

GNMR: Runtime Stability Control for Low-Precision Large Language Model Training

Boao Kong^{1*} Weichen Jia^{1*} Engao Zhang^{1*} Guohong Li^{1*}
Yonghan Dong² Yao Wang² Yaoyuan Wang² Yunke Peng² Kun Yuan^{1†}

¹Peking University ²Huawei Technologies Ltd.
{kongboao, 2501210086, 2300012403, 2601210091}@stu.pku.edu.cn
kunyuan@pku.edu.cn

Abstract

Training stability is a key bottleneck in low-precision language model training: efficient low-cost paths can still produce short-lived numerical risks at a small set of operators. We formulate this as runtime stability control and present **Gradient Norm-to-Mean Ratio (GNMR)**, a lightweight controller that compares each recoverable unit’s current gradient norm with its historical mean. Together with Δ -GNMR for abrupt short-window increases, GNMR maps local risk signals to bounded recovery actions under a hard maxO budget and a short lock interval, without changing the numerical format, kernel, or backend recipe. Across LLaMA-style and Qwen-style activation-quantization stress, DeepSeek-style recipe-level training, GPT-MoE and PanGu stress tests, GNMR preserves high-fidelity quality with selective recovery under a hard active-set budget. These results support GNMR as a backend-agnostic controller to improve low-precision training stability while preserving low-cost execution.

1 Introduction

Training stability is a central constraint in training and adaptation of large language models (LLMs). To reduce computation, memory, and communication costs, modern training stacks use low-cost execution paths. These paths may appear as mixed-precision training (Micikevicius et al., 2017), FP8 training recipes (Peng et al., 2023), compression of activations or optimizer states (Xi et al., 2024), or quantized adaptation (Dettmers et al., 2023). Previous work has shown that such paths can preserve acceptable model quality when combined with suitable formats, scaling rules, outlier handling, and high-fidelity safeguards. The question we study is not whether low-cost execution is useful on average.

Instead, we ask whether training can detect and recover from local stability risks when low-precision perturbations become harmful at runtime.

From a stability angle, low-precision perturbations do not affect all layers, operators, and training steps equally. Most low-cost execution may be safe for most steps, while a few local units can become risky in short windows. LLM.int8 (Dettmers et al., 2021) identifies outlier features as a key difficulty in quantization of LLMs, and SmoothQuant (Xiao et al., 2023) treats activation outliers as a central obstacle to accurate low-bit execution. Reduced-precision training can be sensitive to stability and hyperparameter choices (Lee et al., 2024). More broadly, loss and gradient spikes are known failure modes in large-scale pre-training (Takase et al., 2023) and have motivated optimizer-level stabilization methods (Huang et al., 2025b).

These observations make stability under low precision a runtime control problem, not only a static configuration problem. A layer may be stable for many updates, become risky for a few steps when gradients, activation statistics, scaling factors, or optimizer states shift, and then return to a stable regime. In the same step, most of the other layers may remain safe. A useful controller should therefore identify which local units are becoming unstable, decide whether recovery is needed now, and apply recovery under a fixed overhead budget.

Existing work addresses low-precision numerical issues in several directions. Low-bit training recipes extend practical execution to FP4-style regimes (Wang et al., 2025) and NVFP4-style regimes (Abecassis et al., 2025). Training stability methods include gradient clipping (Pascanu et al., 2013), gradient-noise-scale analysis (McCandlish et al., 2018), and GSNR-style signals (Jiang et al., 2023). Dynamic precision methods study adaptive bit-width choices over time or structure (Yu et al., 2022). These methods are valuable, but they mainly focus on the format, recipe, optimizer, or offline

*Equal contribution.

†Corresponding author.

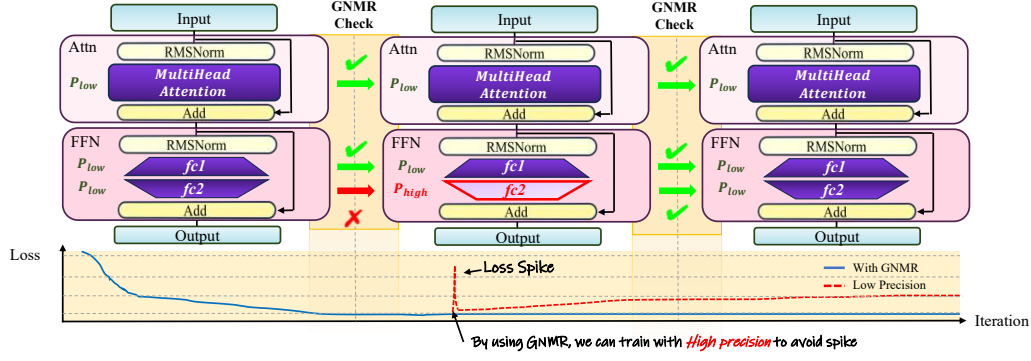


Figure 1: Overview of GNMR as a runtime stability controller. GNMR monitors local risk signals and routes selected recoverable units from the low-cost path to the higher-fidelity recovery path under a fixed active budget.

calibration level. They do not directly answer the controller-level question studied here: given an existing low-cost path and a more accurate recovery path, how should training decide online which local units to recover under a fixed budget?

1.1 Challenges in Runtime Stability Control under Low Precision

The stability-control view above leads to four requirements that existing low-precision approaches do not jointly satisfy.

(L1) Localized and non-stationary runtime risk. Low-precision instability is rarely uniform across all layers and steps. A unit may be safe for most of training process but risky during a short window. Static or semi-static strategies may miss such transient risks, or protect too much of the model with conservative safeguards.

(L2) Local and lightweight risk signals. Existing stability indicators include loss spikes (Takase et al., 2023), global gradient norms (Pascanu et al., 2013), gradient-noise statistics (McCandlish et al., 2018), GSNR-style signals (Jiang et al., 2023), and outlier statistics (Xiao et al., 2023). However, a runtime controller needs more than a general warning signal. It needs a signal that is cheap to compute, available in every single step, local enough to identify the risky unit, and comparable across modules.

(L3) Budgeted and stable intervention. Recovery actions are not free. Sending more units through a higher-fidelity path can increase peak memory, casting overhead, and routing complexity. Frequent switching can also cause thrashing. A controller must therefore decide which units are risky, how many can be recovered at each step, and how long recovery should persist.

(L4) Separation between control quality and

backend implementation. Low-precision studies often evaluate complete systems where formats, kernels, scaling rules, optimizers, and fallback policies change together. Such evaluations are useful, but they can hide the controller question. To evaluate runtime stability control, we need to separate trigger quality, budget behavior, recipe-level integration, and large-model boundedness from claims about new kernels or universal hardware speedups.

1.2 Contributions

To address these challenges, we present **GNMR**, short for **Gradient Norm-to-Mean Ratio**, a runtime stability controller for low-precision language model training, as illustrated in Figure 1. GNMR monitors operator-wise gradient deviations and applies temporary high-fidelity recovery only to risky low-precision-eligible units under a fixed budget. Its core contribution is a stability-aware risk monitor and a budgeted closed-loop control policy operating over existing low-cost and recovery paths. Our contributions can be summarized as:

(C1) Runtime stability-control formulation under low precision. We formulate local instability in low-precision language model training as a constrained online control problem. Given a low-cost execution path and a higher-fidelity recovery path, the controller observes runtime statistics of eligible units and selects a budget-bounded subset of eligible units for temporary recovery. This formulation unifies risk identification, recovery decisions, and budget constraints while decoupling the control policy from backend implementation. **This addresses (L1) and (L4).**

(C2) Dual-timescale GNMR risk monitor. We design GNMR as a lightweight operator-wise statistic that compares a unit’s current gradient norm with

its historical mean, capturing long-horizon deviations from typical training behavior. We further introduce Δ -GNMR, which normalizes GNMR by a short recent window to detect abrupt local changes. Together, they provide a real-time, dimensionless, and local risk signal for runtime control. **This addresses (L2).**

(C3) Budgeted recovery policy with analysis. We design a policy that recovers only top-risk units, uses maxO to control instantaneous overhead, and applies a short T_{lock} interval to reduce thrashing. We also provide a one-step expected-descent analysis showing why selective recovery can reduce low-precision perturbation on critical updates. **This addresses (L3).**

(C4) Controller-level evidence across low-precision settings. We evaluate GNMR in controlled LLaMA-style and Qwen2-like activation-quantization stress benches, a TE-backed low-precision training setting (Perez et al., 2023), GPT-MoE and PanGu architecture stresses. Exact-budget ranking comparisons, precision-wise risk profiles, and footprint-aware accounting separate trigger quality and budget behavior from backend implementation choices. **This addresses (L4).**

A comprehensive review of **Related Work** is provided in Appendix A.

2 Preliminary: Operator-wise Runtime Monitoring

We use an operator-level view of transformer training to define the local runtime statistics used by our controller. This view separates the compositional structure of the network from within-layer parallelism, but we keep only the notation needed in the main text. Detailed instantiations for attention, feed-forward networks, normalization, residual paths, aggregation, and backpropagation are deferred to Appendix B.

Training objective. Let $F(\cdot; \Theta)$ be the model mapping and $r(\cdot; \theta^{(r)})$ the readout to logits. For data $(x, y) \sim \mathcal{D}$, we write the training objective as

$$L(\Theta, \theta^{(r)}) = \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\ell(r(F(x; \Theta); \theta^{(r)}), y) \right],$$

where Θ contains the parameters of the model’s operator blocks.

Operator-level decomposition. We write the network as a composition of U nested operators:

$$\begin{aligned} y^{[0]} &= x, \\ y^{[u]} &= \mathcal{N}^{[u]}(y^{[u-1]}; \Xi^{[u]}), \quad u = 1, \dots, U. \end{aligned} \quad (1)$$

Each block $\mathcal{N}^{[u]}$ may contain multiple sibling operators whose outputs are combined by an aggregation operation. This covers multi-head attention, feed-forward modules, normalization, and residual paths. We leave the explicit sibling-operator and aggregation formulas to Appendix B.

Monitored runtime units. The decomposition above induces a set of monitored units $\mathcal{B}$. A unit $b \in \mathcal{B}$ may correspond to an operator, branch, block, or layer, depending on the implementation granularity. We denote its parameters by θ_b . At training step t , let $\widehat{L}_t$ be the stochastic mini-batch loss. The local gradient and its norm are

$$g_{b,t} = \nabla_{\theta_b} \widehat{L}_t, \quad n_{b,t} = \|g_{b,t}\|_2.$$

The scalar $n_{b,t}$ is the basic runtime statistic used in Section 3. Because raw gradient norms are not directly comparable across operators with different typical scales, the next section introduces a history-normalized statistic for local stability monitoring.

3 GNMR Runtime Stability Controller

Section 2 defines the operator-wise gradient norm $n_{b,t}$ as a local statistic available during training. Although $n_{b,t}$ is local and cheap to obtain, its raw value cannot be used as a risk score directly, since different operators have different typical scales. A useful risk criterion should therefore measure whether a unit’s current update is unusual relative to its own history.

We instantiate this criterion with two history-normalized risk scores: GNMR for long-horizon deviations and Δ -GNMR for short-window changes. These scores are then used to drive the budgeted recovery policy in Section 3.3.

3.1 Long-horizon Risk Score: GNMR

GNMR, short for Gradient Norm-to-Mean Ratio, measures how unusual the current gradient norm of a monitored unit is relative to its own historical scale. For an operator parameter $\theta^{[u,k]}$, this corresponds to comparing its current gradient norm with its running historical mean:

$$\text{GNMR}_t(\theta^{[u,k]}) := \frac{\|g_t^{[u,k]}\|}{\sum_{\tau=1}^{t-1} \|g_\tau^{[u,k]}\| / (t-1)},$$

where $g_t^{[u,k]}$ denotes the evaluated gradient at the t -th step with respect to $\theta^{[u,k]}$. We set $\text{GNMR}_1(\theta^{[u,k]}) = 1$. Instead of comparing raw gradient norms across operators, GNMR compares each unit with its own past behavior, making the

Algorithm 1 Online computation of GNMR and Δ -GNMR for one monitored unit

Input: gradient $g_{b,t}$, historical mean $A_{b,t-1}$, recent GNMR queue Q_b , window length δ , step t , small constant ϵ .
 $n_{b,t} \leftarrow \|g_{b,t}\|_2$
if $t = 1$ or $A_{b,t-1} = 0$ **then**
 $\text{GNMR}_{b,t} \leftarrow 1$
else
 $\text{GNMR}_{b,t} \leftarrow n_{b,t} / (A_{b,t-1} + \epsilon)$
end if
if $|Q_b| = \delta$ **then**
 $\Delta\text{-GNMR}_{b,t} \leftarrow \text{GNMR}_{b,t} / (\text{mean}(Q_b) + \epsilon)$
else
 $\Delta\text{-GNMR}_{b,t} \leftarrow 1$
end if
 $A_{b,t} \leftarrow A_{b,t-1} + (n_{b,t} - A_{b,t-1}) / t$
Push $\text{GNMR}_{b,t}$ into Q_b and keep only the most recent δ values.
Output: $\text{GNMR}_{b,t}$, $\Delta\text{-GNMR}_{b,t}$, updated $A_{b,t}$ and Q_b .

score dimensionless and more suitable for ranking local runtime risks across heterogeneous modules.

