}$
QLoRA-4Merge	**4**	$26.46_{0.71}$	$38.35_{2.76}$	$24.79_{0.66}$	$33.68_{3.55}$	$49.13_{0.52}$	$36.20_{3.07}$
LoQT	**4**	$30.94_{0.58}$	$44.67_{2.63}$	$29.04_{0.61}$	$48.85_{2.81}$	-	-
L4Q	**4**	$29.89_{0.67}$	$40.28_{2.91}$	$31.06_{0.57}$	$51.21_{2.66}$	-	-
PV-Tuning	**4**	$38.70_{0.49}$	$70.51_{3.21}$	$36.92_{0.45}$	$52.63_{1.84}$	$63.08_{0.43}$	$62.36_{2.79}$
QuZO	**4**	$29.45_{0.84}$	$34.35_{3.47}$	$26.54_{0.78}$	$41.57_{3.26}$	-	-
QZO	**4**	$29.76_{0.76}$	$38.99_{3.14}$	$27.60_{0.81}$	$39.39_{3.41}$	-	-
QES	**4**	$28.80_{0.92}$	$41.13_{3.36}$	$26.39_{0.85}$	$38.12_{3.67}$	-	-
GradCodeS (LoRA)	**4**	$\mathbf{45.72}_{0.42}$	$82.24_{2.57}$	$\underline{41.50}_{0.47}$	$54.66_{1.83}$	$\underline{67.08}_{0.31}$	$69.52_{3.26}$
GradCodeS (Full)	**4**	$\underline{43.68}_{0.38}$	$\underline{83.09}_{2.48}$	$\mathbf{41.63}_{0.58}$	$\underline{56.34}_{2.31}$	$66.39_{0.35}$	$\underline{72.04}_{3.14}$

Structured prediction. Table 2 evaluates a structured semantic parsing setting. GradCodeS gives the best fully 4-bit results on both backbones and all three metrics. Exact match requires the entire semantic frame to be correct; Intent accuracy requires if the important Intent slot is correct; Slot F1 measures the quality of the predicted slot structure. The gains on these metrics indicate that deployment-faithful code updates preserve structured output behavior after quantization.

Table 2: MASSIVE en-US test results. Models are fine-tuned on the English-US training split and evaluated on the English-US test split.

Method	Deployment bits	Qwen3-0.6B			Llama-3.2-1B-Inst.			Llama-3.2-3B-Inst.		
		EM	Intent	Slot F1	EM	Intent	Slot F1	EM	Intent	Slot F1
Base-16	16	$0.00_{0.00}$	$0.00_{0.00}$	$3.38_{0.00}$	$0.00_{0.00}$	$0.13_{0.00}$	$1.39_{0.00}$	$0.07_{0.00}$	$3.03_{0.00}$	$3.63_{0.00}$
16-bit SFT	16	$57.92_{0.36}$	$83.95_{0.22}$	$65.31_{0.44}$	$63.06_{0.41}$	$85.57_{0.29}$	$73.39_{0.38}$	$66.16_{0.33}$	$\underline{88.97}_{0.24}$	$75.21_{0.35}$
QLoRA-16A	Mixed (4.08)	$57.16_{0.42}$	$83.49_{0.31}$	$66.74_{0.48}$	$63.39_{0.37}$	$85.81_{0.26}$	$74.13_{0.43}$	$65.23_{0.39}$	$88.06_{0.28}$	$75.14_{0.41}$
QA-LoRA	Mixed (4.05)	$50.17_{0.58}$	$80.77_{0.47}$	$61.19_{0.62}$	$54.10_{0.55}$	$84.30_{0.35}$	$65.41_{0.66}$	-	-	-
Base-4	**4**	$0.00_{0.00}$	$0.61_{0.00}$	$3.97_{0.00}$	$0.00_{0.00}$	$0.03_{0.00}$	$0.62_{0.00}$	$0.00_{0.00}$	$1.95_{0.00}$	$3.32_{0.00}$
QLoRA-4Merge	**4**	$0.00_{0.00}$	$0.67_{0.11}$	$3.61_{0.18}$	$0.00_{0.00}$	$0.03_{0.02}$	$0.48_{0.14}$	$0.00_{0.00}$	$1.95_{0.13}$	$3.29_{0.21}$
LoQT	**4**	$51.42_{0.63}$	$72.09_{0.51}$	$45.83_{0.78}$	$52.14_{0.59}$	$75.27_{0.46}$	$49.90_{0.82}$	-	-	-
L4Q	**4**	$48.27_{0.71}$	$72.03_{0.56}$	$46.02_{0.74}$	$53.43_{0.64}$	$75.88_{0.49}$	$58.62_{0.69}$	-	-	-
PV-Tuning	**4**	$58.71_{0.45}$	$82.92_{0.37}$	$66.29_{0.53}$	$64.50_{0.38}$	$86.31_{0.31}$	$74.88_{0.46}$	$65.89_{0.35}$	$86.95_{0.27}$	$76.20_{0.42}$
QuZO	**4**	$31.26_{0.86}$	$57.30_{0.72}$	$38.55_{0.91}$	$46.35_{0.77}$	$64.17_{0.68}$	$46.80_{0.88}$	-	-	-
QZO	**4**	$30.06_{0.79}$	$59.79_{0.64}$	$42.50_{0.83}$	$45.18_{0.81}$	$62.50_{0.74}$	$41.33_{0.95}$	-	-	-
QES	**4**	$29.25_{0.93}$	$52.53_{0.81}$	$34.41_{1.02}$	$43.56_{0.85}$	$58.35_{0.79}$	$39.81_{0.97}$	-	-	-
GradCodeS (LoRA)	**4**	$\underline{65.88}_{0.34}$	$\underline{85.53}_{0.28}$	$\mathbf{76.60}_{0.39}$	$\underline{67.37}_{0.31}$	$86.89_{0.33}$	$\underline{76.25}_{0.37}$	$\mathbf{70.48}_{0.27}$	$\mathbf{90.97}_{0.19}$	$\mathbf{79.85}_{0.31}$
GradCodeS (Full)	**4**	$\mathbf{66.61}_{0.32}$	$\mathbf{86.85}_{0.24}$	$\underline{75.88}_{0.41}$	$\mathbf{69.17}_{0.28}$	$\mathbf{88.13}_{0.21}$	$\mathbf{78.74}_{0.34}$	$\underline{67.26}_{0.29}$	$87.53_{0.25}$	$\underline{78.24}_{0.36}$

