WHERE LOW PRECISION BENDS GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-precision training reduces training time and memory use for transformers. Realizing these gains requires deciding which tensors, operations, and matrix-multiplication passes retain higher precision. Existing precision-allocation methods have focused on local reconstruction error or forward-loss sensitivity. These criteria measure changes in tensor values or the forward loss, but do not directly measure the resulting change in the parameter gradient that drives optimization. We reveal this mismatch by measuring how quantization errors in forward and backward computation change the full model's parameter gradient: Similar local errors at different tensor locations produce gradient shifts whose magnitudes differ by more than tenfold. These shifts mainly rotate, or bend, the gradient. To preserve gradient direction during low-precision training, we introduce bending-guided precision allocation. The method allocates high precision to selected operation groups or individual matrix-multiplication passes where quantization rotates the gradient. Using only a small amount of calibration data, the proposed method determines how much high precision is needed and where to allocate high precision to bring the gradient directions of 4- and 8-bit training close to the bfloat16 (BF16) reference. In GPT-2 continued pretraining, our calibration probe ranks precision allocations before training, closely matching the allocations' final-loss ranking with a Spearman correlation of 0.95, compared with 0.35 for local reconstruction error. Under BF16 mixed-precision evaluation, a model trained in 8-bit floating point (FP8) with our allocation matches the loss of a BF16-trained model. These allocations use 11% to 24% less high-precision activation-cache memory than a fixed policy that stores the output activations of the token embedding, output head, and every normalization layer in high precision. In matrix-multiplication simulations of NVIDIA's NVFP4 format, our probe identifies forward passes as the main source of bending. Executing forward multiplications in BF16 uses high precision for just one third of the transformer-block matrix-multiplication work and closes 99% of the loss gap to BF16 training. At a matched high-precision compute budget, allocating BF16 to selected forward matrix multiplications closes 52% of the gap, compared with at most 20% for NVIDIA-style block protection. The method also improves low-precision training across four vision backbones.

1 INTRODUCTION

Training deep neural networks, including large language models (LLMs), requires substantial computation and memory. Low-precision training reduces these costs by replacing 32- or 16-bit formats with 8- or 4-bit formats for arithmetic and storage. Eight-bit floating-point (FP8) training is used in LLMs such as DeepSeek-V3 (Micikevicius et al., 2022; DeepSeek-AI, 2024), and 4-bit floating-point (FP4) pretraining now includes Nemotron 3 Super, a 120-billion-parameter model trained on 25 trillion tokens (NVIDIA, 2026). Even in these 8- and 4-bit systems, matrix-multiplication accumulators, optimizer statistics, and selected operations stay in high precision to keep training stable. For example, DeepSeek-V3 retains the original high-precision formats for embeddings, the output head, normalization, and attention operators. NVIDIA's pretraining precision policy for the block-scaled 4-bit format (NVFP4) keeps selected blocks in bfloat16 (BF16), while fully quantized FP4 training switches backward matrix multiplications to BF16 late in training (DeepSeek-AI, 2024; NVIDIA, 2025; Chmiel et al., 2025). Deciding which components need high precision for a new model or format often requires trial and error. The central design problem is to decide where to spend a given precision allocation budget while preserving training stability and model performance.

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

Specifically, existing methods address this problem by selecting high-precision locations based on tensor reconstruction error, loss or curvature sensitivity, and hardware cost (Dong et al., 2019; 2020; Lee et al., 2023). Converting tensor values to a low-precision format introduces quantization error. Reconstruction-error scores measure this change in tensor values, while loss-based sensitivity scores assess the effect of quantization error on the forward loss. Neither directly measures the parameter gradient that drives optimization. The optimizer uses this parameter gradient to update the model. Even when these scores indicate small local tensor errors or no change in the forward loss, the parameter-gradient direction can still change substantially and affect low-precision training. Three findings from our experiments show why this gap matters for training.

First, the effect of quantization error on the parameter gradient depends on how the network propagates the error. In GPT-2, applying the same per-tensor FP8 quantizer to the output logits or the final normalization output gives similar relative tensor errors of 2.6% and 2.8%, respectively. Yet quantizing the logits changes the parameter gradient by 62%, compared with only 3% for the final normalization output (Sec. 3). Allocating higher precision by local error would protect the final normalization output first, even though quantization at the logits has a much larger effect on the gradient.