Compared with stability indicators such as gradient spike score (GSS) (Huang et al., 2025b), gradient signal-to-noise ratio (GSNR) (Jiang et al., 2023), and Jacobian spectral norm (Takase et al., 2023), GNMR is designed for online operator-wise monitoring. GSS relies on full-process gradient statistics, GSNR requires large-batch gradients, and Jacobian-based quantities introduce substantial overhead. In contrast, GNMR only uses the current gradient norm and a running historical average, as summarized in Algorithm 1.

3.2 Short-window Risk Score: Δ -GNMR

GNMR uses long-horizon history, which is useful for measuring persistent deviation but may smooth out abrupt local changes. To capture short-window instability, we introduce Δ -GNMR, which compares the current GNMR value with its recent-window average:

$$\Delta\text{-GNMR}_t(\theta^{[u,k]}) := \frac{\text{GNMR}_t(\theta^{[u,k]})}{\sum_{\tau=1}^{\delta} \text{GNMR}_{t-\tau}(\theta^{[u,k]}) / \delta},$$

where δ is the short-window length. Thus, GNMR asks whether a unit deviates from its long-term behavior, while Δ -GNMR asks whether the current risk rises sharply relative to recent steps.

Similar to GNMR, Δ -GNMR metric can be computed online with a small queue of recent GNMR

Algorithm 2 Causal budgeted recovery from GNMR risk scores

Input: recoverable units $\mathcal{B}_{\text{rec}}$, thresholds α_t, β_t , active budget maxO , lock length T_{lock} .
State: active set $\mathcal{A}_t$ and lock counter $\text{Lock}_{b,t}$ for each $b \in \mathcal{B}_{\text{rec}}$; initialize $\mathcal{A}_1 = \emptyset$ and $\text{Lock}_{b,1} = 0$ for every b .
for each step t **do**
 Use the read-only set $\mathcal{A}_t$ for the step- t forward and backward pass.
 for each $b \in \mathcal{B}_{\text{rec}}$ **do**
 Compute $\text{GNMR}_{b,t}$ and $\Delta\text{-GNMR}_{b,t}$ from the step- t gradient.
 $r_{b,t}^G \leftarrow \text{GNMR}_{b,t} / \alpha_t$,
 $r_{b,t}^\Delta \leftarrow \Delta\text{-GNMR}_{b,t} / \beta_t$.
 $R_{b,t} \leftarrow \max\{r_{b,t}^G, r_{b,t}^\Delta\}$.
 end for
 Use the step- t gradients to update the optimizer and controller state.
 $\mathcal{P}_t \leftarrow \{b : R_{b,t} > 1\}$, $\mathcal{L}_t \leftarrow \{b : \text{Lock}_{b,t} > 0\}$, and $\mathcal{C}_{t+1} \leftarrow \mathcal{P}_t \cup \mathcal{L}_t$.
 $\mathcal{A}_{t+1} \leftarrow \text{Top}_{\leq \text{maxO}}(\mathcal{C}_{t+1}; R_t)$.
 for each $b \in \mathcal{B}_{\text{rec}}$ **do**
 if $b \in \mathcal{A}_{t+1}$ and $b \in \mathcal{P}_t$ **then**
 $\text{Lock}_{b,t+1} \leftarrow T_{\text{lock}}$.
 else if $b \in \mathcal{A}_{t+1}$ **then**
 $\text{Lock}_{b,t+1} \leftarrow \max(\text{Lock}_{b,t} - 1, 0)$.
 else
 $\text{Lock}_{b,t+1} \leftarrow 0$.
 end if
 end for
 Commit $\mathcal{A}_{t+1}$; only the next forward pass uses this set.
end for

values and does not require extra backward passes or full-process statistics. Algorithm 1 gives the concrete computation for a generic monitored unit b ; for an operator parameter $\theta^{[u,k]}$, one can take $b = (u, k)$. The scores themselves do not define a numerical format or kernel; they provide local risk estimates that are later converted into recovery actions by the controller.

3.3 From Risk Scores to Budgeted Recovery

GNMR and Δ -GNMR provide local risk estimates, but the controller must turn these scores into bounded recovery actions. We distinguish monitored units from recoverable units. A monitored unit has runtime statistics, while a recoverable unit is one for which the backend training stack pro-

vides both a low-cost path and a higher-fidelity recovery path. Units fixed by the backend recipe remain outside the action space.

At step t , for each recoverable unit b , we combine the two risk scores as

$$R_{b,t} = \max \left\{ \text{GNMR}_{b,t} / \alpha_t, \Delta\text{-GNMR}_{b,t} / \beta_t \right\}.$$

Here, α_t and β_t control the risk sensitivity of long-horizon deviations and short-window changes. After the step- t backward pass, define the positive-risk set $\mathcal{P}_t = \{b : R_{b,t} > 1\}$, lock-eligible set $\mathcal{L}_t = \{b : \text{Lock}_{b,t} > 0\}$, and candidate set $\mathcal{C}_{t+1} = \mathcal{P}_t \cup \mathcal{L}_t$. The active set for the next step is

$$\mathcal{A}_{t+1} = \text{Top}_{\leq \max\text{O}}(\mathcal{C}_{t+1}; R_t),$$

where $\text{Top}_{\leq K}(C; R)$ returns all candidates when $|C| < K$, and otherwise returns the K highest-scoring candidates with deterministic tie-breaking. Only units in $\mathcal{A}_{t+1}$ use the recovery path at step $t + 1$; hence $|\mathcal{A}_{t+1}| \leq \min\{|\mathcal{C}_{t+1}|, \max\text{O}\}$. The newly admitted set is $\mathcal{S}_t = \mathcal{A}_{t+1} \setminus \mathcal{A}_t$.

The lock mechanism stabilizes active-set membership but does not bypass this active budget. A locked unit remains eligible, and a higher-risk unit can displace it; lock assigns no reserved slot and always honors the cap. This prevents frequent switching while preserving a fixed instantaneous recovery budget. Algorithm 2 summarizes the resulting closed-loop recovery procedure.

3.4 Controller Interface, Budget, and Calibration

Controller interface and backend constraints. GNMR operates on backend-provided low-cost and higher-fidelity recovery paths for a subset of recoverable units. It assumes that the backend provides a low-cost path and a higher-fidelity recovery path for a subset of recoverable units, which can be operators, blocks, or layers depending on the implementation. Units fixed by the backend recipe may still be monitored, but they are not adjusted by the controller. When operator-level recovery may create inconsistent execution within a block, we treat the whole transformer block as the recoverable unit, so all recoverable low-cost operations inside the block share the same recovery decision. This interface lets the same risk monitor work across controlled activation-quantization stress tests, TE-backed low-precision training, and large-model post-training stress tests, while keeping backend-specific constraints explicit.

Active budget and calibration. Recovery incurs extra memory compared with the low-cost path, especially when multiple units become risky in the same step. The budget $\max\text{O}$ is a hard cap on the active recovery set: at each step, the controller executes at most $\max\text{O}$ recoverable units on the recovery path. Let c_b denote the additional saved-activation cost of recovering unit b . Then

$$\begin{aligned} \Delta M_t &= \sum_{b \in \mathcal{A}_t} c_b \\ &\leq \max\text{O} \cdot c_{\max}, \quad c_{\max} = \max_{b \in \mathcal{B}_{\text{rec}}} c_b. \end{aligned}$$

Appendix D instantiates c_b using saved-activation bitwidths and activation footprints.

The lock interval T_{lock} stabilizes active-set membership over time: recently selected units remain in the candidate pool for a short window and can be retained if they remain among the top-risk units, but the final active set is always selected under the same $\max\text{O}$ cap. Thus, the lock mechanism reduces unnecessary switching without bypassing the recovery budget.

The thresholds α_t and β_t control risk sensitivity. The former governs long-horizon GNMR deviations, while the latter governs short-window Δ -GNMR changes. In practice, a higher warm-up threshold reduces noisy early triggers when the history is short, while scheduled relaxation or warm-up calibration maintains sensitivity after running statistics stabilize. Section 4 provides a bound-level motivation connecting active recovery, perturbation reduction, and threshold calibration.

4 Theoretical Motivation

We give a bound-level motivation for budgeted recovery. Low-cost execution may perturb forward computation, saved activations, or backward signals, and such perturbations can propagate through the computation graph. We therefore do not model low-precision error as independent zero-mean gradient noise. Instead, for each recoverable unit b and path $q \in \{\text{low}, \text{rec}\}$, let $\mathcal{E}_q^b(\rho)$ denote an operator-wise perturbation penalty that upper-bounds its contribution to the one-step smoothness bound at risk level ρ . The standing assumptions, including L -smoothness, standard SGD noise, and the perturbation envelope, are given in Appendix C.

Define the recovery gap

$$\Delta_s^b = \Delta \mathcal{E}^b(\rho_s^b) = \mathcal{E}_{\text{low}}^b(\rho_s^b) - \mathcal{E}_{\text{rec}}^b(\rho_s^b).$$

We assume $\Delta_s^b \geq 0$ on the risky regimes targeted by the controller.

Theorem 4.1 (Bound-level gain of budgeted recovery). *Let $\mathcal{A}_s \subseteq \mathcal{B}_{\text{rec}}$ be the active recovery set at step s , with $|\mathcal{A}_s| \leq \text{maxO}$. Let $\bar{\mathcal{U}}_s^{\text{Low}}$ be the one-step upper bound under the always-low-cost path, and $\bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s)$ the bound when units in $\mathcal{A}_s$ use recovery. If we denote $\Delta_s^{\mathcal{U}} := \bar{\mathcal{U}}_s^{\text{Low}} - \bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s)$, then under the standing assumptions,*

$$\Delta_s^{\mathcal{U}} = \sum_{b \in \mathcal{B}_{\text{rec}}} \Delta \mathcal{E}^b(\rho_s^b) \mathbf{1}\{b \in \mathcal{A}_s\}.$$

Moreover, for the transition from step t to $t+1$, let $\mathcal{S}_t = \mathcal{A}_{t+1} \setminus \mathcal{A}_t$ denote the newly admitted set,

$$\begin{aligned} \Pr(\rho_{t+1}^b \geq \tau_b \mid b \in \mathcal{S}_t) &\geq \pi_b, \\ \Delta \mathcal{E}^b(\rho) &\geq \Delta \mathcal{E}^b(\tau_b) \mathbf{1}\{\rho \geq \tau_b\}. \end{aligned}$$

Then it holds that

$$\mathbb{E}[\Delta_{t+1}^{\mathcal{U}}] \geq \sum_{b \in \mathcal{B}_{\text{rec}}} \Pr(b \in \mathcal{S}_t) \pi_b \Delta \mathcal{E}^b(\tau_b).$$

(See the proof in Appendix C.2.)

Corollary 4.2 (Oracle target under an active budget). *If the latent gaps Δ_s^b were observable, the oracle recovery set under budget $|\mathcal{A}_s| \leq \text{maxO}$ would solve*

$$\mathcal{A}_s^* \in \arg \max_{\mathcal{A} \subseteq \mathcal{B}_{\text{rec}}, |\mathcal{A}| \leq \text{maxO}} \sum_{b \in \mathcal{A}} \Delta_s^b.$$

Thus, it selects the largest positive gaps up to the total budget.

Corollary 4.2 identifies the latent target of a budgeted controller: ranking units by their unobserved recovery gaps under the active budget. Since Δ_s^b is not observable online, GNMR and Δ -GNMR instantiate this oracle view with lightweight runtime scores for ranking risky units. Proposition 4.3 gives a deliberately limited sufficient condition: GNMR’s ordering maximizes the certified part of the conditional bound for positive-risk units, while lock-only eligibility contributes persistence and adds no separate guarantee.

Proposition 4.3 (Restricted GNMR-ranking guarantee). *Let $\mathcal{H}_t^+$ denote the information available after the step- t backward pass and the controller-state update, but before the step- $(t+1)$ forward pass; the same step- t gradients may be used for the optimizer update before or after the controller-state update. Write $\Delta \mathcal{E}_{t+1}^b = \Delta \mathcal{E}^b(\rho_{b,t+1})$. Suppose there is a common nondecreasing $\phi : \mathbb{R}_+ \rightarrow \mathbb{R}_+$, with $\phi(0) = 0$, such that $\mathbb{E}[\Delta \mathcal{E}_{t+1}^b \mid \mathcal{H}_t^+] \geq \phi(R_{b,t} -$*

1) for $b \in \mathcal{P}_t$, while lock-only units $b \in \mathcal{L}_t \setminus \mathcal{P}_t$ have nonnegative conditional expected gaps. Then

$$\mathbb{E}[\Delta_{t+1}^{\mathcal{U}} \mid \mathcal{H}_t^+] \geq \sum_{b \in \mathcal{A}_{t+1} \cap \mathcal{P}_t} \phi(R_{b,t} - 1).$$

Under the hard budget, sorting positive-risk units by $R_{b,t}$ maximizes this certified positive-risk lower envelope. Lock-only units provide persistence but add no positive guarantee to the bound. The guarantee concerns ordering and makes no equality claim between $R_{b,t}$ and the latent oracle gap. Appendix C.4 provides the proof.

