QUANTIZED EVOLUTION STRATEGIES: HIGH-PRECISION FINE-TUNING OF QUANTIZED LLMS AT LOW-PRECISION COST

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-Training Quantization (PTQ) is essential for deploying Large Language Models (LLMs) on memory-constrained devices, yet it renders models static and difficult to fine-tune. Standard fine-tuning paradigms, including Reinforcement Learning (RL), fundamentally rely on backpropagation and continuous weights to compute gradients. Thus they cannot be used on quantized models, where the parameter space is discrete and non-differentiable. While Evolution Strategies (ES) offer a backpropagation-free alternative, optimization of the quantized parameters can still fail due to vanishing or inaccurate gradient estimation. This paper introduces Quantized Evolution Strategies (QES), an optimization paradigm that performs full-parameter fine-tuning directly in the quantized space. QES is based on two innovations: (1) it integrates accumulated error feedback to preserve high-precision weight updating signals, and (2) it utilizes a stateless seed replay to reduce memory usage to low-precision inference levels. QES significantly outperforms the state-of-the-art zeroth-order fine-tuning methods on a variety of tasks, making direct fine-tuning for quantized models possible. It therefore opens up the possibility for scaling up LLMs entirely in the quantized space.

1 INTRODUCTION

The scaling of Large Language Models (LLMs) has unlocked emergent capabilities in mathematical reasoning, coding, and general problem-solving (Shao et al., 2024; OpenAI et al., 2024; Guo et al., 2025). However, this performance comes at a significant computational cost. To mitigate the memory bottleneck of large-scale deployment, Post-Training Quantization (PTQ) has become standard practice. Techniques such as activation smoothing (e.g. SmoothQuant, Xiao et al., 2024) and layer-wise weight compression (e.g. GPTQ; Frantar et al., 2023; AWQ; Lin et al., 2024) make 3-4 bit precision inference possible with negligible performance degradation. It is then possible to deploy such models on consumer-grade hardware.

Whereas PTQ democratizes *inference*, the model becomes essentially a static artifact. Any fine-tuning is generally done before the model is quantized, and requires massive, training-grade compute clusters that PTQ practitioners typically do not have access to. Methods that allow fine-tuning quantized models directly are limited. While Zeroth-Order (ZO) optimization and Evolution Strategies (ES) have emerged as memory-efficient alternatives to pre-quantization fine-tuning (Malladi et al., 2024; Qiu et al., 2025), they are severely limited in discrete parameter spaces. Backpropagation-free approaches like QuZO and others (Zhou et al., 2025; Feng et al., 2024) have shown promise in Supervised Fine-Tuning (SFT) on quantized models, but struggle with reasoning tasks. The problem, identified in this paper as the *stagnation problem*, is that the discrete nature of the quantized parameter space causes gradient signals to vanish, leading to a collapse of the optimization process. Moreover, inaccurate gradient signals due to the discretization errors severely reduce the optimization efficiency.

This paper introduces Quantized Evolution Strategies (QES; Figure 1), a novel optimization paradigm designed to perform full-parameter fine-tuning directly on the quantized model. By bridging quantized evolution with signal processing principles, specifically Delta-Sigma modulation (Inose et al., 1962; Razavi, 2016), it proposes an accumulated error feedback mechanism that allows the optimizer to sense and traverse high-precision update trajectories even in ultra-low bit settings (e.g., four-bit

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

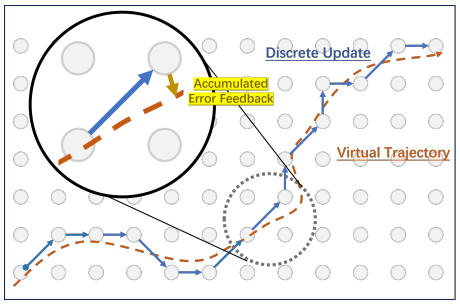

Figure 1: An overview of Quantized Evolution Strategies (QES). The goal is to optimize quantized LLMs directly on discrete parameter space in a memory-efficient manner. Error residuals are accumulated across iterations until they reach a threshold for making a discrete change. QES achieves temporal equivalence to the high-precision optimization trajectory, while maintaining the memory requirements of inference-only quantized models.

integers denoted as INT4). This mechanism mirrors historical techniques in communication-efficient training, such as 1-bit stochastic gradient descent (Seide et al., 2014; Ström, 2015; Karimireddy et al., 2019), but adapts them for the first time to the strict constraints of backpropagation-free quantized optimization. Furthermore, to address the memory overhead of tracking these error residuals, the *Stateless Seed Replay* mechanism is introduced. It reconstructs optimization states on-the-fly, reducing GPU memory requirements of QES to inference level of the low-precision quantized model.

The paper makes three main contributions:

1. **A Mechanism for Optimization in Quantized Space:** QES enables fine-tuning of LLMs directly within their low-precision quantized weights, yet maintains high-precision learning dynamics. Unlike gradient-based methods that require differentiable operations, QES utilizes accumulated error feedback to ensure progress on low-precision non-differentiable landscapes.
2. **Inference-Level Memory Footprint:** By replacing the storage of high-precision optimizer states with stateless seed replay, QES reduces GPU memory consumption to the order of low-precision inference. This level allows full-parameter learning on hardware that previously could only accommodate quantized inference.
3. **Overcoming Gradient Stagnation and Discretization Inaccuracy:** QES resolves two challenges for optimization in quantized space: (1) stagnation of learning due to small gradient signals, and (2) inaccurate parameter updates due to discretization errors. An analysis of the update dynamics shows why: the residual never exceeds half a grid step, so rounding error is held in state rather than accumulating as a random walk, and γ sets the smallest signal the optimizer can act on (Propositions 1 and 2).

Evaluated in two reasoning tasks and four SFT benchmarks, QES improves the quantized baseline models by a wide margin, and does so far better than the state-of-the-art quantized fine-tuning method QuZO. It therefore democratizes fine-tuning, making it possible in low resource environments. The results also suggest a new avenue for scaling LLMs in the future by including more quantized parameters in the same amount of memory.

2 Related Work

Methods for quantizing LLMs, fine-tuning without gradients, and reducing quantization errors are reviewed below.

2.1 LLM Quantization

Post-Training Quantization (PTQ) methods such as SmoothQuant (Xiao et al., 2024), GPTQ (Frantar et al., 2023), and AWQ (Lin et al., 2024) enable 3–4 bit inference with little degradation. Quantization-aware training (QAT) approaches aim at training the high-precision network such that it is easily quantizable afterwards (Liu et al., 2024).

There are also works on fine-tuning the quantized models through backpropagation, e.g. by use of the straight-through estimator (STE) (Bengio et al., 2013). This suffers from two critical limitations: (1) STE is inherently biased, and (2) it requires full-precision latent weights and optimizer states, undermining the memory benefit of quantized models. Another approach is QLoRA (Dettmers et al., 2023), which freezes the quantized weights and optimizes small high-precision adapters; while

memory-efficient, this restricts updates to a low-rank subspace and leaves the quantized parameters themselves unchanged. Therefore, optimization paradigms capable of performing full-parameter fine-tuning directly on the quantized model are needed.

2.2 Zeroth Order Fine-tuning for Quantized Models

Improving alignment and reasoning ability remains the central focus of LLM fine-tuning. The standard approach is Reinforcement Learning with Human Feedback (RLHF), and it has been successful in general preference alignment (Ouyang et al., 2022), mathematical reasoning (Shao et al., 2024; Yu et al., 2025; Liu et al., 2025), code generation (Gehring et al., 2024; Guo et al., 2025), and general reasoning (OpenAI et al., 2024; An et al., 2025). However, RLHF relies fundamentally on backpropagation, restricting its application to differentiable, full-precision architectures with significant memory.

Recently, Zeroth-Order (ZO) optimizer and Evolution Strategies (ES) have emerged as backpropagation-free alternatives. Building on simultaneous perturbation stochastic approximation (SPSA; Spall, 1992), the MeZO method (Malladi et al., 2024) pioneered the scaling of ZO-SGD (Ghadimi & Lan, 2013) to billion-parameter models. Similarly, recent work has demonstrated that ES can match or exceed RL performance in similar high-dimensional spaces (Qiu et al., 2025). However, these methods were originally designed for high-precision continuous parameters, and cannot be directly used on quantized models.

Nevertheless, the backpropagation-free paradigm offers a natural basis for quantized fine-tuning. Recent approaches like QuZO (Zhou et al., 2025) and QZO (Feng et al., 2024) extended ZO optimizers to supervised fine-tuning of quantized models. However, so far their success in reasoning tasks has been limited; they struggle to capture sparse signals, often leading to no meaningful convergence. QES addresses this gap by connecting quantized evolution to signal processing principles.

2.3 Techniques for Reducing Discretization Errors

The fundamental challenge of approximating a continuous signal with discrete steps is well-studied in the context of Delta-Sigma modulation (Inose et al., 1962; Razavi, 2016). In this framework, the error introduced by taking a discrete step (i.e. quantization) is not discarded but accumulated and applied to subsequent steps. This mechanism, known as noise shaping in DAC design, ensures that while individual steps may be coarse, the time-averaged behavior of the system tracks the continuous trajectory accurately.

In the domain of gradient compression, this classical principle can be instantiated as residual accumulation, or error feedback (Seide et al., 2014; Ström, 2015; Karimireddy et al., 2019). These methods compress gradient information for efficiency but accumulate the resulting error for correction in later steps. Unlike Stochastic Rounding (Gupta et al., 2015; Zhou et al., 2025), which achieves unbiased updates via probabilistic sampling but suffers from high variance, this method ensures convergence via deterministic accumulation of residuals. This process effectively "shapes" the quantization noise of the optimization trajectory, making it possible to apply quantized evolution to fine-tuning without the instability of prior methods.

3 Method

The challenge of quantized optimization is described first, followed by the QES method with accumulated error feedback and seed replay.

3.1 The Optimization Challenge

In LLM reasoning tasks, the objective is to select model parameters $\mathbf{W}$ that maximize a reward function $J(\mathbf{W})$ defined for this task:

$$\max_{\mathbf{W}\in\mathcal{W}} \mathbb{E}[J(\mathbf{W})]. \tag{1}$$

Standard first-order optimization methods (e.g., SGD, Adam) cannot be applied in the quantized setting for two reasons: (1) the quantization operator is non-differentiable, and (2) there is insufficient memory for storing high-precision gradients and optimizer states in many edge applications. Consequently, ES is used to estimate descent directions via parameter-space exploration.

In the standard continuous setting (Salimans et al., 2017; Qiu et al., 2025), ES approximates the gradient of the expected reward $\nabla\mathbb{E}[J(\mathbf{W})]$ using a population of N perturbations by optimizing a Gaussian-smoothed objective $\max_{\mathbf{W}\in\mathcal{W}} \mathbb{E}_{\epsilon\in\mathcal{N}(0,\boldsymbol{I})}[J(\mathbf{W}+\sigma\epsilon)]$.