Second, forward-loss scores cannot compare forward and backward quantization: Quantization during backpropagation changes the parameter gradient without changing the forward loss. Local-error scores measure both, but can favor computations whose protection does little to reduce the training loss. In our NVFP4-style GPT-2 experiments, these scores favor protecting the matrix multiplications that compute input gradients because these multiplications have the largest local error. Yet executing the whole backward pass in BF16 closes only 3% of the loss gap to BF16 training, whereas executing the forward multiplications in BF16 closes 99% (Sec. 6). Allocating high precision to the largest local errors thus misses the forward computations that cause most of the training loss gap.

Third, changing gradient norm and changing gradient direction have different effects on optimization. Modern training recipes use learning rate adjustment, clipping, and AdamW's normalization to compensate for changes in gradient norm. These adjustments do not restore the high-precision gradient direction. Our training experiments find no loss penalty from changing only the gradient norm after learning rate tuning, but changing only direction leaves a loss gap of 0.4 to 0.7 relative to BF16 (Sec. 4.2).

To reduce the error in gradient direction that remains after norm adjustment, we measure how closely each candidate precision allocation preserves the high-precision gradient direction. We then allocate high precision to the computations that cause this directional error. Our central objective is:

> Trace quantization errors from the quantization sites to the parameter gradient, and use the resulting directional change to decide before training which tensors, operations, or passes retain high precision.

Figure 1 summarizes the approach. Our contributions are as follows.

- We trace quantization error to the parameter gradient and factor each propagated parameter-gradient contribution into incoming gradient load, directional local error, and propagation gain. Similar local errors yield gradient shifts spanning 17-fold in an 18-layer residual network (ResNet-18) and 32-fold in GPT-2 (Secs. 2 and 3, Appendix B.3).
- We separate the effects of gradient norm and direction by changing one at a time during training. At the best tested learning rates, changing the norm alone incurs no loss penalty, whereas changing direction alone leaves a loss gap of 0.4 to 0.7 relative to BF16 (Sec. 4). Error identities, a one-step bound, and coordinate-wise optimizer analysis formalize this distinction (Appendix B).
- Within a given budget, bending-guided precision allocation identifies which operation groups or individual passes to keep in high precision to preserve gradient direction. The bending–cost path measured on calibration batches predicts the target-level budget and, when one site dominates, the minimum viable budget before training (Sec. 5).
- We validate these predictions in GPT-2 continued pretraining, where calibration bending ranks final losses with Spearman correlations of 0.95 and 0.94 under high-precision memory and compute budgets, compared with 0.35 and 0.50 for local reconstruction error. The se-

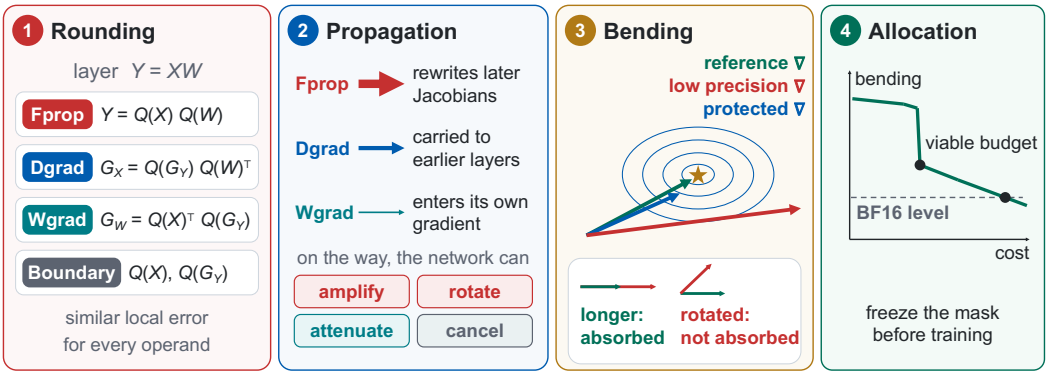


Figure 1: Where low precision enters training and how quantization error rotates the gradient. (1) Low-precision quantizers act on the operands of the three matrix multiplications of each linear layer, or on module-output activations and activation gradients. (2) The errors reach the parameter gradient by different routes. (3) The surviving error mostly rotates the gradient; adjusting the gradient norm through the training recipe cannot undo the rotation. (4) Bending-guided precision allocation estimates the required precision allocation budget from the bending–cost path and freezes the selected mask before training.

lected precision allocations use less high-precision memory than a production-style precision policy and outperform equal-work block protection (Sec. 6, Appendices D.7.1 and E.8.1).