5 Experiments

In this section, we evaluate GNMR in LLaMA-style and Qwen2-like activation-quantization pre-training, DeepSeek-style mixed-precision pre-training, GPT-MoE and PanGu stress tests, and 13B–35B adaptation.

5.1 Pre-training LLaMA-2 under Activation-Quantization

Experiment setup. We pre-train LLaMA-2 (Touvron et al., 2023) models on C4-en (Raffel et al., 2020), following the settings in (Zhao et al., 2024; Kong et al., 2025b). We retain BF16 weights and optimizer states and apply token-wise quantization (Jiang et al., 2022; Ramasinghe et al., 2025) only to saved activations of LLaMA attention and SwiGLU MLP projection operators. These projection operators define the recoverable set, while embeddings, normalization, and output heads remain outside the GNMR action space. GNMR+ Δ -GNMR routes each recoverable unit between the low-cost activation path and the higher-fidelity recovery path, isolating runtime recovery from changes to the optimizer, model, or data.

Experiment results. Table 1 exposes the failure mode targeted by GNMR: a static low-cost activation path can severely degrade or fail, whereas GNMR-controlled 4-bit/8-bit recovery remains close to the fixed 8-bit and BF16/16-bit references across model sizes. This shows that online recovery over the eligible projection set can preserve the higher-fidelity trajectory without changing the training recipe. Table 2 shows that the recovery signal is highly localized: threshold exceedance stays below 0.2% in the reported runs. The sub-0.2% exceedance attributes the quality gain to a narrow dynamic set of unit-level risks. GNMR converts these risks into bounded recovery actions.

Table 1: Validation perplexity ($\downarrow$) in the controlled activation-quantization stress bench for LLaMA-2 pre-training. The low-cost path quantizes saved activations of recoverable projection operators; GNMR+ Δ -GNMR selectively routes risky units to the recovery path. $\times$ means the model fails to converge. Full FP32 denotes end-to-end FP32 training; in the other rows, weights and optimizer states remain BF16 while the listed precision applies to the recoverable saved activations.

precision	α_t	β_t	60M	130M	350M	1.3B
			1.1B tokens	2.2B tokens	6.4B tokens	13.1B tokens
4-bit	N.A.	N.A.	67.22	169.21	26.56	$\times$
4-bit/8-bit	1.5	1.3	30.59	24.66	18.84	15.71
	2.0	1.5	30.66	24.86	19.07	N.A.
	3.0	2.0	30.68	25.33	18.77	N.A.
8-bit	N.A.	N.A.	30.88	25.13	18.59	15.68
16-bit	N.A.	N.A.	30.79	25.06	18.80	15.56
Full FP32	N.A.	N.A.	30.72	24.81	18.63	15.52

Table 2: Threshold-exceedance rates in the activation-quantization stress bench: proportion of eligible linear blocks exceeding GNMR and Δ -GNMR thresholds.

Model size	α_t	β_t	GNMR	Δ -GNMR
60M	2.0	1.5	0.070%	0.081%
	3.0	2.0	0.019%	0.033%
350M	2.0	1.5	0.141%	0.067%
	3.0	2.0	0.019%	0.016%

5.2 Pre-training LLaMA-2 with DeepSeek-style Mixed Precision

Experiment setup. We next evaluate GNMR on C4-en under a DeepSeek-V3-style precision hierarchy (Liu et al., 2024). Operators fixed to 16-bit or 32-bit by the recipe stay unchanged; GNMR controls only the remaining low-precision-eligible subgraph. Because operator-level recovery can create inconsistent execution inside a block, we use each transformer block as the recoverable unit. We cap the active set at $\text{maxO} = 6$, so at most six transformer blocks use the recovery path at a step. We compare dynamic 6-bit/8-bit against fixed 8-bit, and dynamic 8-bit/16-bit against fixed 16-bit.

Experiment results. Table 3 shows that GNMR stays close to the corresponding higher-fidelity references throughout the DeepSeek-style recipe. This result tests recipe-level integration: GNMR controls temporary recovery over the eligible subgraph while leaving the fixed-precision operators unchanged. Table 5 characterizes the reported 1.3B training stack. Dynamic 8-bit/16-bit follows the higher-fidelity quality trend while preserving the

low-cost execution profile under this implementation.

Threshold and budget behavior. Tables 6 and 7 study threshold scheduling and the active recovery budget. The two-stage threshold separates noisy early GNMR history from the more stable running statistics observed later in training. The maxO study shows that the active-set cap controls recovery overhead without requiring a large high-precision set.

5.3 Pre-training Qwen2-like Models with Activation Quantization

Experiment setup. We train a 125M-parameter Qwen2-like grouped-query transformer from configuration for 20,000 C4-en updates. Its 12 layers use 12 query and 4 key/value heads, Q/K/V bias, and 84 recoverable attention/MLP projections. We simulate symmetric per-token 4-bit activation quantization with a straight-through estimator, execute selected projection outputs at 8 bits in the next forward, and retain BF16 weights and optimizer states.

Experiment results. Table 4 shows that threshold-and-lock GNMR preserves the higher-fidelity validation trajectory under 4-bit activation stress, while the fixed 4-bit path encounters severe convergence obstruction. Table 8 locates this obstruction in the upper GNMR tail: the median remains comparable across precision settings, but the fixed 4-bit run develops much larger high-percentile values. Instability therefore concentrates in a limited set of unit-steps instead of shifting the full distribution.

Table 3: Validation perplexity ($\downarrow$) for LLaMA-2 models of varying sizes under DeepSeek-style mixed-precision strategies. α_t and β_t denote the GNMR and Δ -GNMR thresholds, respectively.

precision	α_t	β_t	Training tokens (B)								
			1.3	2.6	3.9	5.2	6.6	7.9	9.2	10.5	13.1
6-bit/8-bit	1.5/1.1 $\diamond$	1.5	28.52	21.17	19.07	17.88	17.06	16.43	15.93	15.57	15.20
8-bit	N.A.	N.A.	29.54	21.57	19.25	17.96	17.11	16.45	15.94	15.56	15.18
8-bit/16-bit	1.5/1.1 $\diamond$	N.A.	29.50	21.48	19.14	17.89	17.04	16.37	15.86	15.48	15.10
16-bit	N.A.	N.A.	29.10	21.35	19.08	17.84	16.98	16.33	15.82	15.44	15.06

$\diamond$ The first threshold applies during the initial 2.5% of updates and the second during the remaining updates.

Table 4: Qwen2-like validation quality under fixed and GNMR-controlled activation precision. Assigned precision reports the activation-element-weighted bitwidth for the threshold-and-lock GNMR operating point.

Setting	PPL ($\downarrow$)	Assigned activation precision	Precision policy
BF16 reference	22.580	16 bit	Fixed
Fixed 8-bit	22.707	8 bit	Fixed
GNMR 4-8 bit	22.675	4.018 bits	Dynamic
Fixed 4-bit	135.290	4 bit	Fixed

Table 5: The per-step time ($\downarrow$) and the throughput ($\uparrow$) of the LLaMA-2 1.3B training task with different precision strategies on the same device.

precision	per-step time (s)	throughput (token/s)
8-bit	1.266	103532
8-bit/16-bit	1.264	103696
16-bit	1.294	101292

Table 6: The validation perplexity ($\downarrow$) for the pre-training LLaMA-2 130M model with different two-stage GNMR threshold, α_t^{init} represents the threshold in the initial 2.5% steps and α_t^{main} represents the threshold in the other steps.

α_t^{init}	α_t^{main}	Training tokens (B)		
		1.3	2.1	2.6
1.1	1.1	26.53	23.95	23.57
1.5	1.1	26.49	23.92	23.54
1.5	1.5	26.56	23.98	23.60

Table 7: The validation perplexity ($\downarrow$) for the pre-training LLaMA-2 130M model with different maxO.

maxO	Training tokens (B)		
	1.3	2.1	2.6
2	26.56	23.97	23.59
4	26.49	23.92	23.54
6	26.50	23.94	23.56
8	26.56	24.01	23.63

Under matched recovery-slot caps, Table 9 shows that GNMR restores training at every tested K_{rec} , whereas the alternative online and static rankers re-

Table 8: Percentiles of monitored unit-step GNMR values across Qwen2-like precision settings.

Setting	p_{50}	p_{95}	p_{99}
BF16	0.728	1.212	1.369
Fixed 8-bit	0.717	1.206	1.319
Fixed 4-bit	0.765	2.510	4.713

Table 9: Exact- K_{rec} Qwen2-like validation PPL. All rankers receive the same nominal recovery-slot cap.

Ranking rule	$K_{\text{rec}} = 1$	$K_{\text{rec}} = 2$	$K_{\text{rec}} = 4$
GNMR	22.656	22.658	22.663
GSNR-style	128.561	130.306	130.524
Jacobian proxy	129.563	132.686	163.589
Static profile	131.211	125.223	113.805

main close to the failed low-precision regime. The matched caps attribute the gain to GNMR ranking under the same nominal recovery budget.

5.4 Stress Test for Fine-tuning Pre-trained Models

Experiment setup. We evaluate low-precision adaptation on LLaMA-2 13B and Qwen3.5-35B-A3B (Qwen Team, 2026). For LLaMA-2 13B, we compare BF16, fixed INT8, and GNMR+ Δ -GNMR INT8/BF16. The controller acts only on the low-precision-eligible recovery path and sets maxO = 10 across 40 decoder layers. We freeze the base model, fine-tune LoRA (Hu et al.,

Table 10: Downstream results for LLaMA-2 13B. GNMR+ Δ -GNMR uses INT8/BF16 recovery.

Metric	BF16	Fixed INT8	GNMR+ Δ -GNMR
GSM8K EM ($\uparrow$)	15.09	14.71	15.01
MMLU Acc. ($\uparrow$)	29.92	29.88	29.92
HellaSwag Acc. ($\uparrow$)	64.62	64.55	64.67
WikiText-2 PPL ($\downarrow$)	5.4503	5.4519	5.4511

Table 11: WikiText-2 validation PPL for Qwen3.5-35B-A3B over four LoRA epochs.

Epoch	Fixed 8-bit	GNMR 8–16-bit	Fixed 16-bit
1	7.6682	7.6672	7.6568
2	7.5994	7.5952	7.5802
3	7.5074	7.5021	7.4857
4	7.4148	7.4045	7.3906

2022) adapters, and evaluate GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), and WikiText-2 (Merity et al., 2016). For Qwen3.5-35B-A3B, we freeze the base model and apply LoRA to all linear layers for four WikiText-2 epochs. GNMR controls simulated Q/K/V/O activation precision between 8 and 16 bits. This setting evaluates adaptation quality and makes no packed-kernel throughput claim.

Experiment results. Table 10 shows that GNMR+ Δ -GNMR matches BF16 on MMLU, gives the highest HellaSwag accuracy, and outperforms fixed INT8 on all four metrics. Its INT8/BF16 recovery policy preserves the BF16 downstream profile with INT8 as the low-precision path. Table 11 shows that GNMR improves over fixed 8-bit and remains between fixed 8-bit and fixed 16-bit at every epoch. These results support GNMR-controlled adaptation quality in this 35B-class Mixture-of-Experts stress test.

6 Conclusion

This paper frames low-precision language model training stability as a runtime control problem and presents GNMR as a causal controller for this setting. GNMR and Δ -GNMR estimate long-horizon and short-window operator-level risks, then allocate a hard-bounded recovery budget to the highest-risk recoverable units. Experiments across LLaMA-style and Qwen-style activation-quantization stress, DeepSeek-style recipe integration, GPT-MoE and PanGu settings show that selective online recovery preserves high-fidelity quality while retaining a predominantly low-cost execution path.

7 Limitations

GNMR targets runtime stability control over backend-provided low-cost and recovery paths. Its effectiveness depends on backend-level observability, recoverable units, and compatible recovery implementations. Further engineering can reduce casting, routing, and recovery-switching overheads.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. W2441021, 12288101, 12301392).