5.3 Robustness across Data, Parameterizations, and Datatypes

Data size and code parameterization. The comparison with 16-bit SFT suggests that the restricted code-space parameterization may provide a regularization effect when fine-tuning data are limited. We treat this as an empirical hypothesis rather than a consequence of the local optimization theory. Table 3 examines the hypothesis by comparing the original 7.0k-example GSM8K training set with a 167k-example training set formed by adding 160k MetaMathQA examples (Yu et al., 2024).

Table 3: GSM8K test accuracy (%) across training-set sizes and parameterizations on Llama-3.2-1B-Instruct. Values are means over three independent runs, with standard deviations shown as subscripts.

Method	Parameterization	7.0k examples	167k examples
16-bit SFT	LoRA (rank 8)	$\underline{38.9}_{0.6}$	$\mathbf{55.9}_{0.7}$
GradCodeS	LoRA (rank 8)	$41.5_{0.6}$	$53.3_{0.6}$
	Full matrix	$\mathbf{41.6}_{0.5}$	$\underline{54.5}_{0.5}$

With 7.0k examples, GradCodeS (Full) performs best; with 167k examples, 16-bit SFT becomes strongest. This result is consistent with that a regularization benefit from the low-bit code parameterization in the smaller-data regime.

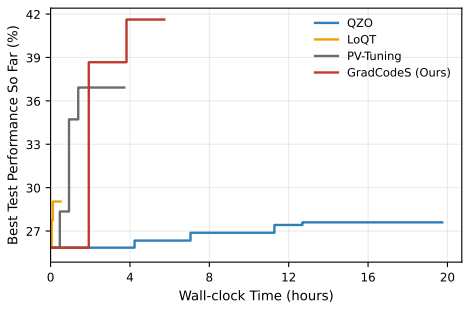
Quantization datatype. Because (1) provides a unified datatype-specific recovery map, GradCodeS does not rely on a single quantization datatype. Different datatypes change the codebook $\mathcal{C}$ and scale grouping, while the search rule requires only ordered code levels and the local effective-step map in Definition 1. Table 4 shows that GradCodeS can be applied to NF4, INT4, and MXFP4. NF4 and MXFP4 are stronger than INT4 in this setting, and all three variants remain competitive with the adaptive PV-Tuning baseline. These results establish compatibility across the tested datatype-specific geometries covering different codebook property.

Table 4: Results of GradCodeS across quantization datatypes on Llama-3.2-1B-Instruct. Values are means over three independent runs, with standard deviations shown as subscripts.

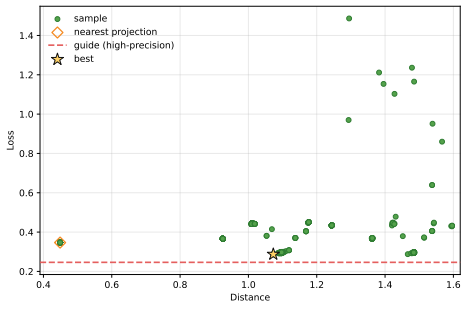
Method	Datatype	GSM8K ACC.	AlpacaEval WR.	MASSIVE EM.
GradCodeS	NF4	$\mathbf{41.6}_{0.5}$	$56.3_{2.3}$	$\mathbf{69.2}_{0.3}$
	INT4	$38.9_{0.6}$	$52.0_{2.4}$	$65.1_{0.5}$
	MXFP4	$41.1_{0.4}$	$\mathbf{57.1}_{2.2}$	$68.8_{0.4}$
PV-Tuning	Adaptive	$36.9_{0.5}$	$52.6_{1.8}$	$64.5_{0.4}$

5.4 Search Efficiency and Component Analysis

End-to-end efficiency. Figure 3(a) compares wall-clock time against the best test accuracy reached so far. GradCodeS is not the cheapest possible step: it uses one backward pass to build the guided reference and then evaluates multiple discrete candidates. The gradient guide concentrates candidate evaluations around the reference point. The baselines improve quickly but plateau below the final GradCodeS accuracy, while GradCodeS reaches a stronger result within the displayed time range. This supports the intended trade-off of spending one gradient computation to reduce wasted discrete search.



(a) Wall-clock time versus the best test accuracy reached so far, evaluated once per epoch.



(b) Candidate loss versus distance from the guided reference in one sampling step.

Figure 3: Training efficiency and candidate-set quality for Llama-3.2-1B-Instruct on GSM8K.

Candidate-set quality. Figure 3(b) gives a candidate-level view of one illustrative sampling step. Candidate loss tends to be lower near the guided reference, but the loss is not monotonic in distance from the reference. This is why GradCodeS does not simply project to the nearest code point: it keeps

several nearby candidates reachable and lets their realized discrete losses determine the accepted update.