2 Where Low Precision Enters a Training Step

Low-precision training quantizes selected activations, weights, and gradients while keeping other tensors and computations in high precision. We study where quantization is applied in the forward and backward passes and how the resulting errors reach the parameter gradient.

Three Matrix Multiplications per Linear Layer. Consider a linear layer $Y = XW$ with input activation X, weight W, and output Y, and write $G_Z := \partial\mathcal{L}/\partial Z$ for the loss gradient with respect to a tensor Z. Each training step uses three general matrix multiplications (GEMMs) to compute the layer's output and gradients. Forward propagation (Fprop) computes $Y = XW$. The input-gradient GEMM, conventionally labeled Dgrad for data gradient, computes $G_X = G_Y W^\top$, the loss gradient with respect to the layer's input, for propagation to earlier layers. The weight-gradient GEMM (Wgrad) computes $G_W = X^\top G_Y$, the parameter gradient that the optimizer uses to update W.

Low-precision training quantizes selected operands using $Q(\cdot)$, a quantize–dequantize map with a specified format and scaling rule. GEMMs use quantized operand copies, while the optimizer updates the original high-precision weights and keeps the optimizer states in high precision. GEMMs also accumulate products in high precision, as in FP8 and FP4 training systems (Micikevicius et al., 2022; DeepSeek-AI, 2024; NVIDIA, 2026). A precision policy specifies which tensors are quantized, in which passes, and in which format. We call the precision used without these quantizers the reference precision: 32-bit floating point (FP32) for the vision backbones and BF16 mixed precision for GPT-2. We compute the reference and low-precision gradients at the same weights and on the same input batch.

Two Quantization Settings. We study two placements of the quantizers. Boundary quantization acts at a module output, such as a convolution, normalization layer, attention block, or output head. In the forward pass, the boundary quantizer quantizes the output activation, which a subsequent module reads as X and may save for Wgrad. In the backward pass, the boundary quantizer quantizes the incoming gradient G_Y before the module's backward computation. This setting quantizes module outputs, including normalization outputs and logits, and activation gradients; weights remain at reference precision. The cost of protecting a boundary is the extra memory needed to cache the activations at that boundary in high precision. FP8 training systems also cache activations in low

precision to reduce memory (DeepSeek-AI, 2024; Xi et al., 2025). The boundary setting examines which outputs need protection through high-precision allocation.

In the GEMM-operand setting, quantization acts on both operands of every weight GEMM in the four linear layers of each transformer block: $Q(X)Q(W)$ in Fprop, $Q(G_Y)Q(W)^\top$ in Dgrad, and $Q(X)^\top Q(G_Y)$ in Wgrad. This operand pattern is used by FP8 and FP4 tensor-core kernels. In this GEMM-operand setting, attention scores, softmax, normalization, embeddings, and the output head remain at reference precision. The cost of protecting a pass is the GEMM work or time spent in high precision.

How Each Operand Error Reaches the Parameter Gradient. Quantization errors in the three GEMMs reach the parameter gradient by three routes (Eq. 1, Appendix B.2). Wgrad errors enter the gradient of W directly. Dgrad errors propagate to earlier layers through successive local Jacobians and affect the parameter gradients computed by those layers' Wgrad GEMMs. Fprop errors change later activations and thereby the backward computation.

Quantization Sites and Protection. A quantization site refers to a module-output boundary in the boundary setting or the operand pair of one GEMM pass in the GEMM-operand setting. The all-low policy quantizes every eligible site. Protecting a site, a group of sites, or one Fprop, Dgrad, or Wgrad pass restores that part of the graph from low precision to reference precision; these actions form the protected set. After backpropagation, we collect the gradients of all trainable parameters into one vector. A policy's propagated quantization error is the difference between this vector and the reference gradient. A selector chooses the protected set before training, and the resulting pattern of active quantizers is the mask. Different selectors can produce the same mask. A calibration probe evaluates a candidate mask through forward and backward computation on a few calibration batches before training.