References

- Felix Abecassis, Anjolie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, and 1 others. 2025. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*.
- Saleh Ashkboos, Bram Verhoef, Torsten Hoefer, Evangelos Eleftheriou, and Martino Dazzi. 2024. Efqt: An efficient framework for quantization-aware training. *arXiv preprint arXiv:2411.11038*.
- Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. 2024. Low-rank quantization-aware training for llms. *arXiv preprint arXiv:2406.06385*.
- Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *International conference on machine learning*, pages 1059–1071. PMLR.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Roberto L Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. 2025. Quartet: Native fp4 training can be optimal for large language models. *arXiv preprint arXiv:2505.14669*.
- Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2024. Efficientqt: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

- Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseek-moe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.
- Wenhao Dai, Ziyi Jia, Yuesi Bai, and Qingxiao Sun. 2025. Convergence-aware operator-wise mixed-precision training. *CCF Transactions on High Performance Computing*, 7(1):43–57.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332.
- Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.
- Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, and 1 others. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pages 5547–5569. PMLR.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Rizhen Hu, Yuan Cao, Boao Kong, Mou Sun, and Kun Yuan. 2026. Synergistic intra-and cross-layer regularization losses for moe expert specialization. *arXiv preprint arXiv:2602.14159*.
- Tianjin Huang, Haotian Hu, Zhenyu Zhang, Gaojie Jin, Xiang Li, Li Shen, Tianlong Chen, Lu Liu, Qingsong Wen, Zhangyang Wang, and 1 others. 2025a. Stable-spam: How to train in 4-bit more stably than 16-bit adam. *arXiv preprint arXiv:2502.17055*.
- Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. 2025b. Spam: Spike-aware adam with momentum reset for stable llm training. *arXiv preprint arXiv:2501.06842*.
- Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and 1 others. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32.
- Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.
- Jinda Jia, Cong Xie, Hanlin Lu, Daoce Wang, Hao Feng, Chengming Zhang, Baixi Sun, Haibin Lin, Zhi Zhang, Xin Liu, and 1 others. 2024. Sdp4bit: Toward 4-bit communication quantization in sharded data parallelism for llm training. *Advances in Neural Information Processing Systems*, 37:8734–8759.
- Guoqing Jiang, Jinlong Liu, Zixiang Ding, Lin Guo, and Wei Lin. 2023. Accelerating large batch training via gradient signal to noise ratio (gsnr). *arXiv preprint arXiv:2309.13681*.
- Ziyu Jiang, Xuxi Chen, Xueqin Huang, Xianzhi Du, Denny Zhou, and Zhangyang Wang. 2022. Back razor: Memory-efficient transfer learning by self-sparsified backpropagation. *Advances in neural information processing systems*, 35:29248–29261.
- Boao Kong, Xu Huang, Yuqi Xu, Yixuan Liang, Bin Wang, and Kun Yuan. 2025a. Clapping: Removing per-sample storage for pipeline parallel distributed optimization with communication compression. *arXiv preprint arXiv:2509.19029*.
- Boao Kong, Junzhu Liang, Yuxi Liu, Renjia Deng, and Kun Yuan. 2025b. Cr-net: Scaling parameter-efficient training with cross-layer low-rank structure. *arXiv preprint arXiv:2509.18993*.
- Lorenz Kummer, Kevin Sidak, Tabea Reichmann, and Wilfried Gansterer. 2023. Adaptive precision training (adapt): A dynamic quantized training approach for dnns. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 559–567. SIAM.
- Joonhyung Lee, Jeongin Bae, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. 2024. To fp8 and back again: Quantifying reduced precision effects on llm training stability. *arXiv preprint arXiv:2405.18710*.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.

- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.
- Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.
- Harshitha Menon, Michael O Lam, Daniel Osei-Kuffuor, Markus Schordan, Scott Lloyd, Kathryn Mohror, and Jeffrey Hittinger. 2018. Adapt: Algorithmic differentiation applied to floating-point precision tuning. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 614–626. IEEE.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- Paulius Micekevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Aleksii Kuchaiev, Ganesh Venkatesh, and 1 others. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.
- Paulius Micekevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, and 1 others. 2022. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.
- Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, and 1 others. 2023. Fp8-llm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*.
- Sergio P Perez, Yan Zhang, James Briggs, Charlie Blake, Josh Levy-Kramer, Paul Balanca, Carlo Luschi, Stephen Barlow, and Andrew William Fitzgibbon. 2023. Training and inference of large language models using 8-bit floating point. *arXiv preprint arXiv:2309.17224*.
- Qwen Team. 2026. [Qwen3.5: Towards native multi-modal agents](#).
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.
- Sameera Ramasinghe, Thalaiyasingam Ajanthan, Gil Avraham, Yan Zuo, and Alexander Long. 2025. Protocol models: Scaling decentralized training with communication-efficient model parallelism. *arXiv preprint arXiv:2506.01260*.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-llm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.
- Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. 2019. A tail-index analysis of stochastic gradient noise in deep neural networks. In *International Conference on Machine Learning*, pages 5827–5837. PMLR.
- Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.
- Xinyi Wan, Penghui Qi, Guangxing Huang, Min Lin, and Jialin Li. 2025. Pipeoffload: Improving scalability of pipeline parallelism with memory optimization. *arXiv preprint arXiv:2503.01328*.

- Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Connor Holmes, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. 2023. Zero++: Extremely efficient collective communication for giant model training. *arXiv preprint arXiv:2306.10209*.
- Ruizhe Wang, Yeyun Gong, Xiao Liu, Guoshuai Zhao, Ziyue Yang, Baining Guo, Zhengjun Zha, and Peng Cheng. 2025. Optimizing large language model training using fp4 quantization. *arXiv preprint arXiv:2501.17116*.
- Shibo Wang and Pankaj Kanwar. 2019. Bfloat16: The secret to high performance on cloud tpus. *URL <https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus>*.
- Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.
- Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. 2024. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313*.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.
- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR.
- Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183.
- Zhongzhi Yu, Yonggan Fu, Shang Wu, Mengquan Li, Haoran You, and Yingyan Lin. 2022. Ldp: Learnable dynamic precision for efficient deep neural network training and inference. *arXiv preprint arXiv:2203.07713*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.
- Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.
- Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

A Related works

This section reviews work on low-precision training and adaptation, with emphasis on how prior methods relate to the runtime stability-control view studied in this paper.

Mixed-precision training and numerical formats. Mixed-precision training reduces memory traffic and improves accelerator throughput by combining low-cost arithmetic with higher-fidelity accumulation or safeguards. Early FP16 training used loss scaling and selective FP32 accumulation to preserve accuracy on modern accelerators (Micikevicius et al., 2017), while BF16 improved dynamic-range robustness and became a common choice for large-scale language model training (Wang and Kanwar, 2019). More recent work has moved toward sub-16-bit training. FP8 formats and recipes, including E4M3/E5M2 variants, have shown that reduced-precision training can converge under carefully designed scaling rules (Micikevicius et al., 2022; Peng et al., 2023). Transformer Engine (TE) further makes FP8 execution practical through delayed scaling and per-tensor format selection (Perez et al., 2023). COAT compresses optimizer states and activations to FP8 (Xi et al., 2024), while recent FP4 and NVFP4 training studies push precision even lower with stronger scaling, outlier handling, and selective high-fidelity components (Wang et al., 2025; Castro et al., 2025; Abecassis et al., 2025). These works provide important backend execution paths; GNMR instead studies when an existing low-cost path should be temporarily recovered for stability.

Quantization, outliers, and memory-efficient adaptation. Quantization methods for language models highlight that low precision is often limited by outliers and fragile layers. LLM.int8 and GPT3.int8 identify outlier features as a key challenge for accurate low-bit execution (Dettmers et al., 2022), while SmoothQuant shifts activation difficulty into weights to enable W8A8 inference (Xiao et al., 2023). GPTQ, AWQ, and

ZeroQuant further improve post-training quantization through layer-wise reconstruction, salient-weight protection, or distillation (Frantar et al., 2022; Lin et al., 2024; Yao et al., 2022). For adaptation, QLoRA trains adapters over frozen 4-bit bases using NF4 quantization and paged optimizers (Dettmers et al., 2023). Quantization-aware training and efficient QAT variants further reduce the quality gap at low bit-widths (Liu et al., 2023; Bondarenko et al., 2024; Chen et al., 2024; Ashkboos et al., 2024; Zhao et al., 2023). Outlier suppression and smoothing methods also show that precision risk is often local rather than uniform (Wei et al., 2023). These methods mainly reduce quantization error through calibration, smoothing, or training procedures; GNMR uses runtime training signals to decide where recovery is needed during training.

Systems support for low-precision training.

Large-scale language model training relies on systems that combine low-precision execution with distributed parallelism. Megatron-LM scales transformer training through tensor and pipeline parallelism (Shoeybi et al., 2019), while ZeRO and ZeRO++ reduce optimizer-state and communication overhead through sharding and communication-aware design (Rajbhandari et al., 2020; Wang et al., 2023). Communication compression and quantization further reduce distributed training cost, including 4-bit communication for sharded training and protocol-level compression (Jia et al., 2024; Ramasinghe et al., 2025; Kong et al., 2025a). Pipeline and offload systems also reduce memory and communication bottlenecks in large training runs (Huang et al., 2019; Wan et al., 2025). These systems are complementary to our work: they determine how low-cost execution is implemented and scaled, while GNMR determines when local runtime evidence justifies temporary recovery.

Training stability signals and optimizer-level stabilization. Stability failures such as loss spikes and gradient spikes are a known challenge in large-scale pre-training (Takase et al., 2023). Classical gradient clipping controls exploding gradients (Pascanu et al., 2013), and Adaptive Gradient Clipping normalizes clipping by parameter scale (Brock et al., 2021). Transformer stabilization methods such as Pre-LN, RMSNorm, and DeepNorm help regulate update magnitudes in deep networks (Xiong et al., 2020). Gradient-noise-scale analysis character-

izes batch-size regimes (McCandlish et al., 2018), while GSNR-style signals use gradient statistics to guide large-batch training (Jiang et al., 2023). Recent spike-aware optimizers reset or adjust updates when instability is detected (Huang et al., 2025b,a). Reduced-precision training can further increase sensitivity to stability and hyperparameters (Lee et al., 2024). These works motivate the use of runtime stability signals, but they do not directly provide an operator-local controller for budgeted precision recovery.

Dynamic and operator-wise precision selection.

Dynamic precision methods adapt numerical precision over time, layer, or operator. ADAPT-style approaches estimate sensitivity and assign precision based on profiling or program analysis (Menon et al., 2018; Kummer et al., 2023). Learnable Dynamic Precision formulates temporal and spatial precision choices as learnable parameters to optimize the compute-accuracy trade-off (Yu et al., 2022). Convergence-aware operator-wise mixed precision methods also explore precision assignment using convergence or sensitivity information (Dai et al., 2025). These methods are closely related to our setting, but they are typically offline, coarse-grained, or aimed at compute-accuracy assignment rather than runtime stability control. GNMR differs by using online operator-wise gradient deviations to trigger temporary recovery under an explicit budget, aligning precision intervention with local training risk rather than a fixed schedule or static sensitivity profile.

B Additional details for the operator-level modeling.

This section complements the operator-wise runtime monitoring setup in Section 2. The main text keeps only the notation needed to define local gradient statistics for GNMR. Here, we spell out the sibling-operator, aggregation, and backpropagation details for common Transformer components. The goal is not to introduce a new architectural abstraction, but to make explicit how attention, feed-forward modules, normalization, residual paths, and MoE layers map to the monitored units used by the controller. The discussion applies to both encoder-only and decoder-style models (Vaswani et al., 2017; Brown et al., 2020).

Nest-parallel operator details. A depth- U Transformer stack corresponds to the composition of

nested operators $\mathcal{N}^{[1]}, \dots, \mathcal{N}^{[U]}$ acting across layers, as in Eq. (1). Inside one block u , sibling operators act on the same input:

$$z^{[u,k]} := f^{[u,k]}(y^{[u-1]}; \theta^{[u,k]}), \quad k = 1, \dots, K_u.$$

An aggregation operation then merges their outputs and may also use the skip input $y^{[u-1]}$:

$$\mathcal{N}^{[u]} = z^{[u,0]}(z^{[u,1]}, \dots, z^{[u,K_u]}; \theta^{[u,0]}, y^{[u-1]}),$$

with $\Xi^{[u]} = \left(\theta^{[u,k]} \right)_{k=0}^{K_u}$. (2)

This pattern captures multi-head attention, feed-forward modules, residual connections, normalization, and MoE variants. It also provides a common index for the monitored units whose gradient norms are used by GNMR.

Multi-head self-attention (MHA). Let H be the layer input. The head-wise projections are parallel operators

$$Q = HW_Q^{[u]}, \quad K = HW_K^{[u]}, \quad V = HW_V^{[u]}.$$

The attention aggregator then maps (Q, K, V) and the residual input H to the layer output:

$$z_{\text{att}}^{[u]}(Q, K, V; W_O^{[u]}, H) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V W_O^{[u]} + H. \quad (3)$$

Stacking multiple heads fits by letting the sibling set contain per-head projections and attention kernels, with the aggregator concatenating head outputs and applying $W_O^{[u]}$.

Feed-forward networks (FFN) and MoE. A standard two-layer FFN with residual can be represented with $K_u = 1$ as

$$z^{[u,0]}(z^{[u,1]}; \theta^{[u,0]}, y) = y + W_2 \varphi(W_1 \text{LN}(y)). \quad (4)$$

Mixture-of-experts (MoE) variants can be represented using $K_u > 1$ sibling experts $f^{[u,k]}$ and an aggregator that gates or mixes expert outputs, e.g.,

$$z^{[u,0]}(z^{[u,1]}, \dots, z^{[u,K_u]}; \theta^{[u,0]}, y) = \sum_{k=1}^{K_u} \alpha_k(y) z^{[u,k]},$$

where $\sum_{k=1}^{K_u} \alpha_k(y) = 1$. This form covers router-based and top- k mixing used in modern MoE systems (Du et al., 2022; Fedus et al., 2022; Dai et al., 2024; Hu et al., 2026). Variants with concatenation followed by projection are obtained by letting $z^{[u,0]}$ first concatenate $(z^{[u,1]}, \dots, z^{[u,K_u]})$ and then apply a linear map.