Table 5 reports the empirical selection probability and selected mean loss under the same candidate budget. Candidates drawn from the guided proposal are selected more often than Gaussian or one-hop sampling, indicating that gradient guidance improves candidate-set quality rather than merely increasing the number of forward evaluations. Guidance using the proposed surrogate ($\nabla_Z L$) produces higher-quality candidates than mapped ordinary weight gradient ($\nabla_{\hat{W}} L$)-guided, indicating it is a more effective first order signal. Together, Figure 3(b) and Table 5 are consistent with the proposal-guidance role analyzed in Proposition 2, without directly testing its candidate-count dependence.

Table 5: Candidate-selection statistics for different sampling distributions under the same candidate budget. Values are means over three independent runs, with standard deviations shown as subscripts.

Sampling	**Sel. Prob.**	**Sel. Mean Loss**
Gaussian	5.83%	0.2955
1-Hop Neighbor	38.16%	0.2963
$\nabla_{\hat{W}} L$-Guided	48.62%	0.2951
$\nabla_Z L$-Guided (Ours)	67.97%	0.2949

Component ablations. We ablate three components of the default GradCodeS configuration: gradient guidance, code-index updates, and scale updates. The *w/o guide* variant samples uniformly from a neighborhood of the current code without conditioning on (5). The $\nabla_{\hat{W}} L$*-guide* replaces code surrogate gradient with mapped ordinary weight gradient. The *w/o scale update* variant freezes group-wise scales and optimizes only codes, while the *w/o code update* variant freezes the indices and optimizes only group-wise scales.

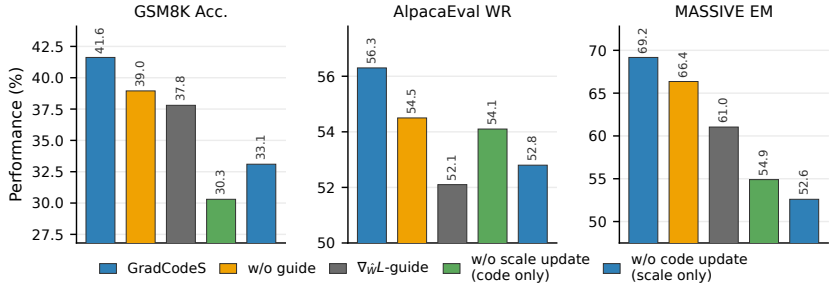


Figure 4: Component ablations for GradCodeS on Llama-3.2-1B-Instruct.

Figure 4 shows that all three components contribute. Both removing gradient guidance and replacing $\nabla_Z L$ signal with $\nabla_{\hat{W}} L$ consistently reduce performance, supporting the use of the code surrogate gradient to shape the proposal. Freezing either codes or scales causes larger task-dependent drops, indicating that both variables are important for the performance of the full method.

6 Conclusion and Limitations

We introduced GradCodeS, a method for fine-tuning low-bit models directly in quantized code space. The method uses a code-space surrogate gradient to guide candidate generation, while the accepted update is chosen by evaluating valid quantized states. This separates first-order guidance from discrete state selection and ensures that every accepted code update corresponds to a deployable low-bit model. Our analysis shows that the proposed surrogate matches the first-order change in deployed weights for feasible, gradient-aligned one-level code moves. Empirically, GradCodeS improves fully 4-bit fine-tuning across arithmetic reasoning, instruction following, and structured prediction, and remains effective with NF4, INT4, and MXFP4 quantization. The ablations also show that code-space guidance, code updates, and scale updates each contribute to performance.

The current results are limited to models up to 3B parameters, and the theoretical guarantees apply to local code moves under specific conditions. Evaluating the method on larger models and developing proposal distributions with stronger guarantees are therefore natural directions for future work.

7 AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs and implementing methods, providing feedback on research methodology and experiments, and proposing or refining hypotheses. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting with translation, supporting qualitative and thematic data analysis, interpreting results. And generating synthetic data sets, cleaning and reformatting dataset are not applicable to this work. We have reviewed all AI-assisted work. We checked LLM-generated proofs for potential mathematical error. LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

Code is provided at `https://anonymous.4open.science/r/GradCodes-EC82`, containing complete implementation details for reproducibility.

REFERENCES

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023.

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

Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris J. Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 3831–3841, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Hyesung Jeon, Yulhwa Kim, and Jae-Joon Kim. L4Q: Parameter efficient quantization-aware fine-tuning on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2002–2024, 2025.

Muheng Li and Ruqi Zhang. Reheated gradient-based discrete sampling for combinatorial optimization. *Transactions on Machine Learning Research*, 2025.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024.

Meng Liu, Haoran Liu, and Shuiwang Ji. Gradient-guided importance sampling for learning binary energy-based models. In *International Conference on Learning Representations*, 2023.

Sebastian Loeschcke, Mads Toftrup, Michael J. Kastoryano, Serge Belongie, and Vésteinn Snæbjarnarson. LoQT: Low-rank adapters for quantized pretraining. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115282–115308, 2024.

Niru Maheswaranathan, Luke Metz, George Tucker, Dami Choi, and Jascha Sohl-Dickstein. Guided evolutionary strategies: Augmenting random search with surrogate gradients. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 4264–4273, 2019.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtárik. PV-Tuning: Beyond straight-through estimation for extreme LLM compression. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5074–5121, 2024.

Sifeng Shang, Jiayi Zhou, Chenyu Lin, Minxian Li, and Kaiyang Zhou. Fine-tuning quantized neural networks with zeroth-order optimization. In *International Conference on Learning Representations*, 2026.

Haoran Sun, Hanjun Dai, Bo Dai, Haomin Zhou, and Dale Schuurmans. Discrete langevin samplers via wasserstein gradient flow. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pp. 6290–6313, 2023a.