Models and Protocols. We apply boundary quantization to four vision backbones and GPT-2. We evaluate GEMM-operand quantization with actual-kernel FP8 training and simulated NVFP4-style and microscaling FP4 (MXFP4) cores. Following standard fake-quantization practice (Jacob et al., 2018; Zhang et al., 2019), we simulate both settings with autograd-compatible quantize–dequantize operations at selected forward and backward sites. The actual-kernel FP8 runs execute scaled FP8 GEMMs on hardware. Our primary boundary policy, called all-low FP8, applies per-tensor FP8 with 4 exponent bits and 3 mantissa bits (FP8-E4M3) to the output activation and incoming activation gradient at every eligible site. The ImageNet[1] precision sweep supports this choice (Table 2, Appendix D.1).

The main decoder experiments continue pretraining GPT-2 with 124M parameters for one pass over 8,192 WikiText-103 sequences of 256 tokens, using AdamW and gradient clipping at 1.0. We report validation loss under each policy's quantizers and, using all-low and BF16 runs paired by seed, the policy's loss-gap closure $100(L_{\text{all-low}} - L_{\text{policy}})/(L_{\text{all-low}} - L_{\text{BF16}})$.

3 Propagated Quantization Error

We trace quantization errors from each site to the full model's parameter gradient. We compare two parameter-gradient vectors computed at the same weights and on the same input batch: one at reference precision and one under a given low-precision policy. Quantizing one site at a time isolates how much each site's error changes that gradient.

Our analysis shows that the size of a site's propagated contribution depends on three factors: how large the arriving gradient is, how much relative error the site adds, and how strongly the network amplifies or attenuates the error along the path to the parameter gradient (Appendix B.3, Eq. 4). These factors multiply: The same relative error can have different effects when the arriving gradient or the propagation gain differs. Measuring local error alone cannot capture how the incoming gradient and propagation through the network affect the parameter gradient.

Local Error Misses Where Precision Matters. Figure 2 compares local reconstruction error with the resulting parameter-gradient shift at each site in the two backbones used in the ImageNet

[1]Throughout this paper, ImageNet denotes ImageNet-1K unless stated otherwise.

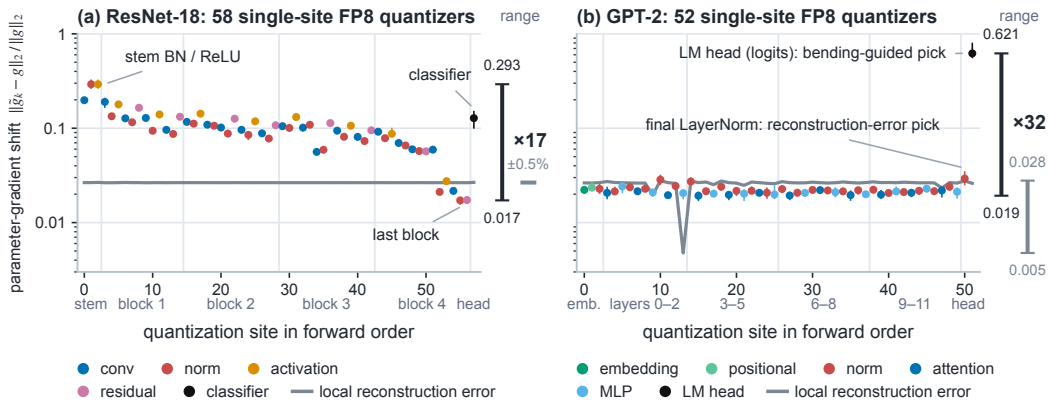


Figure 2: Equal local errors, unequal gradient shifts. The vertical spread shows that similar local errors change the parameter gradient differently depending on where the errors enter the network. Each marker is one module-output site of (a) ResNet-18 or (b) GPT-2 quantized alone by the per-tensor FP8-E4M3 boundary quantizer. Markers show the mean with the minimum-to-maximum range over four calibration batches, colors denote site families, and the parameter-gradient shift is the relative error $\|\widetilde{g}_\theta - g_\theta\|_2/\|g_\theta\|_2$. In (b), the labels mark the first group that bending-guided precision allocation protects and a group that reconstruction-error selection protects.