For a current parameter state $\mathbf{W}_t$, the gradient estimate $\hat{\boldsymbol{g}}$ is computed as:

$$\hat{\boldsymbol{g}} = \frac{1}{N\sigma}\sum_{i=1}^{N} F_i \cdot \boldsymbol{\epsilon}_i, \tag{2}$$

where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \boldsymbol{I})$ is a random perturbation sampled from a standard normal distribution, σ is the standard deviation for scaling the perturbation, and $F_i = F(\mathbf{W}_t + \sigma\boldsymbol{\epsilon}_i)$ is the fitness (reward) corresponding to the perturbed weights. Note that fitness is thus the normalized reward score derived from $J(\mathbf{W}_t + \sigma\boldsymbol{\epsilon}_i)$ to make sure optimization is stable. Fitness is evaluated on a problem set according to the reinforcement learning from verifiable rewards (RLVR) framework (Shao et al., 2024), where the model is queried and assessed based on whether its responses are correct.

Standard ES updates the parameters via gradient ascent: $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t + \alpha\hat{\boldsymbol{g}}$ with learning rate α. While effective in continuous spaces at large scale (Qiu et al., 2025), this update rule fails in quantized spaces due to the vanishing magnitude of the update steps relative to the discretization granularity, or errors introduced during discretization. The solution is to provide error feedback in an accumulated manner, as will be described next.

3.2 Accumulated Error Feedback

QES optimizes quantized LLMs directly within the discrete integer space $\mathbf{W} \in \{0, \ldots, 2^B - 1\}^d$, where d is the number of model parameters and B is the number of bits to represent each parameter, Thus, the parameters $\mathbf{W}$ lie on a low-precision lattice $\mathcal{W}_\mathcal{Q}$ (e.g., INT4) defined by the quantization operator $\mathcal{Q}(\cdot)$. Since the continuous perturbation $\sigma\boldsymbol{\epsilon}$ from Eq. (2) violates quantization constraints, a stochastic perturbation strategy is adapted from prior work (Connolly et al., 2021; Zhou et al., 2025).

A discrete perturbation $\boldsymbol{\delta}$ is created by stochastically rounding the scaled Gaussian noise $\sigma\boldsymbol{\epsilon}$ (where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$):

$$\boldsymbol{\delta} = \lfloor \sigma\boldsymbol{\epsilon} \rfloor + \boldsymbol{b}, \quad \text{where } \boldsymbol{b} \sim \text{Bernoulli}(\sigma\boldsymbol{\epsilon} - \lfloor \sigma\boldsymbol{\epsilon} \rfloor). \tag{3}$$

By recording the random seeds used to generate $\boldsymbol{\epsilon}$ and $\boldsymbol{b}$, the $\boldsymbol{\delta}$ can be reproduced during optimization without the memory overhead of stored perturbation vectors (Qiu et al., 2025; Salimans et al., 2017). To strictly enforce the codebook limits defined by $\mathcal{W}_\mathcal{Q}$, boundary gating is applied to mask invalid updates:

$$\mathbf{W}'_{ij} = \begin{cases} \mathbf{W}_{ij} + \boldsymbol{\delta}_{ij} & \text{if } 0 \le \mathbf{W}_{ij} + \boldsymbol{\delta}_{ij} < 2^B, \\ \mathbf{W}_{ij} & \text{otherwise.} \end{cases} \tag{4}$$

The gradient direction $\hat{g}$ is then estimated by aggregating the discrete search directions weighted by their rewards:

$$\hat{\boldsymbol{g}} = \frac{1}{N\sigma}\sum_{i=1}^{N} F_i \cdot \boldsymbol{\delta}_i. \tag{5}$$

A critical challenge can be seen in Equation 5: the scaled update step $\alpha\hat{\boldsymbol{g}}$ is often smaller than half the spacing of the parameter lattice ($\|\alpha\hat{\boldsymbol{g}}\|_\infty < 0.5$ in integer code units), the point below which round-to-nearest returns zero. Rounding it naively fails either way: deterministic rounding gives $\Delta\mathbf{W} = \mathbf{0}$ and stagnates, while stochastic rounding moves the weights but trades that for unbounded rounding noise (Section 5).

Signal processing has a known solution to this: Delta-Sigma modulation (Inose et al., 1962; Razavi, 2016), which employs a feedback loop to accumulate quantization error over time, preserving signal fidelity. This approach is already used in communication-efficient training, such as 1-bit SGD, where uncompressed errors are carried forward to ensure convergence (Seide et al., 2014; Ström, 2015; Karimireddy et al., 2019). It can be used to solve the stagnation problem and preserve weight updating accuracy in QES.

In QES, a high-precision error vector $\boldsymbol{e}_t$ (typically FP16) that accumulates the quantization error from previous steps is maintained. Rather than discarding the fractional component of the update, it

Algorithm 1 QES with Accumulated Error Feedback

Input: Integer Weights $\mathbf{W}_0 \in \{0, \dots, 2^B - 1\}^d$, Learning Rate α, Decay γ, Population N, Maximum Iteration Number T
Output: Final model weights $\mathbf{W}_T$
Initialize: Residuals $\boldsymbol{e}_0 \leftarrow \mathbf{0}$ (FP16)
for $t = 0$ **to** $T - 1$ **do**
  for each perturbation $i \in \{1, \dots, N\}$ in parallel **do**
    Reconstruct $\boldsymbol{\delta}_i$ using Eq. (3) and seed s_i
    Apply perturbation: $\mathbf{W}' \leftarrow \text{Gate}(\mathbf{W}_t + \boldsymbol{\delta}_i)$
    Execute inference and compute reward F_i
  end for
  Normalize reward for population
  Estimate gradient $\hat{\boldsymbol{g}}_t \leftarrow \frac{1}{N\sigma} \sum_{i=1}^{N} F_i \cdot \boldsymbol{\delta}_i$
  $\boldsymbol{u}_t \leftarrow \alpha \hat{\boldsymbol{g}}_t + \gamma \boldsymbol{e}_t$ *// Apply accum. error*
  $\Delta \mathbf{W}_t \leftarrow \text{Round}(\boldsymbol{u}_t)$ *// Discretize update*
  $\boldsymbol{e}_{t+1} \leftarrow \boldsymbol{u}_t - \Delta \mathbf{W}_t$ *// Update accum. error*
  $\mathbf{W}_{t+1} \leftarrow \text{Gate}(\mathbf{W}_t + \Delta \mathbf{W}_t)$ *// Update weight*
end for
return $\mathbf{W}_T$

Algorithm 2 Stateless QES Update with Seed Replay

Input: Current Weights $\mathbf{W}_t$, New Seeds $\mathcal{S}_t$, New Rewards $\mathcal{F}_t$, History $\mathcal{H}$, Window K, Sigma σ
Output: Updated $\mathbf{W}_{t+1}$, New History $\mathcal{H}'$
Initialize: proxy residual $\tilde{\boldsymbol{e}} \leftarrow \mathbf{0}$
for $(\hat{\mathcal{S}}, \hat{\mathcal{F}}) \in \mathcal{H}$ **do**
  Re-generate $\boldsymbol{\delta}$ using Eq. (3) and history seed $\hat{\mathcal{S}}$
  Re-compute grad $\hat{\boldsymbol{g}} \leftarrow \text{Agg}(\boldsymbol{\delta}, \hat{\mathcal{F}}, \sigma)$
  $\boldsymbol{u}_{sim} \leftarrow \alpha \hat{\boldsymbol{g}} + \gamma \tilde{\boldsymbol{e}}$
  $\Delta_{sim} \leftarrow \text{Round}(\boldsymbol{u}_{sim})$
  $\Delta_{final} \leftarrow \text{Gate}(\mathbf{W}_t + \Delta_{sim}) - \mathbf{W}_t$
  $\tilde{\boldsymbol{e}} \leftarrow \boldsymbol{u}_{sim} - \Delta_{final}$ *// Update proxy error*
end for
Generate current $\boldsymbol{\delta}_t$ from seeds $\mathcal{S}_t$
$\hat{\boldsymbol{g}}_t \leftarrow \text{Agg}(\boldsymbol{\delta}_t, \mathcal{F}_t, \sigma)$
$\boldsymbol{u}_t \leftarrow \alpha \hat{\boldsymbol{g}}_t + \gamma \tilde{\boldsymbol{e}}$ *// Add rematerialized error*
$\Delta \mathbf{W} \leftarrow \text{Round}(\boldsymbol{u}_t)$
$\mathbf{W}_{t+1} \leftarrow \text{Gate}(\mathbf{W}_t + \Delta \mathbf{W})$
$\mathcal{H}' \leftarrow \text{Enqueue}(\mathcal{H}, (\mathcal{S}_t, \mathcal{F}_t))$
return $\mathbf{W}_{t+1}, \mathcal{H}'$

is carried forward. The update dynamics at step t are thus defined as:

$$\boldsymbol{u}_t = \alpha \hat{\boldsymbol{g}}_t + \gamma \boldsymbol{e}_t \tag{6}$$

$$\Delta \mathbf{W}_t = \text{Round}(\boldsymbol{u}_t) \tag{7}$$

$$\boldsymbol{e}_{t+1} = \boldsymbol{u}_t - \Delta \mathbf{W}_t, \tag{8}$$

with $\boldsymbol{e}_0 = \mathbf{0}$, where $\boldsymbol{u}_t$ represents the desired high-precision update, and $\gamma \in (0, 1]$ is a decay factor that controls how long past residuals are retained. This mechanism allows gradient signals far smaller than one grid step to accumulate in $\boldsymbol{e}$ over multiple iterations until they cross the rounding threshold ($|\boldsymbol{u}| \geq 0.5$ in code units), which has the effect of a much lower learning rate on the integer lattice. The full quantized evolution strategy with accumulated error feedback is listed in Algorithm 1.

3.3 Stateless Seed Replay

While Algorithm 1 ensures progress, it incurs a significant memory penalty. Storing the dense high-precision error vector $\boldsymbol{e}_t \in \mathbb{R}^d$ (typically FP16) often consumes more VRAM than the quantized model weights themselves, canceling the primary advantage of quantization.

To resolve this bottleneck, QES includes *Stateless Seed Replay*. Note that the error state $\boldsymbol{e}_t$ is deterministic given the initial condition and the history of optimization steps. Rather than persisting $\boldsymbol{e}_t$, it can be *rematerialized* on-the-fly by replaying a limited history window. To do that, a lightweight history buffer $\mathcal{H}_t = \{(\mathcal{S}_\tau, \mathcal{F}_\tau)\}_{\tau=t-K}^{t-1}$ is maintained, containing only the random seeds $\mathcal{S}$ and the scalar rewards $\mathcal{F}$ for the past K steps' populations. To perform an update at step t, the error accumulation process is re-simulated starting from an assumed zero error state at step $t - K$. Because the decay factor satisfies $\gamma < 1$, the discrepancy introduced by that assumption contracts by a factor of γ per replayed step, and is therefore at most $\gamma^K \Delta/2$ (Appendix G.6).