Haoran Sun, Katayoon Goshvadi, Azade Nova, Dale Schuurmans, and Hanjun Dai. Revisiting sampling for combinatorial optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 32859–32874, 2023b.

Yue Xiang, Dongyao Zhu, Bowen Lei, Dongkuan Xu, and Ruqi Zhang. Efficient informed proposals for discrete distributions via Newton's series approximation. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pp. 7288–7310, 2023.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 38087–38099, 2023.

Yinggan Xu, Risto Miikkulainen, and Xin Qiu. Quantized evolution strategies: High-precision fine-tuning of quantized LLMs at low-precision cost. *arXiv preprint arXiv:2602.03120*, 2026.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

Ruqi Zhang, Xingchao Liu, and Qiang Liu. A Langevin-like sampler for discrete distributions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 26375–26396, 2022.

Jiajun Zhou, Yifan Yang, Kai Zhen, Ziyue Liu, Yequan Zhao, Ershad Banijamali, Athanasios Mouchtaris, Ngai Wong, and Zheng Zhang. QuZO: Quantized zeroth-order fine-tuning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5341–5359, 2025.

A Experimental Details

A.1 Models and Evaluation Tasks

Models. We evaluate three instruction-tuned decoder-only language models, Qwen3-0.6B, Llama-3.2-1B-Instruct, and Llama-3.2-3B-Instruct. Compact backbones are a useful stress test for fully low-bit adaptation: they are realistic for mobile, edge, and other memory-constrained deployment scenarios, and they make it easier to attribute performance differences to the optimization method rather than to very large model capacity. For each backbone, we compare the original 16-bit checkpoint, the directly quantized 4-bit checkpoint, mixed-precision adapter baselines, and fully 4-bit fine-tuned variants under the same data and evaluation protocol.

Datasets and metrics. We evaluate on three datasets covering arithmetic reasoning, general instruction following, and mobile-style language understanding. For **GSM8K**, models are fine-tuned on the official training split and evaluated on the official test split; we report answer accuracy after extracting the final numerical answer. For **Alpaca**, models are fine-tuned on Alpaca Cleaned and evaluated with AlpacaEval; we report win rate under the same evaluator for all methods. For **MASSIVE en-US**, models are fine-tuned on the English-US training split and evaluated on the English-US test split. We formulate MASSIVE as structured semantic parsing and report Exact Match, Intent Accuracy, and Slot F1.

A.2 Quantization Protocol

Following the block-wise setup in Section 3, we quantize all linear parameters in the transformer blocks, including the attention and MLP projection matrices in every layer, while leaving the token embedding and LM head unquantized, following common LLM quantization practice (Dettmers et al., 2023; Frantar et al., 2023; Lin et al., 2024). We use the same target modules, quantization format, training data, and evaluation scripts whenever applicable. The reported 4-bit scores reflect the deployed quantized checkpoint rather than the relaxed training state.

A.3 Implementation Details

Model parameterization. The implementation replaces the attention projections `q_proj`, `k_proj`, `v_proj`, and `o_proj`, and the MLP projections `gate_proj`, `up_proj`, and `down_proj`, with GradCodeS layers. The default quantization format is 4-bit NF4. NF4 and INT4 use group size 64, while MXFP4 uses group size 32. We evaluate both full-matrix search and low-rank code search. The latter uses one rank-8 stage with integer factors A and B, initialized by a random integer A and zero B; their product is merged into the code matrix, rather than retained as an inference-time adapter. Full-matrix search uses the same guide–sample–evaluate–select rule directly on the code coordinates.

Training configuration. The default search batch size is one example per device and eight devices. Models are loaded in bfloat16, temporary recovered-weight gradients used to construct the code-space guide are accumulated in float32, and gradient checkpointing is enabled. We use a maximum sequence length of 768 and optimize a response-only causal language-modeling objective: prompt and padding tokens are masked, while response tokens and the final EOS token contribute to the loss.

Guided proposal and candidate selection. At every search step, the recovered-weight gradient is multiplied by the direction-specific effective quantization step in Definition 1 to obtain the code surrogate gradient. In the low-rank variant, the resulting factor references use constant proposal step sizes $\eta_A = \eta_B = 3$. Candidate factor coordinates are restricted to local integer windows of radius one, and the proposal step has an ℓ_∞ cap of 0.5. Within these windows, coordinates are sampled independently from the factorized inverse-distance distribution in Appendix E, using smoothing constant 10^{-8}. The proposal temperature is initialized to $\tau = 2$; in the default adaptive mode it is recalibrated at each step so that the joint probability of the current state is 0.5.

Each step evaluates 16 discrete states: the current state, the nearest state to the guided reference, and 14 samples from the proposal. We materialize the corresponding quantized weights, evaluate their realized mini-batch objective, and retain the lowest-loss state; ties keep the current state. The M in

Proposition 2 counts only the 14 independent random draws. Under full-objective selection at fixed scales, adding the nearest-reference candidate cannot weaken the bound. The selection objective uses an L2 regularization coefficient of 0.5. Quantization scales are updated alternately with code search using AdamW with learning rate 5×10^{-5}, zero weight decay, $(\beta_1, \beta_2) = (0.9, 0.999)$, and $\epsilon = 10^{-8}$; scale-gradient clipping is disabled.

Evaluation and reporting. GSM8K accuracy is computed by extracting the final numerical answer. MASSIVE uses greedy decoding with batch size 4 and at most 128 new tokens, and generations are parsed into normalized JSON semantic frames for Exact Match, Intent Accuracy, and micro-averaged Slot F1. AlpacaEval responses are evaluated to win/tie/lose by GPT-5.2; AlpacaEval win rate is computed with the same evaluator configuration for all methods. We report the mean and standard deviation over three independent runs.