and decoder language-model (LM) studies. With the dynamic per-tensor scaling used here, E4M3 quantization yields about 2.6% relative reconstruction error at nearly every quantized tensor. The local reconstruction error is nearly identical across the 58 ResNet-18 sites and most of the 52 GPT-2 sites. The corresponding parameter-gradient shifts span 17-fold and 32-fold ranges, respectively, demonstrating that similar local errors can affect the parameter gradient differently depending on the incoming gradient and propagation through the network. This mismatch also appears in the GEMM-operand experiments: Similar operand errors yield gradient-bending values spanning two orders of magnitude (Appendix E.4). Thus, ranking sites for high-precision allocation by local reconstruction error misses the sites that change the parameter gradient most. This mismatch explains why the reconstruction-error masks leave the training loss gap open.

Appendix B.2 gives the network-level error expansion. Appendices D.3, E.4, and D.7.1 provide the single-site protocol, operand-level results, and precision-allocation training results, respectively.

4 What the Optimizer Can Undo: Norm Changes and Bending

We next separate the changes in gradient norm and direction caused by low precision, explain where the rotation comes from, and test how these changes affect optimization.

4.1 Two Coordinates of the Propagated Error

Let g_θ be the full-precision concatenated parameter gradient and $\widetilde{g}_\theta$ the gradient under a low-precision policy with an optional protected set. Gradients with equal relative error can affect optimization differently. For example, doubling the gradient, $\widetilde{g}_\theta = 2g_\theta$, and rotating the gradient by 60° at unchanged norm both give relative error $\|\widetilde{g}_\theta - g_\theta\|_2/\|g_\theta\|_2 = 1$. In a gradient-descent update, halving the optimizer's step size compensates for the doubled norm, whereas no step-size adjustment corrects the rotation. We therefore measure propagated quantization error along two coordinates: gradient bending and the norm ratio. The parameter gradient $\widetilde{g}_\theta$ is computed by backpropagation under the chosen precision policy and supplied to the optimizer.

Angular Gradient Distortion: Gradient Bending. We measure bending with the cosine distance between the reference and precision-policy gradients:

$$D_{\mathrm{g}} := 1 - \frac{\langle g_\theta, \widetilde{g}_\theta \rangle}{\|g_\theta\|_2 \|\widetilde{g}_\theta\|_2}.$$

A bending of $D_{\mathrm{g}} = 0.5$ corresponds to a 60° rotation.

Norm Ratio. The norm ratio is the second primary coordinate of propagated quantization error and measures how much the mixed-precision policy increases or decreases the optimizer gradient norm relative to full precision:

$$R_{\mathrm{norm}} \coloneqq \frac{\|\widetilde{g}_\theta\|_2 + \epsilon_{\mathrm{guard}}}{\|g_\theta\|_2 + \epsilon_{\mathrm{guard}}}, \qquad S_{\mathrm{norm}} \coloneqq |\log R_{\mathrm{norm}}|.$$

$R_{\mathrm{norm}} > 1$ indicates norm amplification, and $R_{\mathrm{norm}} < 1$ indicates norm attenuation.

4.2 Gradient Bending and the Training Gap

Our low-precision training experiments show that errors in gradient direction are the main source of the loss gap.

Sign Flips under AdamW. In the first bias-corrected AdamW step, each parameter's gradient component is divided by the component's own magnitude, reducing the gradient contribution to approximately $+1$ or -1 before learning rate scaling (Appendix E.4). A gradient component with small magnitude therefore produces nearly the same update magnitude as a component with large magnitude, so flipping the small component's sign can barely change the raw gradient while reversing that parameter's update. For example, changing $(1, 0.01)$ to $(1, -0.01)$ barely rotates the gradient, but the sign vectors $(1, 1)$ and $(1, -1)$ are perpendicular. The update bending D_{u}, the cosine distance between reference and policy AdamW updates, is approximately twice the sign-flip fraction in this regime (Appendix B.7).

Learning Rate Tuning Cannot Undo Bending. We isolate the effects of norm and direction by changing one at a time while holding the other at the BF16 value (Figure 7 in Appendix D.9). At each step, we compute a BF16 gradient and a gradient under all-low NVFP4-style boundary quantization on the same batch. In the norm-only setting, we give the optimizer a gradient with the BF16 direction and the NVFP4 gradient's norm. In the direction-only setting, we give the optimizer a gradient with the NVFP4 direction and the BF16 gradient's norm.