During the replay of step τ, boundary constraints are checked using the current weights $\mathbf{W}_t$ rather than reconstructing the historical weights $\mathbf{W}_\tau$. Since discrete updates are sparse, $\mathbf{W}_\tau \approx \mathbf{W}_t$, and the discrepancy in boundary masking is minimal.

Because the recursion in Equations (6)–(8) is coordinate-wise, replay runs chunk by chunk, so the transient working set is bounded by the chunk size rather than by d (Appendix E.1).

This approach, detailed in Algorithm 2, trades computation for memory, reducing optimizer-state memory from $O(d)$ to $O(K)$, with $K \approx 50$ against $d > 10^9$. A short window suffices: at the default

Table 1: SFT accuracy (%) on RoBERTa-large. Best W8 result in each column is in bold. QES achieves the best performance among quantized methods and on several tasks surpasses the W8 first-order baseline and the FP32 MeZO zeroth-order baseline.

METHOD	PREC.	SNLI	MNLI	RTE	SST-5	AVG
Full precision (upper bound)						
FIRST-ORDER	FP32	72.9	61.1	49.0	46.2	57.3
MEZO	FP32	34.0	34.0	56.2	21.7	36.5
8-bit weights						
FIRST-ORDER + STE	W8	50.0	**44.4**	49.0	20.4	41.0
QUZO	W8	32.3	40.3	44.8	19.6	34.2
QES (OURS)	W8	**55.6**	42.4	**55.2**	**24.4**	**44.4**

$\gamma = 0.9$ and $K = 50$, truncating the history leaves a residual error of at most 2.6×10^{-3}, against a residual range of $\Delta/2$.

4 EXPERIMENTS

QES is evaluated on three task families of increasing difficulty: classification-style supervised fine-tuning (SFT), arithmetic reasoning, and grade-school mathematics. The goal is to demonstrate that QES fine-tunes LLMs directly in their quantized form—without full-precision gradients or auxiliary optimizer states—across both differentiable and non-differentiable objectives. Section 4.1 describes the setup; Sections 4.2–4.3 report main results; Sections 4.4–4.6 analyze the residual and seed-replay mechanisms.

4.1 EXPERIMENTAL SETUP

Tasks. RoBERTa-large is fine-tuned on the **SFT** tasks SNLI, MNLI, RTE, and SST-5 following the protocol of MeZO (Malladi et al., 2024); performance is reported as classification accuracy. **Countdown** (Gandhi et al., 2024; Pan et al., 2025): a compact reasoning task in which the model must produce an arithmetic expression over $\{+, -, \times, /\}$ that evaluates to a target value (e.g., from $\{3, 4, 28, 52\}$ targeting 44, the solution is $28+52/4+3$). **GSM8K** (Cobbe et al., 2021): grade-school math word problems requiring multi-step numerical reasoning. For both reasoning tasks the prompt template from GRPO-Zero (Policy-gradient, 2025) is used and the reward is binary correctness.

Models and quantization. Qwen2.5-1.5B and -3B (Qwen et al., 2025) serve as the reasoning backbones, quantized to INT4 and INT8 with GPTQ (Frantar et al., 2023) and to W8A8 (both weights and activations are quantized to INT8 precision) with LLM-Compressor (RedHatAI & vLLM Project, 2024). RoBERTa-large is used at FP32 and at W8 (GPTQ). The W8A8 recipe applies SmoothQuant-style outlier migration, so those rows already exercise a non-GPTQ pipeline.

Baselines. QES is compared against: the **Base Model** (quantized, no fine-tuning); **QuZO** (Zhou et al., 2025), a quantized zeroth-order method that is the primary quantized baseline in both regimes; **MeZO** (Malladi et al., 2024), full-precision zeroth-order SGD (not applicable to quantized space); **First-Order (FO)** backpropagation, in FP32 (upper bound) and on the W8 backbone with straight-through estimation (SFT only); and **Full Residual**, a QES variant that stores FP16 residuals, used as an oracle in the residual ablation (Section 4.4).

We specify more details of training and implementations in Appendix A.

4.2 SUPERVISED FINE-TUNING

Table 1 reports SFT accuracy on the four classification tasks. Among methods constrained to the W8 backbone, QES improves substantially over QuZO (44.4 vs. 34.2 on average) and over the W8 first-order baseline with STE (41.0). Notably, QES also outperforms full-precision MeZO (36.5), suggesting that error feedback recovers more useful update signal than smoothed ZO estimation, even when the latter operates in FP32. The remaining gap to full-precision first-order (57.3) is the cost of optimizing on a discrete lattice with forward passes only, and no optimizer state.

Table 2: Reasoning accuracy (%) on Countdown and GSM8K across model sizes and quantization formats. QES delivers consistent and often large gains over both the quantized base and QuZO across every model size and format.

		COUNTDOWN			GSM8K		
MODEL	FORMAT	BASE	QUZO	QES	BASE	QUZO	QES
QWEN2.5-1.5B	INT4	3.50	5.25	**16.00**	0.00	0.00	**9.86**
	INT8	4.20	4.50	**26.35**	1.59	1.44	**12.21**
	W8A8	4.20	4.20	**15.35**	3.56	4.17	**12.28**
QWEN2.5-3B	INT4	2.80	14.25	**31.85**	48.45	48.60	**77.56**
	INT8	4.50	15.85	**37.40**	11.90	54.28	**78.77**
	W8A8	8.20	10.75	**21.35**	24.49	4.40	**80.82**

4.3 REASONING

Table 2 summarizes accuracy on Countdown and GSM8K across model sizes and quantization formats. Two patterns are immediate. First, QuZO is brittle on the coarse INT4 lattice and on the smaller 1.5B model: on Qwen2.5-1.5B INT4, it improves over base by only 1.75 points on Countdown and fails to attain any improvement on GSM8K. Second, QES achieves consistent and often large gains across all settings. The most striking case is Qwen2.5-3B in W8A8, where QuZO collapses to 4.40% on GSM8K while QES reaches 80.82%. The gap between QuZO and QES widens with task difficulty (Countdown $\rightarrow$ GSM8K) and narrows with larger model scale, consistent with the intuition that smaller, more aggressively quantized models present sharper, more brittle landscapes for stateless ZO methods (Li et al., 2018). Accuracy is graded by strict answer-format extraction, which is why the 1.5B GSM8K base is near zero (Appendix A.3). The conclusion does not rest on that weak base: at 3B, where the base already scores 48.45 under the same strict grading, QES still gains 29.1 points. Two case studies in Appendix C extend the result beyond this grid, with Llama-3.1-8B INT4 reaching 82.64% on GSM8K from a 64.14% base and Qwen2.5-32B-Instruct INT4 reaching 51.40% on Countdown from a 39.25% base; both reuse the Qwen2.5-3B hyperparameters without retuning. Countdown training curves are shown in Figure 2 in Appendix B.

4.4 ABLATION STUDIES OF STATELESS SEED REPLAY

Residual storage (Stateless Replay vs. Full Residual). The Full Residual variant of QES stores FP16 residuals explicitly and serves as an oracle that isolates the contribution of error feedback when memory is unconstrained. Table 6 compares it against Stateless Seed Replay on Countdown. The two are within a few points across all six configurations and, in three of six, the seed-replay variant matches or exceeds the oracle—suggesting that any approximation introduced by replaying historical states with current weights is, in practice, on the scale of seed variance. The largest gap is Qwen2.5-3B W8A8 (21.35 vs. 31.70); on the same configuration on GSM8K, however, the gap closes to within 0.76 points (Full Residual: 81.58%, QES: 80.82%), indicating that this discrepancy is task- and seed-specific rather than a systematic failure of the approximation.

Window size and decay. Appendix D (Table 7 Top) evaluates the replay window K and decay factor γ on Qwen2.5-1.5B INT4 (Countdown). Two regimes are compared: γ scaled so that historical errors vanish within the window ($\gamma^K \approx 0$), and γ fixed at 0.90. Under the scaled regime, performance is highly sensitive to decay: shrinking K from 50 to 10 forces γ down to 0.58 and collapses accuracy from 16.00% to 4.55%. Under fixed $\gamma = 0.90$, even K=10 retains 13.05%, showing that the collapse is driven by aggressive decay rather than truncated history. In practice this means K is a tunable knob: shrinking it from 50 to 20 reduces reconstruction cost by 60% while losing only 1.25 points (fixed decay). The inherited σ, α, and N from continuous ES are likewise robust (Appendix D).

4.5 FIDELITY OF STATELESS SEED REPLAY

Replay regenerates the perturbation exactly from the seed but gates boundaries with the current weights W_t rather than the historical W_τ, so a reconstruction error needs a parameter to cross a quantization boundary inside the window. Define the *update ratio* as the fraction of parameters

changed per step, and the *boundary hit ratio* ρ as the fraction of those changes that land on a quantization boundary. Appendix D (Table 7, bottom) shows the update ratio is uniformly $\approx 10^{-2}$ and ρ is negligible across formats (e.g., $< 10^{-5}$ for INT4). Since active updates at boundaries rarely occur, seed-replay tracks the Full-Residual oracle almost perfectly.

4.6 Computational Cost Analysis

Stateless Seed Replay holds optimizer state at a constant ≈29.7 KB, but its compute cost grows linearly with K. Replaying K=50 generations sequentially makes the QES update 13–17× slower than QuZO's, but rollouts dominate, so end to end it adds 6.1/7.8/12.9% at 1.5B/3B/8B over QuZO. Sharding the reconstruction across the idle inference GPUs cuts that to about 1%. Full wall-clock and peak-VRAM measurements are in Appendix E.

5 Temporal Equivalence to Continuous Optimization

To understand why QES succeeds while other quantized methods stagnate or converge slowly, it helps to compare its dynamics against an ideal continuous model. The quantized parameters $\mathbf{W}$ are confined to a discrete grid $\mathcal{W}_\mathcal{Q}$ with spacing Δ, while the reward $J(\mathbf{W})$ is defined over a continuous domain; assume a Gaussian smoothed objective $J_\sigma(\mathbf{W}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[J(\mathbf{W}+\sigma\boldsymbol{\epsilon})]$ (Figure 3 in Appendix F). It can be optimized by high-precision gradient ascent, but such a mechanism is not available on the discrete grid. Instead, the gradient needs to be approximated. Proofs are given in Appendix G.