A.4 Baselines

We compare with three groups of baselines. A method is considered **fully low-bit** deployable only if inference does not require high-precision adapters or residual branches for the quantized modules. A method is considered **mixed** if it uses a quantized backbone but keeps high-precision inference-time adapter parameters. We also include **unquantized** references to separate adaptation quality from deployment cost.

Unquantized references.

- **Base-16** evaluates the original 16-bit checkpoint without task fine-tuning.
- **16-bit SFT** fine-tunes 16-bit LoRA adapters on the 16-bit backbone and serves as a high-precision parameter-efficient adaptation reference (Hu et al., 2022).

Mixed-precision baselines.

- **QLoRA-16A** freezes a 4-bit backbone and fine-tunes a 16-bit LoRA adapter, giving a strong low-memory training baseline whose deployed model still contains high-precision adapter weights (Dettmers et al., 2023).
- **QA-LoRA** uses quantization-aware group-wise low-rank adapters to improve compatibility between LoRA adaptation and quantized deployment, but optimizes continuous adapter variables (Xu et al., 2024).

Fully 4-bit baselines.

- **Base-4** directly evaluates the quantized 4-bit checkpoint before adaptation.
- **QLoRA-4Merge** follows QLoRA training, then merges the learned adapter into the backbone and requantizes the merged weights to 4-bit for inference (Dettmers et al., 2023).
- **LoQT** trains low-rank adapters for quantized weights and periodically merges them into quantized full-rank weights; we evaluate its final merged-and-requantized checkpoint (Loeschcke et al., 2024).
- **L4Q** is a representative projection/STE-style low-bit fine-tuning baseline (Jeon et al., 2025).
- **PV-Tuning** optimizes a compressed representation and alternates quantized variables with auxiliary continuous quantities (Malinovskii et al., 2024).
- **QuZO**, **QZO**, and **QES** represent deployment-faithful discrete or zeroth-order low-bit optimization baselines (Zhou et al., 2025; Shang et al., 2026; Xu et al., 2026).

GradCodeS searches over low-bit code states directly. Every selected update is a deployable low-bit state, so the final model does not require a 16-bit adapter or high-precision residual branch at inference time. We evaluate both the full-matrix code update, GradCodeS (Full), and the low-rank code parameterization from Appendix G, GradCodeS (LoRA).

B Datatype-Specific Instantiations of the Unified Quantization Model

The common update rule depends on each datatype only through its ordered codebook, scale grouping, and recovery map. Table 6 summarizes these components for the evaluated datatypes.

Datatype	b_q	K_q	Codebook	Scale/group structure
NF4	4	16	Non-uniform	Group-wise scaling
INT4	4	16	Uniform	Group-wise scaling
MXFP4	4	16	E2M1-style	Block/microscaling

Table 6: Datatype-specific components used by the unified quantization representation.

For datatype q, let $\mathcal{C}_q = \{c_1^{(q)}, \ldots, c_{K_q}^{(q)}\}$, let $g_q(i,j)$ denote the scale group of entry (i,j), and let $S_q(s) = [s_{g_q(i,j)}]_{ij}$. The deployed weight is

$$\widehat{W}_q(Z,s) = S_q(s) \odot \mathcal{C}_q(Z), \tag{6}$$

with datatype-specific code and scale constraints. A feasible code move changes an entry by

$$\Delta\widehat{W}_{q,ij} = s_{g_q(i,j)}\Big(c^{(q)}_{Z_{ij}+\Delta Z_{ij}} - c^{(q)}_{Z_{ij}}\Big).$$

Thus the effective step automatically incorporates the datatype-specific codebook gap and scale. Here *datatype-agnostic* refers to the common optimization rule, not to identical feasible spaces; every candidate is still evaluated through the appropriate recovery map in (6).

C Derivation of the Code Surrogate Gradient

The chain rule would suggest $\nabla_Z L = \nabla_{\widehat{W}} L \odot \nabla_Z \widehat{W}$, but the codebook lookup is non-differentiable. For an actual code-coordinate move $\Delta Z \in \mathbb{Z}^{d_{\text{out}} \times d_{\text{in}}}$, the deployed-weight change is exactly

$$\widehat{W}(\text{clip}(Z+\Delta Z, 1, K), s) - \widehat{W}(Z,s) = S(s) \odot \Big(\mathcal{C}(\text{clip}(Z+\Delta Z,1,K)) - \mathcal{C}(Z)\Big).$$

We therefore extend $\mathcal{C}(\cdot)$ piecewise linearly around the current code, using the adjacent gaps $G^+(Z)$ and $G^-(Z)$ from Section 4.1 as directional slopes. For $\|\Delta Z\|_\infty \le 1$, this continuation satisfies

$$\widehat{W}(\text{clip}(Z+\Delta Z, 1, K), s) - \widehat{W}(Z,s) = S(s) \odot \Big(G^+(Z) \odot (\Delta Z)_+ - G^-(Z) \odot (\Delta Z)_-\Big).$$

For a descent-aligned move, a negative weight-gradient entry calls for a forward code move, whereas a positive entry calls for a backward code move. Writing $\sigma = \text{sign}(\nabla_{\widehat{W}} L)$, the corresponding nonnegative directional slope is

$$S(s) \odot \Big(G^+(Z) \odot \sigma_- + G^-(Z) \odot \sigma_+\Big),$$

which is precisely the effective step $D\big(Z, s, \nabla_{\widehat{W}} L\big)$ in Definition 1. Applying the chain rule to this local continuation yields

$$\nabla_Z L(Z,s) = \nabla_{\widehat{W}} L(Z,s) \odot D\big(Z, s, \nabla_{\widehat{W}} L(Z,s)\big),$$

as stated in (3). For a uniform codebook and interior coordinates, the adjacent gaps coincide, so D reduces to the quantization gap multiplied by $S(s)$.