After learning rate tuning, changing only the gradient norm introduces no loss penalty. AdamW in the norm-only setting achieves a final loss within 0.007 of BF16 at every tested learning rate. For stochastic gradient descent (SGD) with momentum, a constant gradient scaling factor would be equivalent to changing the learning rate, but here the factor varies across steps.

Changing only the gradient direction leaves a loss gap at every tested rate, despite matching the BF16 gradient norm at each step. At the best rate, the direction-only three-seed mean loss is 0.68 above BF16 under SGD and 0.40 above BF16 under AdamW.

5 Bending-Guided Precision Allocation

Sec. 4 identified gradient bending as the error that scalar rescaling cannot correct. We now propose a rule that reduces gradient bending within a given precision allocation budget. To predict the required budget, we trace a bending–cost path on calibration batches, measuring how much bending remains as more computations receive high precision.

A protection action assigns reference precision to part of the training graph, such as the LM-head boundary or an attention Fprop pass. Let $\mathcal{Q}$ be the finite set of candidate protection actions. Each action protects a group of module boundaries, a single boundary, or a single GEMM pass. For boundary quantization, a group holds the sites of one operation type, one depth band, or both, *e.g.*, all normalization boundaries, the early layers, or the early normalization boundaries. For GEMM-operand quantization, an action protects one Fprop, Dgrad, or Wgrad pass of the attention or multilayer perceptron (MLP) GEMMs in one layer. We protect the sites or GEMM passes selected by $\mathcal{R} \subseteq \mathcal{Q}$, the protected set, by assigning high precision to those sites or passes. The remaining eligible sites use the low-precision policy.

The cost $c_q > 0$ is the amount of the precision allocation budget used to protect action q. For a memory budget, the cost is the number of activation-cache elements stored in high precision, which

determines the extra cache memory. For a compute budget, the cost is the GEMM work assigned to high precision or the time spent on those GEMMs (Appendix C.1). These costs add: The total cost $c(\mathcal{R}) \coloneqq \sum_{q \in \mathcal{R}} c_q$ must fit within the budget B, so $c(\mathcal{R}) \leq B$.

We measure the bending of each candidate mask on the calibration batches and write this bending as $D(\mathcal{R})$. We use $D = D_{\mathrm{g}}$ for raw gradients and $D = D_{\mathrm{u}}$ for AdamW updates. Two selectors that choose the same mask produce the same precision allocation for training.

Bending-Guided Precision Allocation: Overview

The inputs are the starting weights, calibration batches, candidate protection actions $q \in \mathcal{Q}$ with costs c_q, and a precision allocation budget B.

We first compute the reference gradient and the BF16 target bending level τ, then measure all-low gradient bending. All-low gradient bending near the BF16 target level τ indicates little need for protection. Starting from the all-low mask, we try adding each protection action that fits the remaining budget and choose the action that removes the most gradient or update bending per unit cost. We repeat until no action that fits the budget lowers bending. We then compare the selected set with the best single action within the budget, keep the choice with lower bending, and freeze the mask before training.

The output is a protected set and a bending–cost path showing the remaining bending at each allocation cost. The first point with raw-gradient bending below τ gives the predicted budget needed to reach the target level. When one action removes most of the bending, the sharp drop at that action's cost identifies the minimum viable budget for substantial repair.

Appendix C.1 gives the formal selection rule, path interpretation, reference-floor measurements, baseline scores, and search-cost details for the bending-guided precision allocation described above.

Target Bending Level: The BF16 Reference. BF16 mixed precision has been adopted in practice as an alternative to FP32 training with comparable performance. We therefore use the gradient bending introduced by BF16 arithmetic as the target for precision allocation. We measure this target τ, called the BF16 reference floor, as the bending between the BF16 mixed-precision gradient and the FP32 gradient on the same calibration batches. An allocation reaches the target when the raw-gradient bending D_{g} of that allocation falls below τ.