One approach is to use a stochastic gradient estimator $\hat{\boldsymbol{g}}_t$, like QuZO does (Zhou et al., 2025). Utilizing stochastic perturbation (double quantization) makes the perturbation itself unbiased, $\mathbb{E}[\boldsymbol{\delta} \mid \boldsymbol{\epsilon}] = \sigma\boldsymbol{\epsilon}$, so that the estimator tracks the smoothed gradient $\nabla J_\sigma(\mathbf{W}_t)$. However, an unbiased perturbation is not sufficient for high performance on a discrete grid. The critical failure occurs at the application of the gradient update.

Consider the standard update rule at step t with learning rate α and a quantization operator $\mathcal{Q}$, $\mathbf{W}_{t+1} = \mathbf{W}_t + \mathcal{Q}(\alpha\hat{\boldsymbol{g}}_t)$. The operator decomposes into the identity plus an error term, $\mathcal{Q}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{\xi}$, so expanding the trajectory over T steps yields

$$\mathbf{W}_T = \mathbf{W}_0 + \underbrace{\sum_{t=0}^{T-1} \alpha\hat{\boldsymbol{g}}_t}_{\text{Ideal Continuous Update}} + \underbrace{\sum_{t=0}^{T-1} \boldsymbol{\xi}_t}_{\text{Accumulated Quantization Loss}}. \tag{9}$$

Both failure modes can be read off this equation. The first is stagnation: the update is usually smaller than the grid precision ($\|\alpha\hat{\boldsymbol{g}}_t\|_\infty < \Delta/2$), so $\mathcal{Q}(\alpha\hat{\boldsymbol{g}}_t) \to \mathbf{0}$ and $\boldsymbol{\xi}_t = -\alpha\hat{\boldsymbol{g}}_t$: the *Accumulated Quantization Loss* cancels the *Ideal Continuous Update* exactly, leaving $\mathbf{W}_T = \mathbf{W}_0$. The second is variance explosion: if $\mathcal{Q}$ is stochastic instead, $\boldsymbol{\xi}_t$ becomes a zero-mean noise variable with per-coordinate variance $|\alpha\hat{g}_{t,i}|(\Delta - |\alpha\hat{g}_{t,i}|)$. The accumulated deviation is then a random walk growing as $\Theta(\sqrt{T})$, and a coordinate with a persistent signal needs about $\Delta/(\alpha|\hat{g}_i|)$ steps before its drift clears that noise.

QES solves this problem by introducing a residual state $\boldsymbol{e}_t$ to enforce *temporal equivalence* between the discrete and continuous domains. First, *Virtual Continuous Parameters* $\boldsymbol{\Theta}_t$ are defined as the sum of the physical discrete weights and the carry-over residual error:

$$\boldsymbol{\Theta}_t \triangleq \mathbf{W}_t + \boldsymbol{e}_t. \tag{10}$$

Then, expanding this equation with the QES update rules in Equations (6)–(8), in the undecayed case $\gamma = 1$, gives

$$\begin{aligned} \boldsymbol{\Theta}_{t+1} &= \mathbf{W}_{t+1} + \boldsymbol{e}_{t+1} \\ &= (\mathbf{W}_t + \Delta\mathbf{W}_t) + (\boldsymbol{u}_t - \Delta\mathbf{W}_t) \\ &= \mathbf{W}_t + \boldsymbol{u}_t = \mathbf{W}_t + (\alpha\hat{\boldsymbol{g}}_t + \boldsymbol{e}_t) \\ &= (\mathbf{W}_t + \boldsymbol{e}_t) + \alpha\hat{\boldsymbol{g}}_t = \boldsymbol{\Theta}_t + \alpha\hat{\boldsymbol{g}}_t. \end{aligned} \tag{11}$$

Thus, the virtual parameters $\boldsymbol{\Theta}_t$ evolve according to the dynamics of unconstrained, high-precision gradient ascent. The physical quantized weights W_T at any step T are related to this ideal trajectory

by

$$\mathbf{W}_T = \mathbf{\Theta}_T - \boldsymbol{e}_T = \left(\mathbf{W}_0 + \sum_{t=0}^{T-1} \alpha \hat{\boldsymbol{g}}_t \right) - \boldsymbol{e}_T. \tag{12}$$

Crucially, unlike the stateless baseline where errors sum linearly or as a random walk, the deviation in QES at $\gamma = 1$ is the *single final residual* $\boldsymbol{e}_T$: it is what is left over after rounding $\boldsymbol{u}_{T-1}$ to the nearest grid point, so $\|\boldsymbol{e}_T\|_\infty \leq \Delta/2$. The residual integrates gradient signals far smaller than one grid step until they cross the quantization threshold, triggering a discrete update $\Delta\mathbf{W} \neq 0$ that aligns the discrete model with the virtual continuous path.

The decay factor γ sets what this costs. For $\gamma < 1$ a leak term $-(1-\gamma)\boldsymbol{e}_t$ appears: $\|\boldsymbol{e}_t\|_\infty \leq \Delta/2$ still holds, but the virtual path now trails ideal ascent by up to $(1-\gamma)\Delta/2$ per step, and a coordinate with a persistent signal a moves only if

$$|a| > (1-\gamma)\,\Delta/2. \tag{13}$$

Stateless round-to-nearest is the $\gamma = 0$ case, where any signal below half a grid step never moves the weight, and the threshold shrinks in proportion to $1-\gamma$ until it disappears at $\gamma = 1$. A pure integrator never forgets, though, so its state cannot be rebuilt from a finite window: $\gamma < 1$ is what Stateless Seed Replay needs, whereas Full Residual can run at $\gamma = 1$. It is therefore γ, not K, that decides whether a signal is recoverable, which is what the ablation in Section 4.4 measures. Appendix G proves both for general γ, sharpens the bias–variance contrast, and shows the average squared gradient falling at the standard ES rate down to a floor set by the lattice spacing and the dead zone, neither of which is removed by shrinking α (Propositions 1–3, Theorem 1 and Equation (20)).

6 Limitation and Future Work

Several promising directions exist in overcoming the limitations of the current work and extending the capabilities of QES. First, this paper only focuses on standard linear integer quantization (INT4, INT8, W8A8). It should be possible to extend QES to more aggressive and non-uniform quantization paradigms, such as binary networks and floating-point formats (e.g., FP4). Second, the current stateless replay relies on a fixed lookback window K to manage the compute-memory trade-off. An adaptive algorithm could be created that automatically tunes K and the decay rate γ based on real-time convergence stability or available hardware resources, effectively removing the need for manual hyperparameter selection. Third, the quantization scales are frozen throughout, so QES moves only the integer codes; optimizing them jointly is a natural extension (Appendix A).

Quantization has so far been motivated by running existing models on limited hardware. QES suggests a different use: quantize a much larger model from the start and train it directly in the quantized space. Two factors compound here. Precision can be traded for parameters, four-fold in the case of INT4, and only low-precision inference is needed during training, about 1/12 of the memory required by backpropagation (Malladi et al., 2024). Together they put one to two orders of magnitude more parameters within reach of the same hardware.

7 Conclusion

Several barriers make effective fine-tuning of quantized LLMs difficult: gradient stagnation, inaccurate parameter update due to discretization, and prohibitive memory requirements. This paper introduced QES, a novel backpropagation-free optimization framework designed to perform full-parameter optimization directly on quantized integer weights. First, vanishing or inaccurate gradient estimations were identified as the primary cause of failure for existing ZO methods. Inspired by Delta-Sigma modulation, an accumulated error feedback mechanism was constructed to preserve and leverage these minuscule learning signals effectively. Second, to eliminate the memory overhead of storing high-precision error states, a Stateless Seed Replay mechanism was developed. This mechanism makes full-parameter fine-tuning possible within the strict memory constraints of standard quantized inference. These mechanisms were evaluated on SFT benchmarks and reasoning tasks. QES successfully overcame the quantized gradient issues and memory restrictions, significantly outperforming the state-of-the-art quantized fine-tuning baseline and achieving parity with memory-intensive oracles. Thus, by enabling the adaptation of large models on consumer-grade hardware, QES represents a significant step toward democratizing access to LLM fine-tuning.

AI DISCLOSURE

In this work, we used generative AI tools for grammar and writing check, as well as editing software code. We have reviewed all AI-assisted work. We checked the works by AI in writing and code manually. We did not use generative AI tools to formulate or prove the mathematical claims, to design the method or experiments, or to interpret results; generating synthetic data, translation, and qualitative analysis are not applicable to this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

QES is specified in Section 3 (Algorithms 1 and 2). Appendix A lists the hyperparameters, seeds, and evaluation protocols for every reported cell (Tables 3 and 4); Appendix E gives the hardware and the software versions used for the cost measurements; and Appendix G states the assumptions behind the analysis in Section 5 and gives complete proofs. All models and datasets are public. Anonymized code and launch scripts are in the supplementary material.

REFERENCES

Y. An et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Michael P. Connolly, Nicholas J. Higham, and Theo Mary. Stochastic rounding and its probabilistic backward error analysis. *SIAM Journal on Scientific Computing*, 43(1):A566–A585, 2021. doi: 10.1137/20M1334796. URL https://doi.org/10.1137/20M1334796.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoA: Efficient finetuning of quantized LLMs. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Ramchalam K Ramakrishnan, Zhaocong Yuan, and Andrew Z Li. Stepping forward on the last mile. *Advances in Neural Information Processing Systems*, 37:94851–94870, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (SoS): Learning to search in language, 2024. URL https://arxiv.org/abs/2404.03683.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. RLEF: grounding code LLMs in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*, 2024.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

D. Guo et al. Deepseek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

H. Inose, Y. Yasuda, and J. Murakami. A telemetering system by code modulation - $\Delta\Sigma$ modulation. *IRE Transactions on Space Electronics and Telemetry*, SET-8(3):204–209, 1962. doi: 10.1109/IRET-SET.1962.5008839.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *International conference on machine learning*, pp. 3252–3261. PMLR, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

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 LLM compression and acceleration, 2024. URL https://arxiv.org/abs/2306.00978.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes, 2024. URL https://arxiv.org/abs/2305.17333.

OpenAI et al. OpenAI o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. TinyZero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Policy-gradient. GRPO-Zero: Implementing deepseek r1's grpo algorithm from scratch. https://github.com/policy-gradient/GRPO-Zero, 2025.

Xin Qiu, Yulu Gan, Conor F. Hayes, Qiyao Liang, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: LLM fine-tuning beyond reinforcement learning, 2025. URL https://arxiv.org/abs/2509.24372.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Behzad Razavi. The delta-sigma modulator [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 8(2):10–15, 2016.

RedHatAI and vLLM Project. LLM compressor, 8 2024. URL https://github.com/vllm-project/llm-compressor.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. URL https://arxiv.org/abs/1703.03864.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech*, volume 2014, pp. 1058–1062. Singapore, 2014.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi: 10.1109/9.119632.