D Proofs for Section 4.2

Restatement of Lemma 1. Define the local directional surrogate function of $L(Z,s)$ by extending it to real-valued interpolation $\Delta Z \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ by

$$\bar{L}_{Z,s}(\Delta Z) = L\Big(\widehat{W}(Z,s) + D\big(Z, s, \nabla_{\widehat{W}} L(Z,s)\big) \odot \Delta Z\Big).$$

Then

$$\nabla_{\Delta Z} \bar{L}_{Z,s}(\Delta Z)\big|_{\Delta Z = 0} = \nabla_Z L(Z,s).$$

Consequently, whenever $\nabla_Z L(Z,s) \neq 0$, the direction $-\nabla_Z L(Z,s)$ is the steepest descent direction of $\bar{L}_{Z,s}$ at $\Delta Z = 0$ under the Frobenius norm.

Proof of Lemma 1. By definition,

$$\bar{L}_{Z,s}(\Delta Z) = L\Big(\widehat{W}(Z,s) + D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) \odot \Delta Z\Big).$$

Since $D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big)$ is fixed at the current point, the map

$$\Delta Z \mapsto \widehat{W}(Z,s) + D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) \odot \Delta Z$$

is affine, and its differential in direction H is

$$D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) \odot H.$$

Therefore, by the chain rule,

$$\nabla_{\Delta Z}\bar{L}_{Z,s}(\Delta Z) = \nabla_{\widehat{W}}L\Big(\widehat{W}(Z,s) + D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) \odot \Delta Z\Big) \odot D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big).$$

Evaluating at $\Delta Z = 0$ gives

$$\begin{aligned}
&\nabla_{\Delta Z}\bar{L}_{Z,s}(\Delta Z)\big|_{\Delta Z=0} \\
&= \nabla_{\widehat{W}}L\big(\widehat{W}(Z,s)\big) \odot D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) \\
&= \nabla_{\widehat{W}}L(Z,s) \odot D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big) = \nabla_Z L(Z,s),
\end{aligned}$$

which proves the exact-gradient claim.

For the steepest-descent statement, consider the first-order model

$$m(\Delta Z) = \bar{L}_{Z,s}(0) + \langle \nabla_Z L(Z,s), \Delta Z\rangle_F.$$

For any ΔZ with $\|\Delta Z\|_F \le r$, Cauchy–Schwarz gives

$$\langle \nabla_Z L(Z,s), \Delta Z\rangle_F \ge -\|\nabla_Z L(Z,s)\|_F \, \|\Delta Z\|_F \ge -r\|\nabla_Z L(Z,s)\|_F.$$

Equality is attained at

$$\Delta Z = -r\,\frac{\nabla_Z L(Z,s)}{\|\nabla_Z L(Z,s)\|_F},$$

whenever $\nabla_Z L(Z,s) \neq 0$. Hence $-\nabla_Z L(Z,s)$ is the steepest descent direction of $\bar{L}_{Z,s}$ at $\Delta Z = 0$ under the Frobenius norm. □

Restatement of Proposition 1. Fix the scales s. Let $\Delta Z \in \{-1,0,1\}^{d_{\text{out}}\times d_{\text{in}}}$ be a feasible one-step code move such that

$$Z + \Delta Z \in \mathcal{Z}, \qquad \Delta Z \odot \nabla_{\widehat{W}}L(Z,s) \le 0 \quad \text{entrywise.}$$

Then the surrogate linear term matches the exact first-order deployed-weight change:

$$\langle \nabla_Z L(Z,s), \Delta Z\rangle_F = \Big\langle \nabla_{\widehat{W}}L(Z,s), \widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\Big\rangle_F.$$

Proof of Proposition 1. We first prove the exact first-order identity

$$\langle \nabla_Z L(Z,s), \Delta Z\rangle_F = \Big\langle \nabla_{\widehat{W}}L(Z,s), \widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\Big\rangle_F.$$

Fix one coordinate (i,j). If $\Delta Z_{ij} = 0$, both sides contribute zero. If $\Delta Z_{ij} = 1$, feasibility gives $Z_{ij} < K$, and gradient alignment gives $(\nabla_{\widehat{W}}L(Z,s))_{ij} \le 0$. When $(\nabla_{\widehat{W}}L(Z,s))_{ij} < 0$, the definition of $D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big)$ selects the forward gap, so

$$\begin{aligned}
\widehat{W}_{ij}(Z+\Delta Z,s) - \widehat{W}_{ij}(Z,s) &= S_{ij}(s)\big(c_{Z_{ij}+1} - c_{Z_{ij}}\big) \\
&= d^+_{ij}(Z,s) \\
&= \big[D\big(Z,s,\nabla_{\widehat{W}}L(Z,s)\big)\big]_{ij}\Delta Z_{ij}.
\end{aligned}$$

If instead $(\nabla_{\widehat{W}} L(Z,s))_{ij} = 0$, then both first-order contributions are zero. If $\Delta Z_{ij} = -1$, feasibility gives $Z_{ij} > 1$, and gradient alignment gives $(\nabla_{\widehat{W}} L(Z,s))_{ij} \geq 0$. When $(\nabla_{\widehat{W}} L(Z,s))_{ij} > 0$, the definition of $D\big(Z, s, \nabla_{\widehat{W}} L(Z,s)\big)$ selects the backward gap, so

$$\begin{aligned}\widehat{W}_{ij}(Z+\Delta Z,s) - \widehat{W}_{ij}(Z,s) &= S_{ij}(s)\big(c_{Z_{ij}-1} - c_{Z_{ij}}\big) \\ &= -d^{-}_{ij}(Z,s) \\ &= \big[D\big(Z,s,\nabla_{\widehat{W}} L(Z,s)\big)\big]_{ij}\Delta Z_{ij}.\end{aligned}$$