6 Experiments: Budget Predictions and Training Outcomes

We test whether gradient bending measured before training can identify where to allocate precision efficiently and how much budget is required for low-precision training to learn weights with near-BF16 loss. Our main experiments continue pretraining GPT-2 124M under two constraints: the memory used to cache activations in high precision, and the amount of GEMM work executed in BF16. GEMM work counts the multiply–accumulate operations in the transformer blocks' linear layers per training step. Using only the starting weights and a few calibration batches, the probe selects the module boundaries or GEMM passes to keep in high precision and estimates the cost of reaching the BF16 bending target. We freeze each selected mask before the first training step and train GPT-2 under each fixed mask with three paired seeds.

To compare the quality of the learned weights with BF16 training, we also evaluate each trained model with the quantizers disabled. We measure this unquantized loss at a shared evaluation precision to ensure that the remaining loss gap reflects what was learned during training.

Memory Budget: Protecting the Logits. In GPT-2 124M with FP8 boundary quantizers, the probe identifies the LM-head output, the logits, as the first activations to keep in high precision. Protecting the LM-head output boundary costs 56.4% of the eligible activation-cache footprint, which corresponds to the predicted minimum viable budget for this model. Protecting the logits removes 96.8% of the all-low gradient bending, whereas no protected set without that boundary removes more than 12.3% of that gradient bending. Low-precision continued pretraining validates this predicted budget threshold on every seed: Masks that leave the logits in FP8 close at most 1% of the loss gap, and every mask that protects the logits closes at least 99.7%. Keeping the logits in high precision is therefore the decisive first allocation for recovering the FP8 training loss.