Nikko Ström. Scalable distributed dnn training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*. INTERSPEECH, 2015. URL https://www.isca-archive.org/interspeech_2015/strom15_interspeech.pdf.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024. URL https://arxiv.org/abs/2211.10438.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

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. *arXiv preprint arXiv:2502.12346*, 2025.

A Hyperparameters and Implementation Details

This appendix lists all hyperparameters used to produce the results in Section 4. Section A.1 describes settings shared across both task families; Section A.2 covers the SFT experiments on RoBERTa-large reported in Table 1; and Section A.3 covers the reasoning experiments on Qwen2.5 reported in Table 2. Code and launch scripts to reproduce every cell are provided in the supplementary material.

A.1 Common Settings

Across both task families, QES uses Algorithm 1 with the following defaults: population size $N = 8$ (for SFT) and $N = 50$ (for reasoning) antithetic perturbation pairs per generation, decay factor $\gamma = 0.9$, and a 4-bit stochastically rounded perturbation tensor for $\boldsymbol{\delta}$ (independent of the weight bit-width B). Weights are quantized with a symmetric, per-output-channel grid: scale $s = \max_j |\mathbf{W}_{ij}|/(2^{B-1} - 1)$, range $[-(2^{B-1} - 1), 2^{B-1} - 1]$. The grid is symmetric, so there is no zero-point term, and the scales are produced once by GPTQ (INT4/INT8) or LLM-Compressor (W8A8) and then frozen; QES optimizes only the integer codes. Freezing the grid keeps the fine-tuned model on the deployed inference codebook, so it can be served by the same kernels with no re-quantization or recalibration step. It also keeps the stagnation problem intact: a learnable scale could absorb part of an update by rescaling the grid rather than by moving along it, which would confound the mechanism being studied. QES itself requires only a fixed affine codebook on which rounding and the codebook bounds are well defined, so it is agnostic to how the scales were chosen and applies unchanged to AWQ (Lin et al., 2024) or SmoothQuant (Xiao et al., 2024) checkpoints. Activations are kept in FP for INT4/INT8 and quantized to INT8 in W8A8 via LLM-Compressor (RedHatAI & vLLM Project, 2024). Following the LLM-QAT convention (Liu et al., 2024), the language-model head, classifier, and scoring projections are excluded from quantization. The Stateless Seed Replay variant maintains a circular buffer of $(\mathcal{S}_\tau, \mathcal{F}_\tau)$ tuples for the past K generations; the Full Residual variant instead stores the FP16 residual $\boldsymbol{e}_t$ directly (Section 4).

A.2 SFT TASKS

We follow the prompt-based k-shot fine-tuning protocol of QuZO (Zhou et al., 2025) and MeZO (Malladi et al., 2024). RoBERTa-large is the backbone for all four tasks (SNLI, MNLI, RTE, SST-5); the W8 column of Table 1 uses the same backbone post-training-quantized with GPTQ (Frantar et al., 2023). We use $K_{\text{shot}} = 16$ examples per class, three random seeds (42, 13, 21) for the few-shot draws, and report the mean accuracy across seeds. Optimization uses a constant learning-rate schedule (no warmup, no weight decay), batch size 8, and maximum sequence length 128 for SST-5 and 256 for SNLI/MNLI/RTE. Final-checkpoint evaluation uses the LM-BFF template, verbalizer, and label-fitting pipeline.

Method-specific settings are as follows. The first-order FP32 upper bound uses AdamW at learning rate 1×10^{-5}. The first-order W8 baseline uses the same optimizer and learning rate, with weights snapped onto the W8 grid after each `optimizer.step()` (gradients pass through unchanged; this is equivalent to a post-step straight-through estimator). MeZO at FP32 uses $N = 2$ SPSA pairs, perturbation scale $\sigma = 1\times10^{-3}$, and learning rate 1×10^{-6}. QuZO uses $N = 8$ at a per-task learning rate found by grid search, with stochastic round-to-nearest after each step. QES uses $N = 8$, $\gamma = 0.9$, $\sigma = 1\times10^{-3}$, deterministic per-channel snap, and the per-task learning rate α, replay window K, and step budget shown in Table 3. Per-task values were tuned on seed 42 only and then frozen across the remaining two seeds.

Table 3: Per-task QES hyperparameters for the SFT experiments on RoBERTa-large (Table 1). Here α is the learning rate, K is the seed-replay window, and Steps is the total number of optimization steps. Population $N = 8$ antithetic pairs, decay $\gamma = 0.9$, and perturbation scale $\sigma = 1\times10^{-3}$ for all tasks.

Task	$\boldsymbol{\alpha}$	$\boldsymbol{K}$	**Steps**
SNLI	3×10^{-7}	16	1500
MNLI	5×10^{-7}	16	1500
RTE	1×10^{-6}	16	1000
SST-5	5×10^{-7}	16	1500

A.3 REASONING TASKS

For Countdown and GSM8K we use Qwen2.5-1.5B and Qwen2.5-3B (Qwen et al., 2025) as backbones, post-training-quantized to INT4 and INT8 with GPTQ (Frantar et al., 2023) and to W8A8 with LLM-Compressor (RedHatAI & vLLM Project, 2024). The reward is binary correctness following the GRPO-Zero prompt format (Policy-gradient, 2025); rollouts are evaluated on batches of problems sampled from the training split, and accuracy is reported on a held-out 400-problem evaluation set. Evaluation uses the same protocol as training. Prompting is zero-shot with a fixed template that requires the answer inside a delimited field, and a response is correct only under strict extraction, so training and evaluation signals coincide. This is stricter than the few-shot protocols reported in model technical reports, and the gap is widest for small models, because strict extraction also measures whether the model follows the answer format at all. Qwen2.5-1.5B rarely emits a parseable answer field even when the prompt asks for one, which is why its GSM8K base accuracy is near zero; under the identical harness, Qwen2.5-3B scores 48.45 and Llama-3.1-8B 64.14. The strict metric is kept because every method within a row of Table 2 is graded the same way, which keeps the comparison internally consistent. Each method is run for 300 generations with $N = 50$ antithetic pairs per generation. The σ and α values used by QES are listed in Table 4. The replay window is $K = 50$ for the seed-replay variant on the reasoning tasks (the $K = 20$ comparison in Table 7 is reported as an ablation), and $\gamma = 0.9$ throughout. The Full Residual variant uses the same σ and α as Seed Replay, with FP16 residuals replacing the seed-and-reward buffer.

Table 4: Per-configuration QES hyperparameters for the reasoning experiments (Table 2). Seed Replay and Full Residual share the same σ and α at each cell; the only difference between the two variants is whether residuals are stored in FP16 directly or rematerialized via seed replay. Population $N = 50$ antithetic pairs, decay $\gamma = 0.9$, and replay window $K = 50$ throughout.

Model	Format	σ	α
Qwen2.5-1.5B	INT4	1×10^{-2}	5×10^{-4}
	INT8	1×10^{-3}	1×10^{-4}
	W8A8	1×10^{-2}	1×10^{-3}
Qwen2.5-3B	INT4	5×10^{-3}	3×10^{-4}
	INT8	1×10^{-3}	1×10^{-4}
	W8A8	1×10^{-2}	1×10^{-3}



Figure 2: Countdown training curves for QuZO, QES, and Full-Residual QES against the Base Model. QuZO (orange) is unstable and frequently collapses on the INT4 lattice and on the 1.5B model. QES (green) tracks the Full-Residual oracle (blue) closely while using orders of magnitude less optimizer memory.

B Training Dynamics

C Scaling and Cross-Family Case Studies

The main experiments cover Qwen2.5 at 1.5B and 3B. Two further case studies test whether QES carries over to a different model family and to a much larger model. Llama-3.1-8B is fine-tuned on GSM8K, and Qwen2.5-32B-Instruct on Countdown. Both models were post-training-quantized to INT4 with GPTQ (Frantar et al., 2023) following the same protocol as the Qwen2.5 backbones in Section 4.1, and both reuse the Qwen2.5-3B (INT4) configuration verbatim: $N = 50$ antithetic pairs per generation, $K = 50$, $\gamma = 0.9$, and 300 generations. No per-model hyperparameter search was performed in either case.

Table 5: Accuracy (%) beyond the main experimental grid. Llama-3.1-8B tests a different model family; Qwen2.5-32B-Instruct tests a $10\times$ larger model. Both reuse the Qwen2.5-3B (INT4) hyperparameters without retuning.

Model	Family	Task	Base	QES
Llama-3.1-8B (INT4)	Llama-3.1	GSM8K	64.14	**82.64**
Qwen2.5-32B-Instruct (INT4)	Qwen2.5	Countdown	39.25	**51.40**

Table 5 reports both results. On Llama-3.1-8B, QES improves GSM8K accuracy from 64.14% to 82.64%, an 18.5-point gain. The result holds along two axes that the main experiments do not test: a different architecture family (Llama-3.1 vs. Qwen2.5) and a larger scale (8B vs. 1.5B/3B). On Qwen2.5-32B-Instruct, QES improves Countdown accuracy from 39.25% to 51.40%, a 12.15-point gain, at a scale an order of magnitude beyond the main grid. Together with the Qwen2.5 results in Table 2, these results suggest that the error-feedback mechanism is not backbone-specific and that its hyperparameters transfer across model families and scales without retuning.

The scaling pattern is also consistent with the intuition that larger models are easier to optimize on a discrete lattice. With more redundant parameters and smoother loss landscapes (Li et al., 2018), the search for valid descent directions becomes more stable at the 8B and 32B scales than at 1.5B, where every grid step traverses a sharper region of the loss surface. This intuition matches the trend in Table 2, where QES gains grow with model size on GSM8K.

The memory implications are particularly relevant at this scale. Storing FP16 residuals for the 8B model costs an extra 12.15 GB of VRAM (Table 9)—more than twice the quantized weights themselves, and beyond the optimizer-state budget on most consumer GPUs. By contrast, QES holds the optimizer state to $\sim$29.7 KB regardless of model scale, making INT4 fine-tuning of an 8B model feasible within the same memory envelope as inference.

D Detailed Ablation Results

This appendix provides the full tables for the ablations summarized in Sections 4.4 and 4.5. Table 6 compares Stateless Seed Replay against the Full Residual oracle on Countdown across all six (model, format) configurations. Table 7 reports the effect of the replay window K and decay factor γ (top) and the update- and boundary-hit ratios used in the fidelity analysis (bottom).