Again, if $(\nabla_{\widehat{W}} L(Z,s))_{ij} = 0$, both first-order contributions are zero. Therefore, for every coordinate,

$$(\nabla_{\widehat{W}} L(Z,s))_{ij}\big(\widehat{W}_{ij}(Z+\Delta Z,s) - \widehat{W}_{ij}(Z,s)\big) = (\nabla_{\widehat{W}} L(Z,s))_{ij}\big[D\big(Z,s,\nabla_{\widehat{W}} L(Z,s)\big)\big]_{ij}\Delta Z_{ij}.$$

Summing over coordinates yields

$$\begin{aligned}\Big\langle \nabla_{\widehat{W}} L(Z,s), \widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\Big\rangle_F &= \big\langle \nabla_{\widehat{W}} L(Z,s) \odot D\big(Z,s,\nabla_{\widehat{W}} L(Z,s)\big), \Delta Z\big\rangle_F \\ &= \langle \nabla_Z L(Z,s), \Delta Z\rangle_F,\end{aligned}$$

which proves (4). □

Restatement of Corollary 1. Under the assumptions of Proposition 1, if L is β-smooth as a function of the deployed weight in a neighborhood of $\widehat{W}(Z,s)$, then

$$L(Z+\Delta Z,s) \leq L(Z,s) + \langle \nabla_Z L(Z,s), \Delta Z\rangle_F + \frac{\beta}{2}\left\|\widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\right\|_F^2.$$

Proof of Corollary 1. If L is β-smooth as a function of the deployed weight in a neighborhood of $\widehat{W}(Z,s)$, the descent lemma gives

$$\begin{aligned}L\big(\widehat{W}(Z+\Delta Z,s)\big) \leq L\big(\widehat{W}(Z,s)\big) &+ \Big\langle \nabla_{\widehat{W}} L\big(\widehat{W}(Z,s)\big), \widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\Big\rangle_F \\ &+ \frac{\beta}{2}\left\|\widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\right\|_F^2.\end{aligned}$$

By Proposition 1,

$$\Big\langle \nabla_{\widehat{W}} L\big(\widehat{W}(Z,s)\big), \widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\Big\rangle_F = \langle \nabla_Z L(Z,s), \Delta Z\rangle_F.$$

Substituting this identity together with

$$L(Z,s) = L\big(\widehat{W}(Z,s)\big), \qquad L(Z+\Delta Z,s) = L\big(\widehat{W}(Z+\Delta Z,s)\big)$$

yields

$$L(Z+\Delta Z,s) \leq L(Z,s) + \langle \nabla_Z L(Z,s), \Delta Z\rangle_F + \frac{\beta}{2}\left\|\widehat{W}(Z+\Delta Z,s) - \widehat{W}(Z,s)\right\|_F^2,$$

as claimed. □

E A Concrete Factorized Candidate-Sampling Distribution

To instantiate the local candidate-sampling rule in Section 4.3, we center a finite integer lattice at the rounded guided reference.

$$\mathcal{N}_{ij}^{(\rho)}(Z^{\mathrm{ref}}) = \big\{u \in [K] \mid |u - \mathrm{round}(Z_{ij}^{\mathrm{ref}})| \leq \rho\big\},$$

where $\rho \in \mathbb{Z}^+$ is the local search radius and infeasible boundary values are removed. For efficient sampling in high-dimensional spaces, we sample each matrix entry $(i,j) \in [1, d_{\mathrm{out}}] \times [1, d_{\mathrm{in}}]$ independently from a discrete scalar distribution $p_{ij}(u \mid Z^{\mathrm{ref}})$, which jointly induces

$$p(Z^m \mid Z^{\mathrm{ref}}) = \prod_{i=1}^{d_{\mathrm{out}}} \prod_{j=1}^{d_{\mathrm{in}}} p_{ij}(Z_{ij}^m \mid Z^{\mathrm{ref}}).$$

We choose

$$p_{ij}(u \mid Z^{\mathrm{ref}}) = \frac{(|u - Z_{ij}^{\mathrm{ref}}| + \varepsilon)^{-\tau}}{\sum_{v \in \mathcal{N}_{ij}^{(\rho)}(Z^{\mathrm{ref}})} (|v - Z_{ij}^{\mathrm{ref}}| + \varepsilon)^{-\tau}}, \qquad u \in \mathcal{N}_{ij}^{(\rho)}(Z^{\mathrm{ref}}),$$

where $\varepsilon > 0$ prevents singular probabilities and $\tau > 0$ controls concentration around the reference. The implementation applies the same construction entrywise to the integer factors A and B in the low-rank variant. We use $\rho = 1$ and $\varepsilon = 10^{-8}$. Although τ is initialized to 2, the default adaptive rule chooses it at each step so that the joint probability assigned to the current state is 0.5. This normalization keeps the proposal's overall exploration level comparable as the number and geometry of active coordinates change.

F Expected Descent of the Code-Search Substep

For the proof of Proposition 2, we analyze the deterministic full-batch code-search substep of Algorithm 1 with a fixed scale s. At iteration t, define the code-space surrogate gradient and continuous reference step by

$$g_t = \nabla_Z L(Z_t, s), \qquad \bar{\Delta}_t = -\eta_Z g_t, \qquad Z_t^{\mathrm{ref}} = Z_t + \bar{\Delta}_t.$$

For a sampled candidate state Z_t^m, let

$$\Delta_t^m = Z_t^m - Z_t$$

denote its code update relative to the current state. The proof uses the following assumptions.