Backpropagation in the operator view. Denote the upstream gradient at layer u by $\bar{y}^{[u]} := \partial L / \partial y^{[u]}$, and define $\bar{z}^{[u,k]} := \partial L / \partial z^{[u,k]}$. The aggregator routes sensitivity to each sibling output via

$$\bar{z}^{[u,k]} = \left(\frac{\partial z^{[u,0]}}{\partial z^{[u,k]}} \right)^\top \bar{y}^{[u]}, \quad k = 1, \dots, K_u.$$

Parameter gradients follow directly from the local Jacobians:

$$\nabla_{\theta^{[u,k]}} L = \left(\frac{\partial f^{[u,k]}}{\partial \theta^{[u,k]}} \right)^\top \bar{z}^{[u,k]},$$

$$\nabla_{\theta^{[u,0]}} L = \left(\frac{\partial z^{[u,0]}}{\partial \theta^{[u,0]}} \right)^\top \bar{y}^{[u]}.$$

The gradient w.r.t. the layer input $y^{[u-1]}$ sums the sibling and aggregator contributions:

$$\bar{y}^{[u-1]} = \sum_{k=1}^{K_u} \left(\frac{\partial f^{[u,k]}}{\partial y} \right)^\top \bar{z}^{[u,k]} + \left(\frac{\partial z^{[u,0]}}{\partial y} \right)^\top \bar{y}^{[u]}.$$

These expressions are shape-consistent with Eq. (2) and specialize to standard Transformer derivatives: for attention, the aggregator derivatives backpropagate through the softmax and output projection; for FFN/MoE, they reduce to linear maps and elementwise nonlinearity Jacobians.

Implementation scope. Residual paths are naturally included by passing $y^{[u-1]}$ to the aggregator, as in Eqs. (3)–(4). Layer normalization can be modeled either as a sibling $f^{[u,k]}$ or inside the aggregator, depending on implementation needs. Parameter sharing, masking, and causal structure can be represented as operators inside $f^{[u,k]}$ or the aggregator.

In summary, the nest-parallel operator view recovers standard Transformer components while providing a common indexing scheme for operator-wise gradient statistics. This makes it a convenient interface for GNMR-based runtime stability monitoring and budgeted recovery.

C Assumptions and detailed proofs

This appendix first states the standing assumptions and proves Theorem 4.1 and Corollary 4.2. It then proves Proposition 4.3 for the deployed ranking rule and gives the supplementary fixed-threshold result retained as Theorem 4.3. Throughout, $\mathcal{B}_{\text{rec}}$ is a finite set of recoverable units, $\mathcal{F}_s$ denotes the history available before the update at step s , and the active recovery set $\mathcal{A}_s$ is $\mathcal{F}_s$ -measurable. We use the lock/update convention from Theorem 4.1: the newly admitted set at step t is exactly $\mathcal{S}_t = \mathcal{A}_{t+1} \setminus \mathcal{A}_t$.

C.1 Assumptions

Assumption C.1 (Smoothness). The training objective $\mathcal{L}(\theta)$ is L -smooth, i.e., for any θ, θ' ,

$$\mathcal{L}(\theta') \leq \mathcal{L}(\theta) + \langle \nabla \mathcal{L}(\theta), \theta' - \theta \rangle + \frac{L}{2} \|\theta' - \theta\|^2.$$

Assumption C.2 (SGD noise). The ideal mini-batch stochastic gradient at step s can be written as

$$g_s = \nabla \mathcal{L}(\theta_s) + \xi_s, \quad (5)$$

where

$$\mathbb{E}[\xi_s \mid \mathcal{F}_s] = 0, \quad \mathbb{E}[\|\xi_s\|^2 \mid \mathcal{F}_s] \leq \sigma^2. \quad (6)$$

Assumption C.3 (Perturbation envelope). For each recoverable unit $b \in \mathcal{B}_{\text{rec}}$, and for each path $q \in \{\text{low}, \text{rec}\}$, let $\mathcal{E}_q^b(\rho_s^b)$ denote an operator-wise upper-envelope penalty contributed by unit b to the one-step smoothness upper bound at risk state ρ_s^b . For an active set $\mathcal{A}_s$, let $q_b(\mathcal{A}_s) = \text{rec}$ if $b \in \mathcal{A}_s$, and let $q_b(\mathcal{A}_s) = \text{low}$ otherwise. If $r_s(\mathcal{A}_s)$ is the path-dependent perturbation in the actual update direction, then the conditional path-dependent remainder in the smoothness bound satisfies

$$\begin{aligned} & \mathbb{E} \left[-\eta \langle \nabla \mathcal{L}(\theta_s), r_s(\mathcal{A}_s) \rangle + L\eta^2 \langle g_s, r_s(\mathcal{A}_s) \rangle + \frac{L\eta^2}{2} \|r_s(\mathcal{A}_s)\|^2 \middle| \mathcal{F}_s \right] \\ & \leq \sum_{b \in \mathcal{B}_{\text{rec}}} \left[\mathbf{1}\{b \in \mathcal{A}_s\} \mathcal{E}_{\text{rec}}^b(\rho_s^b) + \mathbf{1}\{b \notin \mathcal{A}_s\} \mathcal{E}_{\text{low}}^b(\rho_s^b) \right]. \end{aligned} \quad (7)$$

We use the right-hand side of (7) to define the path-dependent part of the surrogate one-step upper bound. The recovery gap is

$$\Delta \mathcal{E}^b(\rho) = \mathcal{E}_{\text{low}}^b(\rho) - \mathcal{E}_{\text{rec}}^b(\rho). \quad (8)$$

The gap in (8) is assumed to satisfy the lower-envelope condition in Assumption C.5 on the risky regimes used in Theorem 4.1.

Remark C.4 (One sufficient parameterization). Assumption C.3 does not require low-cost errors to be independent or zero-mean. One sufficient parameterization is as follows. Let $e_{b,s}^q$ be the effective perturbation induced by the precision transformation of unit b under path q , after it propagates through the relevant forward, activation-storage, or backward computation. Define

$$\begin{aligned} \mu_q^b(\rho) &= \left\| \mathbb{E}[e_{b,s}^q \mid \mathcal{F}_s, \rho_s^b = \rho] \right\|_2, \\ v_q^b(\rho) &= \mathbb{E}[\|e_{b,s}^q\|_2^2 \mid \mathcal{F}_s, \rho_s^b = \rho]. \end{aligned} \quad (9)$$

If $G_s \geq \|\nabla \mathcal{L}(\theta_s)\|$, and $\Gamma_b \geq 1$ absorbs cross terms, forward/backward propagation, residual or normalization amplification, and operator coupling, then one may take

$$\mathcal{E}_q^b(\rho) = \eta G_s \mu_q^b(\rho) + \frac{L\eta^2}{2} \Gamma_b v_q^b(\rho). \quad (10)$$

Indeed, by (9), the linear term in the smoothness remainder is bounded by

$$\begin{aligned} & \eta \|\nabla \mathcal{L}(\theta_s)\| \\ & \times \left\| \mathbb{E}[e_{b,s}^q \mid \mathcal{F}_s, \rho_s^b = \rho] \right\|_2 \leq \eta G_s \mu_q^b(\rho). \end{aligned}$$

The quadratic term and the mixed term $L\eta^2 \langle g_s, r_s \rangle$ can be bounded by a shared path-independent term plus operator-wise second-moment penalties using Cauchy–Schwarz and Young inequalities; the constants and coupling losses are collected into Γ_b in (10). Thus the envelope can include bias, variance, activation quantization error, forward/backward perturbations, and backend-specific scaling or casting effects.

Assumption C.5 (Selected-risk persistence). Let $\mathcal{S}_t$ be the newly selected recovery set at step t . For each $b \in \mathcal{B}_{\text{rec}}$, assume that there exist $\pi_b \in (0, 1]$ and τ_b such that, whenever $\Pr(b \in \mathcal{S}_t) > 0$,

$$\Pr(\rho_{t+1}^b \geq \tau_b \mid b \in \mathcal{S}_t) \geq \pi_b. \quad (11)$$

We also assume the following lower envelope for the perturbation-penalty gap in (8):

$$\Delta \mathcal{E}^b(\rho) \geq \Delta \mathcal{E}^b(\tau_b) \mathbf{1}\{\rho \geq \tau_b\}, \quad \Delta \mathcal{E}^b(\tau_b) \geq 0. \quad (12)$$

Equivalently, the recovery path is not worse than the low-cost path in the regimes used by the lower-bound argument, and the gap is at least $\Delta \mathcal{E}^b(\tau_b)$ once the next-step risk exceeds τ_b .

This is a sufficient controller-efficacy condition rather than a universal distributional assumption; it can be estimated from training traces.

Remark C.6 (Budget convention). In Theorem 4.1, $\max O$ is interpreted as a budget on the active recovery set, i.e., $|\mathcal{A}_s| \leq \max O$. This matches the claim that $\max O$ controls instantaneous overhead. If an implementation instead caps only newly admitted units, then the same proof applies with $|\mathcal{S}_t| \leq \max O$, but the runtime overhead should be reported using the active set size $|\mathcal{A}_s|$.

C.2 Proof of Theorem 4.1

We prove the theorem through several lemmas. The proof is stated at the level of smoothness upper bounds. Thus, the quantity

$$\bar{\mathcal{U}}_s^{\text{Low}} - \bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s)$$

is a difference between two surrogate one-step upper bounds, not an exact difference between actual losses.

Lemma C.7 (Smoothness upper bound with perturbation envelope). *For any active recovery set $\mathcal{A}_s \subseteq \mathcal{B}_{\text{rec}}$, the surrogate one-step upper bound can be written as*

$$\bar{\mathcal{U}}_s(\mathcal{A}_s) = \mathcal{U}_{0,s} + \sum_{b \in \mathcal{B}_{\text{rec}}} \left[\mathbf{1}\{b \in \mathcal{A}_s\} \mathcal{E}_{\text{rec}}^b(\rho_s^b) + \mathbf{1}\{b \notin \mathcal{A}_s\} \mathcal{E}_{\text{low}}^b(\rho_s^b) \right], \quad (13)$$

where $\mathcal{U}_{0,s}$ collects the ideal SGD terms and all terms shared by the two paths. Moreover,

$$\mathbb{E}[\mathcal{L}(\theta_{s+1}) \mid \mathcal{F}_s] \leq \bar{\mathcal{U}}_s(\mathcal{A}_s). \quad (14)$$

Proof. Let $f_s := \nabla \mathcal{L}(\theta_s)$, and let $\tilde{g}_s$ denote the actual update direction under a given path assignment. By Assumption C.1, for the update $\theta_{s+1} = \theta_s - \eta \tilde{g}_s$, we have

$$\mathcal{L}(\theta_{s+1}) \leq \mathcal{L}(\theta_s) - \eta \langle f_s, \tilde{g}_s \rangle + \frac{L\eta^2}{2} \|\tilde{g}_s\|^2. \quad (15)$$

Write the actual update direction as the ideal mini-batch stochastic gradient plus a path-dependent perturbation:

$$\tilde{g}_s = g_s + r_s(\mathcal{A}_s), \quad g_s = f_s + \xi_s. \quad (16)$$

Substituting (16) into (15) gives

$$\begin{aligned} \mathcal{L}(\theta_{s+1}) &\leq \mathcal{L}(\theta_s) - \eta \langle f_s, g_s \rangle + \frac{L\eta^2}{2} \|g_s\|^2 \\ &\quad - \eta \langle f_s, r_s(\mathcal{A}_s) \rangle + L\eta^2 \langle g_s, r_s(\mathcal{A}_s) \rangle \\ &\quad + \frac{L\eta^2}{2} \|r_s(\mathcal{A}_s)\|^2. \end{aligned} \quad (17)$$

Taking conditional expectation with respect to $\mathcal{F}_s$, the ideal part in (17) is

$$\begin{aligned} &\mathbb{E} \left[\mathcal{L}(\theta_s) - \eta \langle f_s, g_s \rangle + \frac{L\eta^2}{2} \|g_s\|^2 \mid \mathcal{F}_s \right] \\ &= \mathcal{L}(\theta_s) - \eta \|f_s\|^2 + \frac{L\eta^2}{2} \mathbb{E}[\|f_s + \xi_s\|^2 \mid \mathcal{F}_s] \\ &= \mathcal{L}(\theta_s) - \eta \|f_s\|^2 + \frac{L\eta^2}{2} (\|f_s\|^2 + \mathbb{E}[\|\xi_s\|^2 \mid \mathcal{F}_s]) \\ &\leq \mathcal{L}(\theta_s) - \eta \left(1 - \frac{L\eta}{2} \right) \|f_s\|^2 + \frac{L\eta^2}{2} \sigma^2. \end{aligned} \quad (18)$$