Sensitivity to the inherited hyperparameters. QES adds exactly two hyperparameters to standard ES: the decay factor γ and the replay window K. The rest—the perturbation scale σ, the learning rate α, and the population size N—are inherited from continuous ES for LLMs (Qiu et al., 2025), where they are reported to be robust to tuning. To check that this robustness survives the move to a discrete lattice, Qwen2.5-3B INT4 was re-run on Countdown with both σ and α doubled relative to Table 4. Accuracy was 31.65%, against 31.85% at the default setting: a 0.2-point difference, within run-to-run variation. Tuning effort is therefore best spent on γ, and, as Table 7 shows, hardly at all on K.

Table 6: Countdown accuracy (%): Stateless Seed Replay (QES) vs. Full Residual oracle. The seed-replay variant tracks the oracle within a few points across all configurations while reducing optimizer-state memory from gigabytes to kilobytes (Appendix E).

Model	Format	QES	Full Residual
Qwen2.5-1.5B	INT4	16.00	18.05
	INT8	26.35	22.10
	W8A8	15.35	15.25
Qwen2.5-3B	INT4	31.85	33.50
	INT8	37.40	33.30
	W8A8	21.35	31.70

E Memory and Compute Footprint

This appendix details the peak VRAM and wall-clock measurements summarized in Section 4. Section E.1 lists everything QES keeps beyond the quantized inference checkpoint, Section E.2 reports measured peak VRAM, and Section E.3 reports wall-clock time.

Table 7: Seed-replay ablations on Qwen2.5-1.5B INT4 (Countdown). **Top:** Replay window K and decay γ; *Scaled* sets $\gamma^K \approx 0$, *Fixed* uses γ=0.90. **Bottom:** Update ratio and boundary-hit ratio ρ in Stateless Seed Replay; reconstruction errors require both to coincide and are therefore vanishingly rare.

WINDOW K	DECAY γ	ACCURACY (%)
	Scaled decay	
50	0.90	16.00
40	0.87	14.00
30	0.83	14.15
20	0.78	10.80
10	0.58	4.55
	Fixed decay	
50	0.90	16.00
40	0.90	14.80
30	0.90	16.15
20	0.90	14.75
10	0.90	13.05

QUANTIZATION	UPDATE RATIO	HIT RATIO (ρ)
INT4	$\approx 10^{-2}$	$< 1 \times 10^{-5}$
INT8	$\approx 10^{-2}$	$\approx 1.4 \times 10^{-4}$
W8A8	$\approx 10^{-2}$	$\approx 6 \times 10^{-6}$

E.1 WHAT QES STORES

Table 8 is a complete inventory of the state QES maintains beyond the quantized inference checkpoint. There is no momentum or second-moment buffer, no stored gradients, no stored activations, no full-precision master copy of the weights, and no dense residual tensor. This is the entire difference between QES and pure quantized inference, and it is why the QES and QuZO columns of Table 9 coincide.

Table 8: State maintained by QES beyond the quantized inference checkpoint. Only the history buffer is persistent, and its size depends on K and N rather than on the parameter count d.

STATE	CONTENTS	SIZE	SCALES WITH
HISTORY BUFFER $\mathcal{H}$	K SEEDS + $K \times N$ REWARDS	29.7 KB (K=N=50)	K, N (NOT d)
REPLAY SCRATCH BUFFER	ONE CHUNK OF $\boldsymbol{\delta}, \boldsymbol{e}, \boldsymbol{u}$	20–70 MB, FREED PER CHUNK	CHUNK SIZE
QUANTIZATION SCALES	PER-OUTPUT-CHANNEL, FP16	ALREADY IN THE CHECKPOINT	—

The replay scratch buffer deserves a note, because it is the one quantity that is materialized rather than stored. The residual recursion in Equations (6)–(8) is coordinate-wise: each parameter's residual depends only on its own history. Replay is therefore executed chunk by chunk, with parameter blocks in the outer loop and the K replay steps in the inner loop (Section 3.3). For a given chunk, only that chunk's perturbation $\boldsymbol{\delta}$ and its proxy residual are materialized; the K-step recursion is run, the integer update is committed, and the buffer is freed before moving on. The transient working set is therefore bounded by the chunk size rather than by d, and it is the same kind of scratch buffer that QuZO and any other ES method already needs in order to apply a perturbation. The figures in Table 9 are measured peak VRAM over the full training loop, including these transient buffers, not an analytic count of persistent tensors.

Table 9: Peak VRAM. QES matches QuZO's inference-only footprint (optimizer state ≈29.7 KB across all settings); Full Residual adds gigabytes of FP16 buffers, and the gap widens with scale.

		Model	Total VRAM (GB)		
Model	Fmt.	Wts. (GB)	QuZO	Full Res.	QES
Qwen2.5-1.5B	INT4	1.071	1.071	3.511	**1.071**
	INT8	1.686	1.686	4.126	**1.686**
	W8A8	2.091	2.091	4.532	**2.091**
Qwen2.5-3B	INT4	1.926	1.926	7.094	**1.926**
	INT8	3.228	3.228	8.396	**3.228**
	W8A8	3.746	3.746	8.914	**3.746**
Llama-3.1-8B	INT4	5.190	5.190	17.340	**5.190**

Table 10: Per-iteration wall-clock time on H100 GPUs (INT4, N=50, K=50). Under the 8-GPU budget the rollouts are sharded across all eight GPUs, and the residual reconstruction is sharded across the same devices during the update phase.

		1 GPU (s)		8 GPUs (s)	
Model	Method	Rollout	Update	Rollout	Update
Qwen2.5-1.5B	QuZO	1784	7.02	223	7.02
	Full Residual	1784	7.40	223	7.40
	QES	1784	115.6	223	9.35
Qwen2.5-3B	QuZO	3174	16.85	429	16.85
	Full Residual	3174	15.50	429	18.50
	QES	3174	264.7	429	16.88
Llama-3.1-8B	QuZO	4014	43.29	531	43.29
	Full Residual	4014	37.30	531	37.30
	QES	4014	565.9	531	44.86

E.2 Memory

Table 9 reports peak VRAM for the model weights and the optimizer state across methods. QES holds optimizer state to a constant ≈29.7 KB across all configurations—the size of the seed-and-reward history buffer—and therefore matches QuZO's total footprint, which is itself the inference-only baseline. Full Residual, by contrast, must materialize FP16 residuals for every parameter, adding 2.44 GB at 1.5B, 5.17 GB at 3B, and 12.15 GB at 8B, which effectively undoes the memory benefit of quantization. For reference, standard Quantization-Aware Training (QAT) on a 3B model requires over 24 GB of VRAM for full-precision weights, gradients, activations, and optimizer states—roughly 13× the QES footprint at INT4.

E.3 Wall-clock Time

Stateless Seed Replay introduces a reconstruction step whose cost scales linearly with the window K. All measurements below were taken on H100 GPUs; reasoning rollouts use vLLM 0.17.1, and the SFT rollouts use plain PyTorch. The reasoning runs use INT4, $N = 50$, and $K = 50$; the SFT runs use $N = 8$ and $K = 16$ (Appendix A.2).

Table 10 reports per-iteration rollout and update times under two hardware budgets. With everything on a single GPU, the QES update is 13–17× slower than the QuZO update, because it replays K generations sequentially. End to end this is largely amortized, since rollouts dominate the iteration: QES adds 6.0% (1.5B), 7.8% (3B), and 13.0% (8B) to total iteration time relative to the Full Residual oracle.

Sharding the rollouts across eight GPUs removes that amortization, so the sequential update becomes the bottleneck. Two levers remove it again.

Table 11: Per-step cost of the SFT experiments (RoBERTa-large, N=8, K=16, one H100). Inference time is identical across methods because all three run the same forward passes.

METHOD	OPTIMIZER STATE	UPDATE (S)	INFERENCE (S)
QUZO	0	0.805	0.210
FULL RESIDUAL	576 MIB	0.885	0.210
QES	0.02 MIB	2.030	0.210

- **Parallel reconstruction.** No inference traffic is served during the update phase, so the pre-allocated KV-cache memory and the inference GPUs are both idle. Because the residual recursion is coordinate-wise, reconstruction can be sharded across those devices, with the parameters updated together in batches; the extra memory sits well under the freed KV-cache allocation. With this, the QES update is on par with QuZO and Full Residual at every scale (Table 10, right), and adds about 1% to end-to-end iteration time over QuZO while retaining the 29.7 KB optimizer state. The time-for-space trade is therefore an artifact of a sequential implementation rather than something intrinsic to seed replay.
- **Tunable window size.** Reconstruction cost is linear in K, and accuracy degrades gracefully as K shrinks at fixed decay (Section 4.4). Moving from K=50 to K=20 costs 1.25 accuracy points and cuts single-GPU update time by 37–43%: 115.6 $\rightarrow$ 71.67 s at 1.5B, 264.7 $\rightarrow$ 150.88 s at 3B, and 565.9 $\rightarrow$ 354.00 s at 8B. The reduction is less than the 60% cut in replay work because part of the update cost is independent of K.

The SFT setting is smaller and shows the same pattern (Table 11): QES adds 1.2 s of update time per step and 0.02 MiB of state, against the 576 MiB of FP16 residuals that the oracle must hold.

The actual penalty is therefore highly dependent on the inference engine and target hardware, rather than being a strict linear multiplier on training time.

F CONTINUOUS OPTIMIZATION IN THE DISCRETE GRID

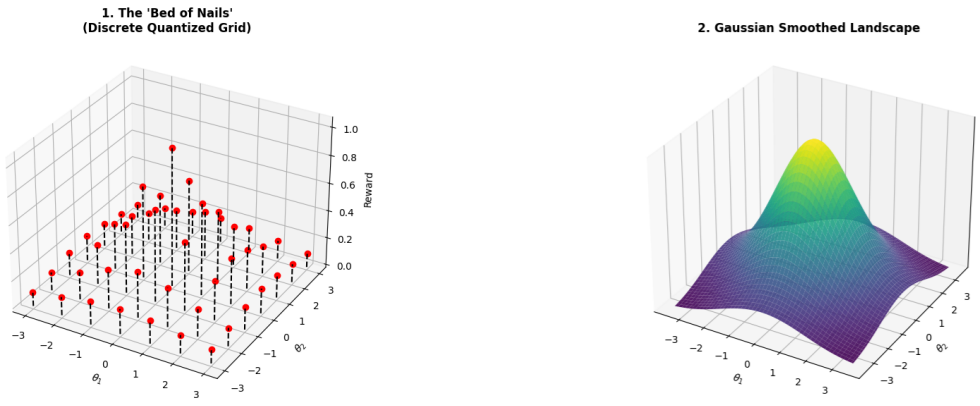

Figure 3: A continuous reward function and its instantiation on a discrete grid. While the reward function can be optimized with a high-precision gradient ascent, that method cannot be applied to the discrete case. Instead, the gradient needs to be estimated without vanishing signals or inaccurate moves between grid points. That is the challenge that QES is designed to solve.