Assumption A1 (smoothness on admissible sampled moves). There exist constants $\beta > 0$ and $\kappa > 0$ such that, for every iteration t and every candidate update Δ considered below, Corollary 1 applies and

$$\left\| \widehat{W}(Z_t + \Delta, s) - \widehat{W}(Z_t, s) \right\|_F \le \kappa \|\Delta\|_F. \tag{7}$$

Assumption A2 (near-reference proposal mass). Fix some $c \in [0, 1)$. For each iteration t, define the good-update set

$$\mathcal{G}_t = \left\{ \Delta \;\middle|\; \Delta \text{ is gradient-aligned and one-step feasible, and } \|\Delta + \eta_Z g_t\|_F \le c\, \eta_Z \|g_t\|_F \right\}.$$

Let q_t denote the total proposal mass of candidate states whose update belongs to $\mathcal{G}_t$:

$$q_t = \sum\nolimits_{Z' \,:\, Z' - Z_t \in \mathcal{G}_t} p(Z' \mid Z_t^{\mathrm{ref}}).$$

The set $\mathcal{G}_t$ can be empty even for a nonzero gradient. For example, if $g_t \neq 0$ and $\eta_Z \|g_t\|_F < 1/(1+c)$, every nonzero integer update satisfies

$$\|\Delta + \eta_Z g_t\|_F \ge 1 - \eta_Z \|g_t\|_F > c \eta_Z \|g_t\|_F,$$

while the zero update also fails the defining inequality since $c < 1$. Thus $q_t = 0$ in this case, and the bound reduces to non-increase from retaining the current state. A smaller reference step alone is therefore insufficient to ensure a positive descent term.

Proof of Proposition 2. Fix an iteration t. If $\Delta \in \mathcal{G}_t$, then by definition of $\mathcal{G}_t$,

$$\langle g_t, \Delta \rangle = \langle g_t, -\eta_Z g_t \rangle + \langle g_t, \Delta + \eta_Z g_t \rangle \le -\eta_Z \|g_t\|_F^2 + \|g_t\|_F \, \|\Delta + \eta_Z g_t\|_F \le -(1-c)\eta_Z \|g_t\|_F^2,$$

and also

$$\|\Delta\|_F \le \|\eta_Z g_t\|_F + \|\Delta + \eta_Z g_t\|_F \le (1 + c)\eta_Z \|g_t\|_F.$$

Combining this with (7) gives

$$\left\| \widehat{W}(Z_t + \Delta, s) - \widehat{W}(Z_t, s) \right\|_F^2 \le \kappa^2 (1 + c)^2 \eta_Z^2 \|g_t\|_F^2.$$

Therefore, by Corollary 1,

$$L(Z_t + \Delta, s) \le L(Z_t, s) - \left(1 - c - \frac{\beta\kappa^2\eta_Z}{2}(1+c)^2\right)\eta_Z\|g_t\|_F^2 = L(Z_t, s) - \alpha\eta_Z\|g_t\|_F^2.$$

Now let

$$E_t = \Big\{\text{at least one of the } M \text{ sampled candidates has update in } \mathcal{G}_t\Big\}.$$

By independence of the M samples,

$$\Pr(E_t \mid Z_t) = 1 - (1 - q_t)^M.$$

On the event E_t, the sampled set contains a candidate whose loss is at most $L(Z_t, s) - \alpha\eta_Z\|g_t\|_F^2$. On E_t, the best-of-M rule therefore returns a candidate whose loss is no larger than this value. On the complementary event E_t^c, the strict accept-if-improving rule gives $L(Z_{t+1}, s) \le L(Z_t, s)$. Combining the two cases yields the pointwise bound

$$L(Z_{t+1}, s) \le L(Z_t, s) - \alpha\eta_Z\|g_t\|_F^2\mathbf{1}_{E_t}.$$

Taking conditional expectation gives

$$\mathbb{E}[L(Z_{t+1}, s) \mid Z_t] \le L(Z_t, s) - \alpha\,\eta_Z\big[1 - (1 - q_t)^M\big]\|g_t\|_F^2,$$

which proves the claimed one-step descent inequality. □

G Low-Rank Code Update

The core search rule described above only requires a code matrix $Z \in \mathcal{Z}$ to induce weight $\widehat{W}(Z)$, and a differentiable loss to compute (3). As a result, the same rule can also be used with other parameterizations. Low-rank parameterizations are widely used in LLM fine-tuning to reduce the number of optimization variables and improve parameter efficiency. We apply this principle in code space by parameterizing a local code update as a low-rank decomposition with two integer-valued factors:

$$\Delta Z(A, B) = AB^\top, \qquad A \in \mathbb{Z}^{d_{\text{out}} \times R}, \quad B \in \mathbb{Z}^{d_{\text{in}} \times R}.$$

This reduces the number of searched coordinates from $d_{\text{out}}d_{\text{in}}$ to $R(d_{\text{out}} + d_{\text{in}})$. Unlike ordinary LoRA, A and B are not inference-time residual adapters; after training, only the deployed code state $\text{clip}(Z + \Delta Z(A, B), 1, K)$ needs to be stored. The factor-wise guidance follows from the same surrogate chain rule. Given $\nabla_Z L$ by (3) at the induced code state, we use

$$\nabla_A L = \nabla_Z L\, B, \qquad \nabla_B L = (\nabla_Z L)^\top A.$$

We form guided references for A and B, sample integer-valued factor candidates, and select by the deployed loss after merging the induced code update.