The first equality in (18) uses (5), and the last inequality uses (6). Then, define the shared ideal term by

$$\mathcal{U}_{0,s} := \mathcal{L}(\theta_s) - \eta \left(1 - \frac{L\eta}{2} \right) \|f_s\|^2 + \frac{L\eta^2}{2} \sigma^2. \quad (19)$$

The remaining terms in (17) are exactly the path-dependent perturbation terms. By Assumption C.3, their conditional contribution is bounded by (7). Combining (19) and (7) yields (13), and hence (14). $\square$

Lemma C.8 (Active-set bound-level gain). *Let $\bar{\mathcal{U}}_s^{\text{Low}}$ be the upper bound when every recoverable unit uses the low-cost path, and let $\bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s)$ be the upper bound under active recovery set $\mathcal{A}_s$. Then it holds that:*

$$\bar{\mathcal{U}}_s^{\text{Low}} - \bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s) = \sum_{b \in \mathcal{B}_{\text{rec}}} \Delta \mathcal{E}^b(\rho_s^b) \mathbf{1}\{b \in \mathcal{A}_s\}. \quad (20)$$

Proof. By Lemma C.7, the always-low-cost upper bound is

$$\bar{\mathcal{U}}_s^{\text{Low}} = \mathcal{U}_{0,s} + \sum_{b \in \mathcal{B}_{\text{rec}}} \mathcal{E}_{\text{low}}^b(\rho_s^b). \quad (21)$$

The recovery-policy upper bound is

$$\begin{aligned} \bar{\mathcal{U}}_s^{\text{Rec}}(\mathcal{A}_s) &= \mathcal{U}_{0,s} + \sum_{b \in \mathcal{B}_{\text{rec}}} \left[\mathbf{1}\{b \in \mathcal{A}_s\} \mathcal{E}_{\text{rec}}^b(\rho_s^b) \right. \\ &\quad \left. + \mathbf{1}\{b \notin \mathcal{A}_s\} \mathcal{E}_{\text{low}}^b(\rho_s^b) \right]. \end{aligned} \quad (22)$$

Subtracting (22) from (21), the shared term $\mathcal{U}_{0,s}$ cancels. For any $b \notin \mathcal{A}_s$, both bounds contain $\mathcal{E}_{\text{low}}^b(\rho_s^b)$, so its contribution to the difference is zero. For any $b \in \mathcal{A}_s$, the difference is

$$\mathcal{E}_{\text{low}}^b(\rho_s^b) - \mathcal{E}_{\text{rec}}^b(\rho_s^b) = \Delta \mathcal{E}^b(\rho_s^b),$$

where the last equality is (8). Therefore (20) follows. $\square$

Lemma C.9 (Selected-risk persistence lower bound). *Under Assumption C.5, and assuming $\mathcal{S}_t \subseteq \mathcal{A}_{t+1}$, for each $b \in \mathcal{B}_{\text{rec}}$,*

$$\begin{aligned} \mathbb{E}[\Delta \mathcal{E}^b(\rho_{t+1}^b) \mathbf{1}\{b \in \mathcal{A}_{t+1}\}] \\ \geq \Pr(b \in \mathcal{S}_t) \pi_b \Delta \mathcal{E}^b(\tau_b). \end{aligned} \quad (23)$$

Proof. Since $\mathcal{S}_t \subseteq \mathcal{A}_{t+1}$, we have

$$\mathbf{1}\{b \in \mathcal{A}_{t+1}\} \geq \mathbf{1}\{b \in \mathcal{S}_t\}. \quad (24)$$

By (12), $\Delta \mathcal{E}^b(\rho_{t+1}^b) \geq 0$, so (24) implies

$$\begin{aligned} \mathbb{E}[\Delta \mathcal{E}^b(\rho_{t+1}^b) \mathbf{1}\{b \in \mathcal{A}_{t+1}\}] \\ \geq \mathbb{E}[\Delta \mathcal{E}^b(\rho_{t+1}^b) \mathbf{1}\{b \in \mathcal{S}_t\}]. \end{aligned} \quad (25)$$

Using (12) again,

$$\begin{aligned} \mathbb{E}[\Delta \mathcal{E}^b(\rho_{t+1}^b) \mathbf{1}\{b \in \mathcal{S}_t\}] \\ \geq \Delta \mathcal{E}^b(\tau_b) \Pr(\rho_{t+1}^b \geq \tau_b, b \in \mathcal{S}_t). \end{aligned} \quad (26)$$

If $\Pr(b \in \mathcal{S}_t) = 0$, (23) is immediate. Otherwise, the product rule and (11) give

$$\begin{aligned} \Pr(\rho_{t+1}^b \geq \tau_b, b \in \mathcal{S}_t) \\ = \Pr(b \in \mathcal{S}_t) \Pr(\rho_{t+1}^b \geq \tau_b \mid b \in \mathcal{S}_t) \\ \geq \Pr(b \in \mathcal{S}_t) \pi_b. \end{aligned} \quad (27)$$

Combining (25), (26), and (27) gives (23). $\square$

Proof of Theorem 4.1. The identity in (20) follows directly from Lemma C.8. This is an equality because it is the algebraic difference between two one-step smoothness upper bounds.

For the second statement, apply Lemma C.9 to each $b \in \mathcal{B}_{\text{rec}}$ and sum over b . Using (20) at $s = t+1$, we obtain

$$\begin{aligned} \mathbb{E}[\bar{\mathcal{U}}_{t+1}^{\text{Low}} - \bar{\mathcal{U}}_{t+1}^{\text{Rec}}(\mathcal{A}_{t+1})] \\ = \sum_{b \in \mathcal{B}_{\text{rec}}} \mathbb{E}[\Delta \mathcal{E}^b(\rho_{t+1}^b) \mathbf{1}\{b \in \mathcal{A}_{t+1}\}] \\ \geq \sum_{b \in \mathcal{B}_{\text{rec}}} \Pr(b \in \mathcal{S}_t) \pi_b \Delta \mathcal{E}^b(\tau_b). \end{aligned} \quad (28)$$

This proves (28). $\square$

Lemma C.10 (Variance-gap model as a special case). *Suppose the path-induced perturbations are zero-mean, conditionally independent, and only the second-moment term is retained in the smoothness upper bound. If $\Gamma_b = 1$, then*

$$\mathcal{E}_q^b(\rho) = \frac{L\eta^2}{2} v_q^b(\rho), \quad (29)$$

and hence

$$\Delta \mathcal{E}^b(\rho) = \frac{L\eta^2}{2} (v_{\text{low}}^b(\rho) - v_{\text{rec}}^b(\rho)). \quad (30)$$

Proof. Under the stated zero-mean assumption, the bias strength satisfies $\mu_q^b(\rho) = 0$. If only the second-moment contribution is retained and $\Gamma_b = 1$, the perturbation envelope in Remark C.4 reduces to (29). Subtracting the recovery-path penalty from the low-cost-path penalty and using (8) gives (30). Thus, the previous variance-gap analysis is recovered as a special case of the perturbation-envelope model. $\square$

C.3 Proof of Corollary 4.2

Proof. By Theorem 4.1, the bound-level gain for active set $\mathcal{A}_s$ is $\sum_{b \in \mathcal{A}_s} \Delta_s^b$. Therefore, under budget $|\mathcal{A}_s| \leq K$, the oracle set solves

$$\max_{\mathcal{A} \subseteq \mathcal{B}_{\text{rec}}} \sum_{b \in \mathcal{A}} \Delta_s^b \quad \text{s.t.} \quad |\mathcal{A}| \leq K. \quad (31)$$

The objective in (31) is modular. If an optimal set contains i but excludes j with $\Delta_s^j > \Delta_s^i$, replacing i with j strictly increases the summation $\sum_{b \in \mathcal{A}_s} \Delta_s^b$, contradicting optimality. Thus, the oracle selects the largest positive gaps under the budget. When all selected gaps are nonnegative and the budget is filled, this is the top- K set by Δ_s^b . $\square$

C.4 Proof of Proposition 4.3

Use $\mathcal{H}_t^+$ from Proposition 4.3 and the definitions of $\mathcal{P}_t$, $\mathcal{L}_t$, $\mathcal{C}_{t+1}$, and $\mathcal{A}_{t+1}$ from Section 3.3. The controller commits $\mathcal{A}_{t+1}$ before the next forward pass, so $\mathcal{A}_{t+1}$ is $\mathcal{H}_t^+$ -measurable. The additive recovery identity in Theorem 4.1 gives

$$\Delta_{t+1}^{\mathcal{U}} = \sum_{b \in \mathcal{A}_{t+1}} \Delta \mathcal{E}_{t+1}^b.$$

Conditional linearity and the two assumptions in Proposition 4.3 imply

$$\begin{aligned} \mathbb{E}[\Delta_{t+1}^{\mathcal{U}} \mid \mathcal{H}_t^+] &= \sum_{b \in \mathcal{A}_{t+1} \cap \mathcal{P}_t} \mathbb{E}[\Delta \mathcal{E}_{t+1}^b \mid \mathcal{H}_t^+] \\ &\quad + \sum_{b \in \mathcal{A}_{t+1} \cap (\mathcal{L}_t \setminus \mathcal{P}_t)} \mathbb{E}[\Delta \mathcal{E}_{t+1}^b \mid \mathcal{H}_t^+] \\ &\geq \sum_{b \in \mathcal{A}_{t+1} \cap \mathcal{P}_t} \phi(R_{b,t} - 1). \end{aligned}$$

The right-hand side is modular. Consider a feasible set that contains $i \in \mathcal{P}_t$ and excludes $j \in \mathcal{P}_t$ with $R_{j,t} > R_{i,t}$. Exchanging i for j cannot decrease the objective because ϕ is nondecreasing. Repeating this exchange yields the highest-risk positive units. If $|\mathcal{P}_t| \geq \text{maxO}$, these units fill the budget. If $|\mathcal{P}_t| < \text{maxO}$, the controller admits every positive-risk unit and may use lock-only units for

the remaining slots. Each positive-risk score exceeds one, whereas each lock-only score is at most one, so the controller orders positive-risk units first. Lock-only units add persistence without increasing the certified lower envelope. This proves Proposition 4.3 without identifying GNMR scores with latent oracle gaps.

C.5 Fixed-Threshold GNMR Concentration and Proof

We state the fixed-threshold concentration result and its assumptions here because the result provides a diagnostic interpretation of GNMR thresholds. Proposition 4.3 gives the separate ranking guarantee for the deployed controller.

Assumption C.11 (A sufficient sub-exponential model for historical-mean concentration). For a monitored unit b and an analysis time $t \geq 2$, let $X_\tau^b = \|g_{b,\tau}\|_2$. Assume that the analyzed window $\tau \in \{1, \dots, t\}$ has a common positive mean $\mathbb{E}[X_\tau^b] = \mu_b > 0$. Define $Y_\tau^b = X_\tau^b / \mu_b$. The centered variables $Z_\tau^b := Y_\tau^b - 1$ have zero mean and remain independent across the historical window. They satisfy the uniform bound

$$\sup_{1 \leq \tau \leq t} \|Y_\tau^b - 1\|_{\psi_1} \leq \kappa_b$$

for a time-independent $\kappa_b > 0$. Hence, an absolute constant $c > 0$ gives, for every $x > 0$,

$$\Pr(Y_t^b - 1 \geq x) \leq \exp\left(-c \min\left\{\frac{x^2}{\kappa_b^2}, \frac{x}{\kappa_b}\right\}\right). \quad (32)$$

Theorem 4.3 (Historical-mean concentration under fixed GNMR thresholds). *For unit b and $t \geq 2$, let $n_{b,t} = \|g_{b,t}\|_2$. For any fixed $\alpha = 1 + \varepsilon > 1$, Assumption C.11 implies that an absolute constant $c > 0$ satisfies*

$$\Pr(\text{GNMR}_{b,t} \geq \alpha) \leq \exp\left(-c \min\left\{\frac{\varepsilon^2}{\kappa_b^2}, \frac{\varepsilon}{\kappa_b}\right\}\right) + 2 \exp\left(-c(t-1) \min\left\{\frac{(\varepsilon/\alpha)^2}{\kappa_b^2}, \frac{\varepsilon/\alpha}{\kappa_b}\right\}\right).$$

Interpretation. Theorem 4.3 applies only to fixed high thresholds $\alpha > 1$; deployed thresholds are calibrated empirically. The first term controls current-step deviation, whereas the second controls uncertainty in the historical mean and decreases with history length. Lower deployment thresholds govern candidate qualification, while maxO independently caps high-precision execution. Sections E.5 and E.6 report both quantities for GPT-MoE and PanGu model.

Remark C.12 (Scope of the sub-exponential assumption). Assumption C.11 supplies a local sufficient condition for the Bernstein step in Lemma C.13. It is not a global claim that operator-level gradient norms remain independent, light-tailed, or stationary throughout training. Deep-network gradients can be heavy-tailed or nonstationary (Simsekli et al., 2019). Sub-exponential tails give a standard Bernstein bound (Vershynin, 2018); a clipped, martingale, mixing, robust, or empirical concentration bound can replace that step while retaining the ratio decomposition below.