G PROOFS AND EXTENDED ANALYSIS

This appendix states and proves the results summarized in Section 5, gives the general-γ versions, and lists what the analysis does and does not cover.

G.1 SETUP

All statements are for the QES recursion of Equations (6)–(8), started from $\boldsymbol{e}_0 = \mathbf{0}$:

$$\boldsymbol{u}_t = \alpha\hat{\boldsymbol{g}}_t + \gamma\boldsymbol{e}_t, \qquad \Delta\mathbf{W}_t = \mathrm{Round}(\boldsymbol{u}_t), \qquad \boldsymbol{e}_{t+1} = \boldsymbol{u}_t - \Delta\mathbf{W}_t, \qquad \mathbf{W}_{t+1} = \mathbf{W}_t + \Delta\mathbf{W}_t.$$

Round is round-to-nearest on a lattice of spacing Δ, applied coordinatewise, with ties resolved away from zero; $\Delta = 1$ in integer code units, and with per-output-channel scales all statements are read per quantization group. The reward is maximized, so all updates are ascent steps. Write $\mathbb{E}_t$ for the expectation conditional on everything up to step t, and $\boldsymbol{\Theta}_t \triangleq \mathbf{W}_t + \boldsymbol{e}_t$, so that $\boldsymbol{\Theta}_0 = \mathbf{W}_0$. Residual and trajectory statements use the ℓ_∞ norm, where they are exact and free of dimension factors, because Round acts coordinatewise and Δ is a per-coordinate quantity. The convergence bound is a smoothness argument and is therefore stated in ℓ_2, with the dimension factor written out: $\|\boldsymbol{e}_t\|_\infty \le \Delta/2$ becomes $\|\boldsymbol{e}_t\|_2 \le \sqrt{d}\,\Delta/2$.

Propositions 1–3 and Lemma 1 are algebraic and use no assumptions. The assumptions below are consumed only by Theorem 1:

- (A1) $J_\sigma(\mathbf{W}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[J(\mathbf{W} + \sigma\boldsymbol{\epsilon})]$, the Gaussian-smoothed reward of Section 5, is L-smooth. Rounding $\sigma\boldsymbol{\epsilon}$ onto the lattice in Equation (3) perturbs that law, and its effect is carried by $\boldsymbol{b}_t$ in (A2).
- (A2) $\mathbb{E}_t[\hat{\boldsymbol{g}}_t] = \nabla J_\sigma(\mathbf{W}_t) + \boldsymbol{b}_t$ with $\|\boldsymbol{b}_t\|_2 \le \beta$, and $\mathbb{E}_t\|\hat{\boldsymbol{g}}_t\|_2^2 \le G^2$.
- (A3) $J^\star \triangleq \sup_{\mathbf{W}} J_\sigma(\mathbf{W}) < \infty$; for a binary-correctness reward, $J^\star \le 1$.
- (A4) No update is clipped by the codebook bounds along the trajectory.

G.2 EXACT ERROR FEEDBACK

Proposition 1. For every step t and every γ,

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t + \alpha\hat{\boldsymbol{g}}_t - (1-\gamma)\,\boldsymbol{e}_t, \qquad \|\boldsymbol{e}_t\|_\infty \le \Delta/2. \tag{14}$$

Telescoping gives, for every T,

$$\mathbf{W}_T = \mathbf{W}_0 + \sum_{t<T} \alpha\hat{\boldsymbol{g}}_t - \boldsymbol{e}_T - (1-\gamma)\sum_{t<T}\boldsymbol{e}_t, \tag{15}$$

so the deviation from the ideal continuous trajectory obeys $\|\mathbf{W}_T - (\mathbf{W}_0 + \sum_{t<T}\alpha\hat{\boldsymbol{g}}_t)\|_\infty \le (\Delta/2)(1 + (1-\gamma)(T-1))$, which is linear in T unless $\gamma = 1$, and equal to $\|\boldsymbol{e}_T\|_\infty \le \Delta/2$ when $\gamma = 1$.

Proof. By definition of $\boldsymbol{\Theta}$ and the two update rules,

$$\boldsymbol{\Theta}_{t+1} = \mathbf{W}_{t+1} + \boldsymbol{e}_{t+1} = (\mathbf{W}_t + \Delta\mathbf{W}_t) + (\boldsymbol{u}_t - \Delta\mathbf{W}_t) = \mathbf{W}_t + \boldsymbol{u}_t,$$

in which $\Delta\mathbf{W}_t$ cancels identically, so no property of Round is used. Substituting $\boldsymbol{u}_t = \alpha\hat{\boldsymbol{g}}_t + \gamma\boldsymbol{e}_t$ and $\mathbf{W}_t = \boldsymbol{\Theta}_t - \boldsymbol{e}_t$ gives Equation (14). For the residual bound, $\boldsymbol{e}_{t+1} = \boldsymbol{u}_t - \mathrm{Round}(\boldsymbol{u}_t)$ is by construction the leftover of a round-to-nearest operation, so $\|\boldsymbol{e}_{t+1}\|_\infty \le \Delta/2$ for every $t \ge 0$, every γ, and every step size, while $\|\boldsymbol{e}_0\|_\infty = 0$ by initialization. Telescoping and using $\boldsymbol{\Theta}_0 = \mathbf{W}_0$ and $\mathbf{W}_T = \boldsymbol{\Theta}_T - \boldsymbol{e}_T$ gives Equation (15); since $\boldsymbol{e}_0 = \mathbf{0}$, its last sum has at most $T-1$ nonzero terms. □

G.3 THE DEAD ZONE

Proposition 2. Fix a coordinate carrying a persistent update signal $\alpha\hat{g}_{t,i} = a$ for all t. While no update fires, the recursion reduces to a leaky integrator with fixed point $a/(1-\gamma)$. The coordinate moves after finitely many steps, and then keeps moving indefinitely, if and only if

$$|a| > (1-\gamma)\Delta/2, \qquad \text{that is,} \qquad |\hat{g}_i| > g_{\min} \triangleq \frac{(1-\gamma)\Delta}{2\alpha}. \tag{16}$$

The first update occurs at

$$t_{\text{fire}} = \left\lceil \frac{\log\big(1 - (1-\gamma)\Delta/(2|a|)\big)}{\log\gamma} \right\rceil - 1 \;\; (\gamma < 1), \qquad t_{\text{fire}} = \left\lceil \frac{\Delta}{2|a|} \right\rceil - 1 \;\; (\gamma = 1), \tag{17}$$

and the mean period p between updates satisfies

$$\frac{\Delta}{|a| + (1-\gamma)\Delta/2} \;\le\; p \;\le\; \frac{\Delta}{|a| - (1-\gamma)\Delta/2}, \tag{18}$$

which collapses to $p = \Delta/|a|$ exactly at $\gamma = 1$.

Proof. Coordinates decouple, so fix one and drop the index; take $a > 0$ without loss of generality. While no update has fired, $\Delta W_\tau = 0$ and the recursion is the leaky integrator $e_{\tau+1} = a + \gamma e_\tau$ with $e_0 = 0$, whose solution is $e_{t+1} = u_t = e^\star(1 - \gamma^{t+1})$ with $e^\star = a/(1-\gamma)$ for $\gamma < 1$, and $u_t = (t+1)a$ for $\gamma = 1$, where $e^\star$ is unbounded. In both cases u_t increases strictly, and for $\gamma < 1$ it approaches $e^\star$ without reaching it.

An update fires at the first t with $u_t \ge \Delta/2$. If $a \le (1-\gamma)\Delta/2$ then $e^\star \le \Delta/2$ and $u_t < \Delta/2$ for every t, so the coordinate never moves: the dead zone is closed, and a signal at exactly the threshold is frozen as well. If $a > (1-\gamma)\Delta/2$, solving $u_t \ge \Delta/2$ for the smallest such t gives Equation (17). Updates then recur indefinitely, because after a fire the residual satisfies $|e| \le \Delta/2$, and rerunning the argument from any such starting value gives $u_t \to e^\star$, which exceeds $\Delta/2$ whenever the firing condition holds. The iteration therefore cannot fire once and stall.

For the period, summing Equation (15) over a horizon T shows that the lattice advances at the average rate $a - (1-\gamma)\bar{e}$, where $\bar{e}$ is the time-averaged residual. Since $|\bar{e}| \le \Delta/2$, the mean period obeys Equation (18). The direction of the deviation at $\gamma < 1$ depends on the sign of $\bar{e}$, which is set by the geometry of the orbit rather than by the sign of a, so only the two-sided bracket holds. □

Setting $\gamma = 0$ recovers stateless round-to-nearest, with $g_{\min} = \Delta/(2\alpha)$; setting $\gamma = 0.9$ lowers the threshold by a factor of ten; and $\gamma = 1$ removes it. This is the formal version of the stagnation problem: below $g_{\min}$, stagnation is not slow convergence but a fixed point of the dynamics, with $\mathbf{W}_T = \mathbf{W}_0$ for every T.

G.4 Deviation from the Ideal Trajectory

The next result compares the three update rules on a common gradient sequence. Fix one sequence $(\hat{\boldsymbol{g}}_t)_{t<T}$ shared by all three and let $\boldsymbol{D}_T \triangleq \mathbf{W}_T - (\mathbf{W}_0 + \sum_{t<T} \alpha \hat{\boldsymbol{g}}_t)$ be the deviation from the ideal continuous trajectory.

Proposition 3. Under stateless round-to-nearest, if $\|\alpha \hat{\boldsymbol{g}}_t\|_\infty < \Delta/2$ for all $t < T$, then $\Delta \mathbf{W}_t = \mathbf{0}$, $\mathbf{W}_T = \mathbf{W}_0$, and $\boldsymbol{D}_T = -\sum_{t<T} \alpha \hat{\boldsymbol{g}}_t$: the deviation is deterministic and equals the entire intended update. Under stochastic rounding, $\mathbb{E}[\boldsymbol{D}_T] = \mathbf{0}$ but $\mathrm{Var}(D_{T,i}) = \sum_{t<T} \mathbb{E}[\,|\alpha \hat{g}_{t,i}|(\Delta - |\alpha \hat{g}_{t,i}|)\,]$, so its standard deviation grows as $\Theta(\sqrt{T})$ and is unbounded. Under QES at $\gamma = 1$, $\boldsymbol{D}_T = -\boldsymbol{e}_T$, which is determined by the trajectory and satisfies $\|\boldsymbol{D}_T\|_\infty \le \Delta/2$ for every T.

Proof. The first case is immediate, since round-to-nearest of a value below $\Delta/2$ in magnitude returns zero. The second follows because unbiased rounding of a value u to the two adjacent lattice points returns the far point with probability $|u|/\Delta$, so the injected error has mean zero and variance $|u|(\Delta - |u|)$, and the per-step errors are independent. The third is Proposition 1 at $\gamma = 1$. □

The second case is the reason unbiasedness alone is not enough. For a coordinate with a persistent signal $a = \alpha \hat{g}_i \ll \Delta$, the accumulated signal after T steps is Ta while the rounding noise has standard deviation $\sqrt{Ta\Delta}$, so their ratio is $\sqrt{Ta/\Delta}$ and reaches one only after $T \approx \Delta/a$ steps. That is twice the number of steps the deterministic accumulator needs to produce its first update, by Equation (17). In a fine-tuning run of a few hundred generations, most coordinates never reach that point, so stochastic rounding spends the entire run inside its own noise while error feedback has already moved the weights along the ideal path.

G.5 Convergence

Let $\rho \triangleq \sup_t \|\boldsymbol{e}_t\|_2 \le \sqrt{d}\,\Delta/2$ be the Euclidean radius of the residual, that is, half a grid step per coordinate.

Theorem 1. Under (A1)–(A4) with $\gamma = 1$, for any constant $\alpha > 0$ and every $T \geq 1$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\|\nabla J_\sigma(\boldsymbol{\Theta}_t)\big\|_2^2 \;\leq\; \underbrace{\frac{2\,(J^\star - J_\sigma(\mathbf{W}_0))}{\alpha T} + \alpha L G^2}_{\text{standard ES rate}} \;+\; \underbrace{\big(L\rho + \beta\big)^2}_{\text{grid resolution}}, \tag{19}$$

where the first two terms are $O(1/\sqrt{T})$ at $\alpha = \Theta(1/\sqrt{T})$.

Proof. Write $\boldsymbol{a}_t \triangleq \nabla J_\sigma(\boldsymbol{\Theta}_t)$. At $\gamma = 1$, Proposition 1 gives $\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t + \alpha\hat{\boldsymbol{g}}_t$, so (A1) yields

$$J_\sigma(\boldsymbol{\Theta}_{t+1}) \;\geq\; J_\sigma(\boldsymbol{\Theta}_t) + \alpha\langle \boldsymbol{a}_t, \hat{\boldsymbol{g}}_t\rangle - \tfrac{L}{2}\alpha^2\|\hat{\boldsymbol{g}}_t\|_2^2.$$

Take the conditional expectation. By (A2) and $\boldsymbol{\Theta}_t - \mathbf{W}_t = \boldsymbol{e}_t$ together with L-smoothness, $\|\nabla J_\sigma(\mathbf{W}_t) - \boldsymbol{a}_t\|_2 \leq L\|\boldsymbol{e}_t\|_2$, so

$$\alpha\,\mathbb{E}_t\langle \boldsymbol{a}_t, \hat{\boldsymbol{g}}_t\rangle = \alpha\|\boldsymbol{a}_t\|_2^2 + \alpha\langle \boldsymbol{a}_t, \nabla J_\sigma(\mathbf{W}_t) - \boldsymbol{a}_t + \boldsymbol{b}_t\rangle \;\geq\; \alpha\|\boldsymbol{a}_t\|_2^2 - \alpha\|\boldsymbol{a}_t\|_2\big(L\|\boldsymbol{e}_t\|_2 + \beta\big).$$

Applying $AB \leq \frac{1}{2}A^2 + \frac{1}{2}B^2$ with $A = \|\boldsymbol{a}_t\|_2$ and $B = L\|\boldsymbol{e}_t\|_2 + \beta$ consumes exactly half of the first term and leaves

$$\mathbb{E}_t\big[J_\sigma(\boldsymbol{\Theta}_{t+1})\big] \;\geq\; J_\sigma(\boldsymbol{\Theta}_t) + \tfrac{\alpha}{2}\|\boldsymbol{a}_t\|_2^2 - \tfrac{\alpha}{2}\big(L\|\boldsymbol{e}_t\|_2 + \beta\big)^2 - \tfrac{L}{2}\alpha^2 G^2.$$

Taking total expectations, summing over $t = 0, \ldots, T-1$, telescoping, and using $J_\sigma(\boldsymbol{\Theta}_T) \leq J^\star$ with $\boldsymbol{\Theta}_0 = \mathbf{W}_0$ gives

$$\tfrac{\alpha}{2}\sum_{t<T}\mathbb{E}\|\boldsymbol{a}_t\|_2^2 \;\leq\; J^\star - J_\sigma(\mathbf{W}_0) + \tfrac{\alpha}{2}\sum_{t<T}\mathbb{E}\big(L\|\boldsymbol{e}_t\|_2 + \beta\big)^2 + T\tfrac{L}{2}\alpha^2 G^2.$$

Multiplying by $2/(\alpha T)$ and using $\|\boldsymbol{e}_t\|_2 \leq \rho$ gives Equation (19). The constants are 2, 1, and 1, with nothing hidden. Balancing the first two terms gives $\alpha^\star = \sqrt{2(J^\star - J_\sigma(\mathbf{W}_0))/(LG^2T)}$ and a leading term $2\sqrt{2LG^2(J^\star - J_\sigma(\mathbf{W}_0))/T}$. □

QES therefore converges at the usual ES rate until the gradient becomes as small as the one thing a quantized optimizer cannot resolve, namely the grid itself: the weights sit up to half a step per coordinate away from the point at which the gradient is measured, which by smoothness is a residual gradient of at most $L\rho$. Every method that keeps its weights on the grid pays this floor, and the floor does not shrink with α, which is what distinguishes lattice-constrained error feedback from error feedback in continuous space, where the residual itself is $O(\alpha)$.

For $\gamma < 1$ write $\nu \triangleq 1 - \gamma$ and $G_{\min} \triangleq \nu\rho/\alpha$, the Euclidean counterpart of $g_{\min}$ in Equation (16). The step is now $\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}_t = \alpha\hat{\boldsymbol{g}}_t - \nu\boldsymbol{e}_t$. Repeating the argument with $AB \leq \frac{\alpha}{2}A^2 + \frac{1}{2\alpha}B^2$ and $\|\alpha\hat{\boldsymbol{g}}_t - \nu\boldsymbol{e}_t\|_2^2 \leq 2\alpha^2\|\hat{\boldsymbol{g}}_t\|_2^2 + 2\nu^2\|\boldsymbol{e}_t\|_2^2$ gives

$$\frac{1}{T}\sum_{t<T}\mathbb{E}\|\nabla J_\sigma(\boldsymbol{\Theta}_t)\|_2^2 \;\leq\; \frac{2(J^\star - J_\sigma(\mathbf{W}_0))}{\alpha T} + 2\alpha LG^2 + \big(L\rho + G_{\min} + \beta\big)^2 + 2\alpha L\,G_{\min}^2. \tag{20}$$

The dead zone therefore enters the floor exactly as the extra term $G_{\min}$, which vanishes at $\gamma = 1$ and recovers Equation (19) up to the factor 2 on αLG^2; that factor comes from splitting the squared step and can be dropped at $\gamma = 1$, where the step is simply $\alpha\hat{\boldsymbol{g}}_t$. One caveat on step sizes: taking $\alpha = \Theta(1/\sqrt{T})$ makes the first two terms $O(1/\sqrt{T})$ but drives $G_{\min} \propto 1/\alpha$ upward, so the $O(1/\sqrt{T})$ reading is valid at $\gamma = 1$, or under the coupling $1 - \gamma = O(\alpha L)$. For a fixed $\gamma = 0.9$ the correct reading is a constant step size with a constant floor.

Theorem 1 bounds the gradient at the virtual iterate $\boldsymbol{\Theta}_t$, while the model that is served is $\mathbf{W}_t$. The two are within ρ of each other, so $\|\nabla J_\sigma(\mathbf{W}_t)\|_2^2 \leq 2\|\nabla J_\sigma(\boldsymbol{\Theta}_t)\|_2^2 + 2L^2\|\boldsymbol{e}_t\|_2^2$ and

$$\frac{1}{T}\sum_{t<T}\mathbb{E}\|\nabla J_\sigma(\mathbf{W}_t)\|_2^2 \;\leq\; \frac{4(J^\star - J_\sigma(\mathbf{W}_0))}{\alpha T} + 2\alpha LG^2 + 2\big(L\rho + \beta\big)^2 + 2L^2\rho^2, \tag{21}$$

a floor of the same order. This is the price of keeping the weights on the lattice: the iterate can only ever reach the nearest lattice point to a stationary point, and that point generically has a nonzero gradient of exactly this order.

G.6 Fidelity of the Replayed Residual

Stateless Seed Replay does not use $\boldsymbol{e}_t$ but a proxy $\tilde{\boldsymbol{e}}_t$ obtained by rerunning the recursion over the last K steps from a zero initial state. The following bounds the gap.

Lemma 1. Let $\boldsymbol{d}_\tau \triangleq \boldsymbol{e}_\tau - \tilde{\boldsymbol{e}}_\tau$. On the event that the exact and replayed recursions round into the same lattice cell at every step of the window, $\boldsymbol{d}_{\tau+1} = \gamma \boldsymbol{d}_\tau$ coordinatewise, and therefore $\|\boldsymbol{e}_t - \tilde{\boldsymbol{e}}_t\|_\infty \le \gamma^K \Delta/2$.

Proof. The two recursions share $\hat{\boldsymbol{g}}_\tau$, so $\tilde{\boldsymbol{u}}_\tau - \boldsymbol{u}_\tau = -\gamma \boldsymbol{d}_\tau$. If $\mathrm{Round}(\tilde{\boldsymbol{u}}_\tau) = \mathrm{Round}(\boldsymbol{u}_\tau)$ then the rounded parts cancel in the difference of the residual updates, leaving $\boldsymbol{d}_{\tau+1} = \boldsymbol{u}_\tau - \tilde{\boldsymbol{u}}_\tau = \gamma \boldsymbol{d}_\tau$. At the start of the window $\tilde{\boldsymbol{e}} = \boldsymbol{0}$ and $\|\boldsymbol{e}\|_\infty \le \Delta/2$, so $\|\boldsymbol{d}\|_\infty \le \Delta/2$ there, and K contractions give the bound. □

The gap is therefore geometric in K, with $\gamma^K \Delta/2$ equal to 2.6×10^{-3} at the reasoning setting ($\gamma = 0.9$, $K = 50$), 6.1×10^{-2} at $K = 20$, and 9.3×10^{-2} at the SFT setting ($K = 16$), against a residual range of $\Delta/2$. A coordinate whose u happens to lie within $\gamma^K \Delta/2$ of a rounding threshold instead flips its decision and differs by a full lattice step; how often that happens is bounded by the joint update and boundary-hit rates measured in Table 7, which is why the seed-replay variant tracks the Full Residual oracle so closely in Table 6. The contraction also requires $\gamma < 1$ strictly, which is the formal reason Stateless Seed Replay cannot be run at $\gamma = 1$ while the Full Residual variant can.