Proof. Fix a monitored unit b . By Assumption C.11, the analyzed window has common mean $\mu_b > 0$. Define

$$X_t^b := n_{b,t} = \|g_{b,t}\|_2, \\ \mathbb{E}[X_\tau^b] = \mu_b \quad (1 \leq \tau \leq t),$$

and normalize it as

$$Y_t^b := \frac{X_t^b}{\mu_b}, \quad \mathbb{E}[Y_t^b] = 1.$$

The normalized historical mean is

$$S_{t-1}^b = \frac{1}{t-1} \sum_{\tau=1}^{t-1} Y_\tau^b = \frac{\bar{n}_{b,t-1}}{\mu_b}. \quad (33)$$

Therefore,

$$\text{GNMR}_{b,t} = \frac{X_t^b}{\bar{n}_{b,t-1}} = \frac{Y_t^b}{S_{t-1}^b}. \quad (34)$$

Lemma C.13 (Concentration of the historical mean). *Under Assumption C.11, every $\nu \in (0, 1)$ and $t \geq 2$ satisfy*

$$\Pr(S_{t-1}^b \leq 1 - \nu) \\ \leq 2 \exp\left(-c(t-1) \min\left\{\frac{\nu^2}{\kappa_b^2}, \frac{\nu}{\kappa_b}\right\}\right). \quad (35)$$

Proof. Let $Z_\tau = Y_\tau^b - 1$. Assumption C.11 gives $\mathbb{E}[Z_\tau] = 0$, $\|Z_\tau\|_{\psi_1} \leq \kappa_b$, and independence across τ . Equation (33) gives

$$S_{t-1}^b - 1 = \frac{1}{t-1} \sum_{\tau=1}^{t-1} Z_\tau.$$

Bernstein's inequality for independent sub-exponential variables yields

$$\Pr(|S_{t-1}^b - 1| \geq \nu) \\ \leq 2 \exp\left(-c(t-1) \min\left\{\frac{\nu^2}{\kappa_b^2}, \frac{\nu}{\kappa_b}\right\}\right).$$

Keeping its lower-tail event proves Equation (35). $\square$

Lemma C.14 (Ratio event decomposition). *Let $\alpha = 1 + \varepsilon > 1$, $s_1 = \varepsilon/2$, and $s_2 = \varepsilon/(2\alpha)$. Then*

$$\{\text{GNMR}_{b,t} \geq \alpha\} \subseteq \{Y_t^b \geq 1 + s_1\} \cup \{S_{t-1}^b \leq 1 - s_2\}. \quad (36)$$

Proof. We prove the contrapositive. Assume $Y_t^b < 1 + s_1$ and $S_{t-1}^b > 1 - s_2$. Equation (34) gives

$$\text{GNMR}_{b,t} < \frac{1 + s_1}{1 - s_2}.$$

The chosen values satisfy

$$\alpha(1 - s_2) = \alpha - \frac{\varepsilon}{2} = 1 + \frac{\varepsilon}{2} = 1 + s_1.$$

Hence $(1 + s_1)/(1 - s_2) = \alpha$, which gives $\text{GNMR}_{b,t} < \alpha$. This proves Equation (36). $\square$

Proof of Theorem 4.3. Let $\alpha = 1 + \varepsilon > 1$, $s_1 = \varepsilon/2$, and $s_2 = \varepsilon/(2\alpha)$. Lemma C.14 and the union bound give

$$\Pr(\text{GNMR}_{b,t} \geq \alpha) \leq \Pr(Y_t^b \geq 1 + s_1) + \Pr(S_{t-1}^b \leq 1 - s_2). \quad (37)$$

For the first term, Equation (32) gives

$$\Pr(Y_t^b \geq 1 + s_1) \leq \exp \left[-c \min \left\{ \frac{s_1^2}{\kappa_b^2}, \frac{s_1}{\kappa_b} \right\} \right].$$

Because $s_1 = \varepsilon/2$, absorbing numerical constants into c yields

$$\Pr(Y_t^b \geq 1 + s_1) \leq \exp \left(-c \min \left\{ \frac{\varepsilon^2}{\kappa_b^2}, \frac{\varepsilon}{\kappa_b} \right\} \right). \quad (38)$$

Lemma C.13 bounds the second term as

$$\begin{aligned} & \Pr(S_{t-1}^b \leq 1 - s_2) \\ & \leq 2 \exp \left(-c(t-1) \min \left\{ \frac{s_2^2}{\kappa_b^2}, \frac{s_2}{\kappa_b} \right\} \right). \end{aligned}$$

Substituting $s_2 = \varepsilon/(2\alpha)$ and absorbing numerical constants into c gives

$$\begin{aligned} & \Pr(S_{t-1}^b \leq 1 - s_2) \\ & \leq 2 \exp \left(-c(t-1) \min \left\{ \frac{(\varepsilon/\alpha)^2}{\kappa_b^2}, \frac{\varepsilon/\alpha}{\kappa_b} \right\} \right). \end{aligned} \quad (39)$$

Combining Equations (37), (38), and (39) proves Theorem 4.3. $\square$

D Peak-memory modeling under the maxO budget

This section provides a simple upper-bound model for the additional peak *activation* memory introduced when the controller promotes a subset of operators to higher precision under a hard maxO budget.

Setup. Let $\mathcal{B}$ be the set of eligible operators. For each operator $b \in \mathcal{B}$, let A_b denote the number of activation elements that must be saved for backward under a fixed batch and sequence configuration. At step t , let $\mathcal{A}_t \subseteq \mathcal{B}$ be the active set of operators executed in high precision. By design, the budget enforces $|\mathcal{A}_t| \leq \text{maxO}$ for all t .

From bitwidth to bytes. Let p_{low} and p_{high} denote the storage precisions in bits per element. Memory is typically measured in bytes, and one byte equals eight bits. Therefore, the bytes-per-element conversion is

$$\text{bytes per element} = \frac{\text{bits per element}}{8}.$$

Additional peak activation memory. Relative to a baseline that runs all eligible operators in low precision, the additional activation memory at step t is

$$\Delta M_t = \frac{p_{\text{high}} - p_{\text{low}}}{8} \sum_{b \in \mathcal{A}_t} A_b,$$

measured in bytes. Using $|\mathcal{A}_t| \leq \text{maxO}$, we obtain the upper bound

$$\begin{aligned} \Delta M_t & \leq \frac{p_{\text{high}} - p_{\text{low}}}{8} \sum_{b \in \text{TopK}(\mathcal{B}, \text{maxO}; A_b)} A_b \\ & \leq \frac{p_{\text{high}} - p_{\text{low}}}{8} \cdot \text{maxO} \cdot A_{\text{max}}, \end{aligned} \quad (40)$$

where $A_{\text{max}} := \max_{b \in \mathcal{B}} A_b$.

Interpretation. Eq. (40) shows that the additional peak activation memory grows linearly with the budget maxO and with the activation footprint of the promoted operators. This explains why a hard maxO constraint is an effective and model-agnostic knob for controlling peak memory when dynamic precision promotion is triggered by rare but bursty instability events.

E Experimental Setup and Additional Results

This appendix records the training configurations and controller-level analyses for Section 5. We

use public research datasets and model checkpoints under their licenses or access terms. Our release excludes the original datasets and checkpoints.

E.1 Pre-training LLaMA-2 with Activation Quantization

Experiment setup. We implement the framework of Zhao et al. (2024) across all LLaMA scales. Each run uses sequence length 256 and global batch size 512, or about 131K tokens per optimization step. We linearly warm up the learning rate for the first 10% of updates, then apply cosine decay to 10% of its peak value. Table 12 lists each architecture and training budget.

Static sensitivity-profile comparison. We compare online control with a tuned static sensitivity-profile proxy on the 130M and 350M activation-quantization stress tests. All methods share the data, optimizer, learning-rate schedule, token budget, and BF16 weights and optimizer states from the main activation-quantization setting. The actuator quantizes the activations of all Transformer-block linear operators to simulated 4 or 8 bits.

The static proxy runs 500 BF16 warm-up steps while monitoring the GNMR/gradient-stability statistics. It averages the per-block scores over the warm-up, fixes the top 25% of blocks to 8-bit activations, and executes the remaining blocks at 4 bits for the rest of training. This controlled proxy fixes one sensitivity profile after warm-up; ADAPT can reassign precision stage-wise (Menon et al., 2018).

Experiment results. Table 13 separates online control from a fixed sensitivity profile. The static profile remains in the failed low-precision regime, while GNMR+ Δ -GNMR updates the recovery set as risk moves during training.

Matched-trigger and overhead characterization. We further isolate trigger quality from the recovery actuator on the 130M benchmark. All rankers use the same LLaMA-style model, C4-en data (Raffel et al., 2020), optimizer, 2.6B-token budget, 4-bit low-cost path, 8-bit recovery path, and $\max O = 4$. GNMR applies thresholded online selection with $T_{\text{lock}} = 1$, so the active cap remains an upper bound. The GSNR-style (Jiang et al., 2023) and Jacobian (Takase et al., 2023) proxies fill four recovery slots after each backward pass. Step time and peak memory characterize the reported software stack.

Table 14 shows that GNMR obtains the best final PPL with selective promotion. It also records the

lowest step time, and its peak memory falls below GSNR. The Jacobian proxy uses comparable memory but gives worse PPL while filling the recovery budget. This comparison attributes the result to selective trigger quality under a shared actuator.

E.2 Pre-training LLaMA-2 with DeepSeek-style Recipe-Level Recovery

Experiment setup. This setting layers GNMR on an existing DeepSeek-style precision recipe. We retain the sequence length, batch size, and learning-rate scheduler from the activation-quantization experiments. The recipe fixes selected operators at 16 or 32 bits; GNMR controls only the low-precision-eligible subgraph. We use a two-stage GNMR threshold, with a higher threshold during the first 2.5% of updates and a lower threshold thereafter.

We instantiate two low-cost/recovery hierarchies. The first uses Transformer Engine hybrid FP8 (Perez et al., 2023) as the low-cost path and BF16 as recovery. Transformer Engine applies E4M3 in the forward pass, E5M2 in the backward pass, and BF16 parameter updates. The second uses simulated E2M1 or E3M2 operators as the low-cost path and hybrid FP8 as recovery. It quantizes token activations, weights, incoming gradients, and computed gradients to emulate the target format.

GNMR acts on the units exposed by each backend. The activation-quantization stress test exposes saved attention and SwiGLU projection activations. Embeddings, normalization layers, output heads, and operators fixed by the recipe remain outside the controller action space.

Experiment results. Figures 2 and 3 show that GNMR-controlled recovery suppresses high-risk excursions relative to the fixed low-cost path. Figure 4 preserves the expected model-and-token scaling trend from 130M to 3B.

E.3 Controlled Qwen2-like Pre-training

Experiment setup. We instantiate the 125M-parameter Qwen2-like model from configuration without an official checkpoint. Its 12 decoder layers use hidden size 768, FFN size 2048, 12 query heads, 4 key/value heads, grouped-query attention with 256-wide K/V projections, Q/K/V bias, rotary-position-embedding base 10^6 , and RMSNorm epsilon 10^{-6} . All runs use the C4-en dataset, T5-base tokenizer, seed 1, sequence length 256, global/micro batch sizes 512/64, and 20,000 updates. AdamW uses peak learning rate 2.5×10^{-3} ,

Table 12: LLaMA configurations and nominal end-to-end training budgets for GNMR-guided activation quantization. The main validation table labels the evaluation checkpoints at which PPL is reported.

Params	Hidden	Intermediate	Heads	Layers	Training Tokens	Learning Rate
60M	512	1376	8	8	1.3B	2.5E-3
130M	768	2048	12	12	2.6B	2.5E-3
350M	1024	2736	16	24	7.8B	1E-3
1.3B	2048	5461	24	32	13.1B	6E-4

Table 13: Validation PPL ($\downarrow$) for LLaMA activation quantization under a static sensitivity profile and online GNMR control.

Method	α_t	β_t	130M	350M
			2.2B tokens	6.4B tokens
Static sensitivity profile	N.A.	N.A.	215.10	88.97
GNMR + Δ -GNMR	1.5	1.3	24.66	18.84
	2.0	1.5	24.86	19.07
	3.0	2.0	25.33	18.77

Table 14: Matched-trigger and overhead comparison on the 130M activation-quantization stress test.

Trigger	Final PPL	Promotion Ratio	Hit-maxO	Step Time (s)	Peak Mem. (MB)
GNMR	23.685	0.0040	0.0043	0.9972	3732
GSRN	23.729	—	—	1.2775	4009
Jacobian proxy	23.750	—	—	1.0167	3682

Table 15: LLaMA configurations and final validation PPL for GNMR-guided DeepSeek-style training.

Params	Hidden	Intermediate	Heads	Layers	Training Tokens	Learning Rate	Validation PPL
130M	768	2048	12	12	2.6B	1E-3	24.62
350M	1024	2736	16	24	7.8B	5E-4	18.87
1.3B	2048	5472	24	32	13.1B	5E-4	15.20
3B	2560	6848	32	32	28.8B	3E-4	13.93

$(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, gradient clipping 1.0, 10% linear warm-up, and cosine decay to 10% of the peak. The recoverable scope contains 84 Q/K/V/O and gate/up/down projection units.

Threshold-and-lock GNMR denotes the operating point without an exact fixed K_{rec} . We set $\alpha = 1.5$, $\beta = 1.3$, a five-step Δ -GNMR window, $\text{maxO} = 4$, and $T_{\text{lock}} = 1$; threshold-qualified and lock-eligible units form the candidate pool, while maxO caps the active recovery set. The exact- K_{rec} online rankers use $K_{\text{rec}} \in \{1, 2, 4\}$,

fill the selected budget after each backward pass, and set $T_{\text{lock}} = 0$. The GSNR-style score divides the absolute gradient-norm deviation from its running mean by its running standard deviation. The Jacobian proxy uses the update-to-weight ratio, $\|g_b\|_2 / \|\theta_b\|_2$, normalized by its running mean. The tuned static sensitivity-profile proxy monitors GNMR/ Δ -GNMR for 500 low-cost warm-up steps, then permanently recovers the top K_{rec} units. Full ADAPT includes additional stage-wise reassignment (Menon et al., 2018); this comparison isolates a fixed sensitivity profile.

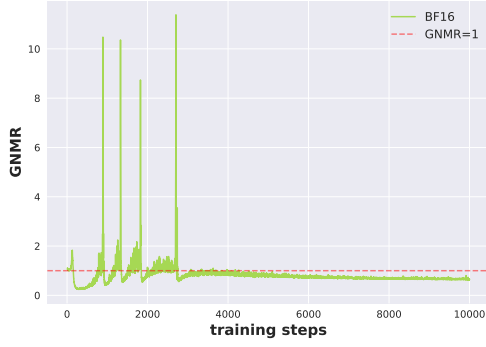


Figure 2: GNMR during LLaMA-2 BF16 pre-training.



Figure 3: Average GNMR across precision strategies.

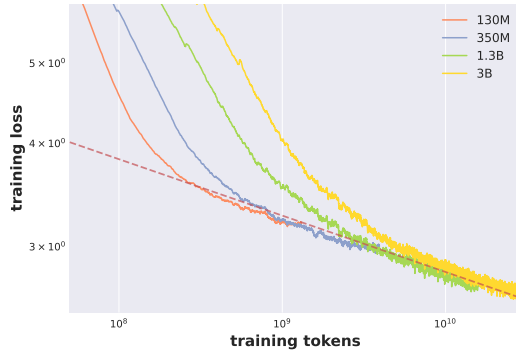


Figure 4: GNMR-controlled LLaMA scaling under the 6/8-bit setting. Table 15 lists the configurations and final PPL.

The low-cost path applies simulated symmetric per-token uniform 4-bit quantization with a straight-through estimator to each recoverable projection output. The next forward pass quantizes selected outputs to 8 bits. We keep weights and optimizer states in BF16. The simulator returns quantized tensors in BF16 storage, so these experiments support quality and theoretical footprint claims. Packed-memory and low-bit-kernel speedups remain outside their scope.

For unit b , let N_b denote the number of output activation elements and q_b its assigned bitwidth. We define the activation-element-weighted assigned bitwidth as $\bar{b}_{\text{act}} = \sum_b N_b q_b / \sum_b N_b$. The count-weighted unit-promotion ratio uses a different denominator; applying $4 + 4 \times 0.00894$ to the 4.018-bit value would mix the two denominators. The activation-byte ratio weights selected outputs by N_b ; the recovery-MAC proxy weights them by projection multiply-accumulate count.

Experiment results. Table 16 shows that GNMR restores the higher-fidelity training trajectory at ev-

ery tested recovery budget. The alternative online scores and fixed sensitivity profile remain in the obstructed low-precision regime.

Threshold-and-lock operating point. Threshold-and-lock GNMR reaches 22.675 PPL with an activation-element-weighted assigned bitwidth of 4.018. Only 0.894% of the $84 \times 20,000$ forward unit-steps use the recovery path. Profiling records emitted every 10 updates give 0.920% sampled active occupancy, and 5.26% of controller decisions reach maxO. This executed recovery corresponds to a 0.613% theoretical activation-byte recovery ratio and a 0.537% theoretical recovery-MAC proxy. The different denominators prevent deriving the 4.018-bit average from the unit promotion ratio or equating whole-run promotion with sampled occupancy.

Ranking sanity check. A separate random-selection run at $K_{\text{rec}} = 2$ obtains 138.140 PPL, compared with 22.658 for GNMR at the same slot count. We exclude the random run from the Pareto table because its log lacks a cost profile. The run tests whether arbitrary recovery suffices; its failure supports the role of GNMR ranking.

Cross-architecture recovery profiles. The Qwen2-like and LLaMA-2-like 130M runs qualify 1.367% and 7.191% of unit decisions, respectively; their sampled active frequencies are 0.920% and 4.553%. Q and K have the highest within-type selection rates in both architectures: 3.167% and 2.000% for Qwen2-like, and 11.875% and 11.542% for LLaMA-2-like. The controller selects attention projections more often than FFN projections overall. The Qwen2-like layer profile peaks at L1, with L11 second, whereas LLaMA-2-like concentrates in L8–L10. This profile supports an association between architecture and online risk location without attributing instability causally to projection iden-

Table 16: Exact- K_{rec} Qwen2-like ranking comparison with theoretical footprint accounting. The nominal K_{rec} matches the post-initialization slot cap; byte and MAC columns use the executed whole-run footprints, including the static proxy’s 500-step low-cost warm-up.

Ranking rule	K_{rec}	PPL ($\downarrow$)	$\bar{b}_{\text{act}}$	Activation-byte recovery	Recovery-MAC proxy
GNMR	1	22.656	4.036	0.797%	0.857%
GSRN-style	1	128.561	4.019	0.721%	0.826%
Jacobian proxy	1	129.563	4.030	0.729%	0.642%
Static sensitivity profile	1	131.211	4.012	0.301%	0.254%
GNMR	2	22.658	4.075	1.615%	1.627%
GSRN-style	2	130.306	4.057	1.485%	1.788%
Jacobian proxy	2	132.686	4.054	1.354%	1.210%
Static sensitivity profile	2	125.223	4.025	0.602%	0.508%
GNMR	4	22.663	4.176	3.327%	3.400%
GSRN-style	4	130.524	4.091	3.521%	3.923%
Jacobian proxy	4	163.589	4.106	3.041%	2.832%
Static sensitivity profile	4	113.805	4.049	1.204%	1.016%

Table 17: Validation PPL across lock lengths in the independent Qwen 8–16-bit setting.

T_{lock}	1	3	10
Validation PPL ($\downarrow$)	22.6753	22.6721	22.6800

Table 18: LLaMA-2 13B fine-tuning stress and downstream evaluation. Higher is better for GSM8K, MMLU, and HellaSwag; lower is better for WikiText-2 PPL.

Method	GSM8K EM	MMLU Acc.	HellaSwag Acc.	WikiText-2 PPL
BF16	15.09	29.92	64.62	5.4503
Fixed INT8	14.71	29.88	64.55	5.4519
GNMR+ Δ -GNMR INT8/BF16	15.01	29.92	64.67	5.4511

Table 19: Four-epoch WikiText-2 validation PPL for Qwen3.5-35B-A3B.

Epoch	Fixed 8-bit	GNMR 8–16-bit	Fixed 16-bit
1	7.6682	7.6672	7.6568
2	7.5994	7.5952	7.5802
3	7.5074	7.5021	7.4857
4	7.4148	7.4045	7.3906

tity.

Lock sensitivity. The lock study uses an independent 8–16-bit Qwen setting, separate from the main 4–8-bit pre-training run. Lock retains short-term candidate eligibility, and maxO still caps every active set.

All three lock lengths produce good validation quality, with $T_{\text{lock}} = 3$ numerically best in this run. Extending persistence to 10 introduces no visible quality penalty.

E.4 LLaMA-2 13B and Qwen3.5-35B-A3B Adaptation Stress Tests

Experiment setup. We evaluate BF16, fixed INT8, and GNMR+ Δ -GNMR INT8/BF16 on a frozen LLaMA-2 13B base model. Each run fine-tunes LoRA adapters on two NVIDIA A100 80GB GPUs. AdamW uses learning rate 2×10^{-5} , zero weight decay, gradient clipping 1.0, LoRA rank 8, LoRA scaling $\alpha_{\text{LoRA}} = 16$, and dropout 0.05. GNMR

Table 20: GPT-based MoE configuration. GNMR controls the low-precision-eligible MoE components exposed by the backend.

Params	Attn Hidden	Per Expert Hidden	MoE Heads	Attn Heads	Topk	Layers	Learning Rate
3.7B	7168	2048	64	128	8	3	5E-5

Table 21: Training loss ($\downarrow$) for the GPT-based MoE stress test. $\times$ marks the run that fails to converge.

Precision	Training steps (K)											
	1	2	3	4	6	8	10	12	14	17	20	29
8-bit	0.223	0.205	0.189	0.312	0.176	0.170	0.167	0.163	0.160	0.159	0.158	$\times$
8-bit/16-bit	0.224	0.201	0.189	0.183	0.173	0.170	0.168	0.164	0.162	0.159	0.159	0.154
BF16	0.226	0.205	0.187	0.181	0.172	0.171	0.167	0.163	0.159	0.159	0.156	0.152

Table 22: PanGu-1B-class configuration for GNMR-guided 4-bit training.

Params	Hidden	Intermediate	Heads	Layers	KV groups	Channels	Learning Rate
1.2B	1536	6144	12	26	6	128	2E-3

Table 23: Training loss ($\downarrow$) for the PanGu-1B-class 4-bit stress test.

Precision	Training Tokens (B)											
	0	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0	
4-bit	12.261	4.656	4.177	3.670	3.527	3.348	3.314	3.238	3.170	3.156	3.124	
4-bit+GNMR	12.261	4.650	3.929	3.575	3.564	3.309	3.290	3.215	3.155	3.128	3.076	
BF16	12.261	4.608	3.878	3.526	3.472	3.210	3.160	3.077	3.004	2.978	2.913	

uses $\max O = 10$, $T_{\text{lock}} = 1$, a two-stage threshold $\alpha_t = 1.5 \rightarrow 1.1$, and raw Δ -GNMR threshold $\beta_t = 1.5$ with window size 5. GSM8K uses answer-extracted generation, MMLU uses multiple-choice conditional log-likelihood, HellaSwag uses continuation log-likelihood, and WikiText-2 uses autoregressive perplexity.

We additionally run four WikiText-2 epochs of all-linear LoRA adaptation on Qwen3.5-35B-A3B. We freeze the base model and let GNMR control simulated Q/K/V/O activation precision between 8 and 16 bits. This experiment tests large-model adaptation quality and makes no packed-kernel throughput claim.

Experiment results. All three curves improve monotonically. GNMR stays between fixed 8 and fixed 16 bits at every epoch and finishes 0.0139 PPL above fixed 16-bit. Its sampled unit-average assigned precision is 8.463 bits, and its sampled selected-unit frequency is 5.788%. These unit-

sampled statistics use a different denominator from activation-element-weighted byte fractions.

E.5 GPT-based MoE Stress Test

Experiment setup. We pre-train a 3.7B-parameter GPT-based Mixture-of-Experts (MoE) model (Radford et al., 2019; Brown et al., 2020; Shazeer et al., 2017; Jacobs et al., 1991). The run uses global batch size 32, sequence length 4096, learning rate 5×10^{-5} , and cosine decay. GNMR routes low-precision-eligible MoE components between 8-bit execution and 16-bit recovery. We use $\alpha_{\text{GNMR}} = 0.9$ after a 5% high-fidelity warm-up, which stabilizes the running history before GNMR starts making recovery decisions.

Experiment results. Table 21 shows that fixed 8-bit execution becomes unstable and fails to converge, while GNMR-controlled 8-bit/16-bit recovery tracks the BF16 reference. At the deployed threshold $\alpha = 0.9$, 7.06% of recoverable expert

components qualify; 3.12% exceed the diagnostic threshold 1.5. The model exposes $3 \times 64 = 192$ expert components, so $\text{maxO} = 8$ caps instantaneous high-precision execution at $8/192 = 4.17\%$. The threshold determines candidate qualification, while maxO bounds actual high-precision work.

E.6 PanGu-1B-class 4-bit Stress Test

Experiment setup. We train a PanGu-1B-class model under fixed 4-bit execution, GNMR-controlled recovery, and BF16. The run uses global batch size 256, sequence length 4096, learning rate 2×10^{-3} , cosine decay, GNMR threshold $\alpha = 0.4$, and a 4B-token budget.

Experiment results. Table 23 shows that GNMR-controlled recovery improves convergence over the fixed 4-bit path, while BF16 remains the higher-fidelity reference. At $\alpha = 0.4$, 53.28% of recoverable units qualify. The model exposes six projections in each of 26 layers, so $\text{maxO} = 8$ caps instantaneous high-precision execution at $8/156 = 5.13\%$. GNMR can therefore use a permissive qualification threshold while the hard budget limits executed recovery.