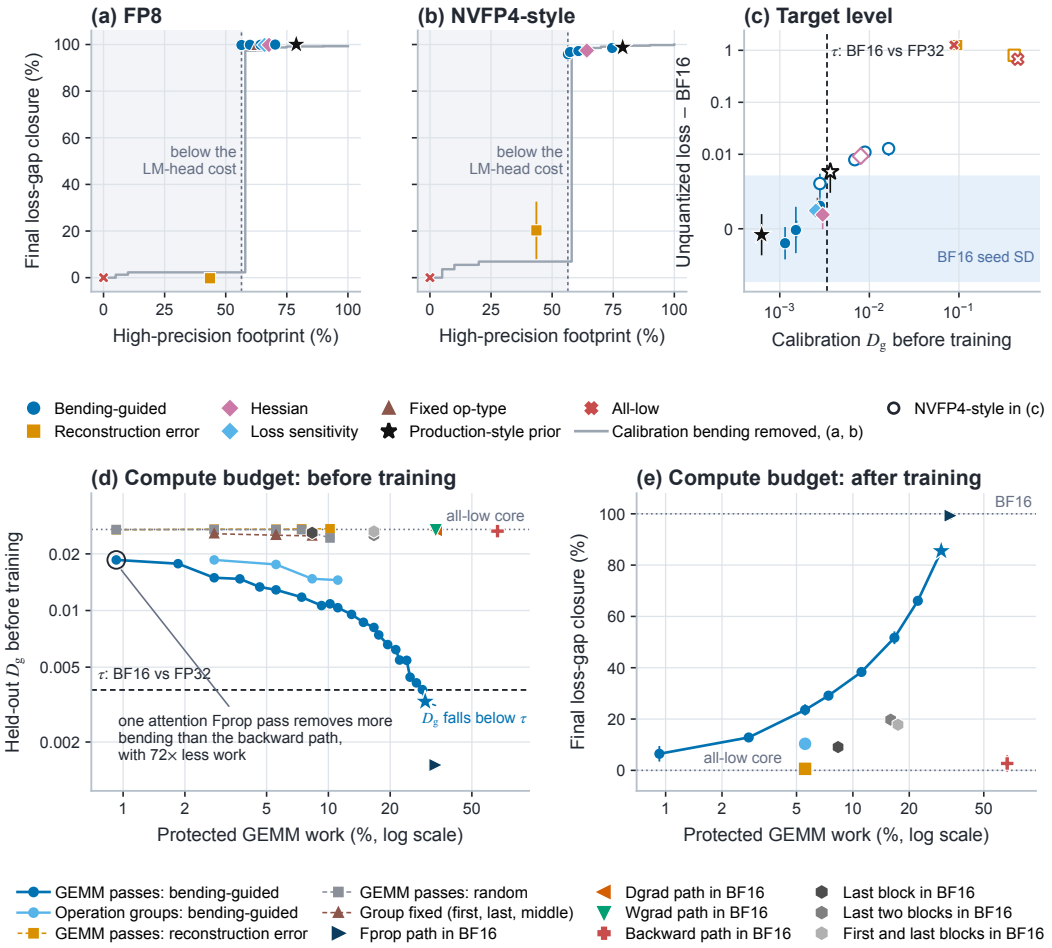


Figure 3: Gradient bending measured on calibration batches accurately predicts where to allocate precision to close the loss gap to BF16. Results are for bending-guided precision allocation under high-precision memory and compute budgets in GPT-2 continued pretraining, with three seeds per mask; bars span one standard deviation. Loss-gap closure is the fraction of the all-low policy's loss gap to BF16 training removed by each allocation. (a, b) Final loss-gap closure against the high-precision boundary footprint. (c) Excess of the unquantized loss over BF16 training against gradient bending D_g measured on calibration batches. (d) Held-out raw-gradient bending D_g before training against protected GEMM work in the NVFP4-style core; bending-guided paths score AdamW update bending D_u, the directional difference between the policy and BF16 updates. (e) Final loss-gap closure of the trained compute masks; stars in (d, e) mark the target-level mask.

The same LM-head protection already reaches the BF16 bending target: $D_g = 0.0028$ is below $\tau = 0.0034$, corresponding to a rotation of $4.3°$ against the BF16 reference's $4.7°$. For every trained boundary mask below τ, the unquantized loss exceeds BF16 training by less than the BF16 seed-to-seed standard deviation of 0.0037 on average over seeds. Reaching the BF16 bending target thus identifies allocations that learn weights with loss close to BF16 training.

Memory Budget: Additional Protection at Four Bits. With NVFP4-style boundary quantization in the same GPT-2 model, the 56.4% cost of protecting the LM-head boundary again corresponds to the predicted minimum viable budget. Protecting all groups other than the LM head closes about 20% of the loss gap, compared with 96% from protecting that boundary alone. For these 4-bit quantizers, protecting the logits leaves more bending than the BF16 target permits. The probe therefore selects

additional boundaries for protection to reach the BF16 bending target, using 74.6% of the activation-cache footprint, compared with 56.4% under FP8. The weights learned with this allocation have an unquantized loss 0.003 above BF16 on average over seeds. The fixed production-style policy, which protects the token embedding, output head, and all normalization outputs, uses 78.9% of the footprint, remains above the bending target, and reaches a final unquantized loss 0.005 above BF16. The selected allocation thus reaches the target with less high-precision activation storage than this fixed policy.

Compute Budget: Protecting Forward GEMMs. For the simulated NVFP4-style GEMM core of GPT-2, the probe identifies forward GEMMs as the main source of bending. Executing all Fprop GEMMs in BF16 uses one third of the GEMM work and removes 94% of the held-out bending, whereas executing all Dgrad and Wgrad GEMMs in BF16 uses two thirds of the work and removes only 2%. In the corresponding low-precision continued-pretraining runs, protecting Fprop closes 99.3% of the loss gap, while protecting Dgrad and Wgrad closes 2.7%. Forward protection therefore recovers almost the entire loss gap with half the BF16 GEMM work required to protect the entire backward pass.

At 16.7% of the GEMM work, the mask over individual GEMM passes removes 70% of the bending and closes 51.7% of the loss gap. At the same cost, NVIDIA-style protection of the last two or the first and last blocks removes 7% or 2% of the bending and closes 19.8% or 17.8% of the gap, respectively. Selecting individual forward passes therefore recovers more than twice as much of the loss gap as protecting whole blocks at the same compute budget. The probe predicts that the BF16 bending target is reached at 29.6% of the GEMM work. The mask selected at this budget closes 85.5% of the loss gap and learns weights whose unquantized loss is within 0.004 of BF16 on average over seeds.

When No Additional Protection Is Needed. The actual-kernel FP8 GEMM core starts near the BF16 bending target, with $D_{\mathrm{g}} = 0.004$ to 0.005, so the probe predicts that all eligible block GEMMs can remain in FP8. All six from-scratch runs, three each at 124M and 355M parameters, reach final losses within 1% of the paired BF16 losses. The low gradient bending measured before training correctly predicts that FP8 can train close to BF16 without protecting any GEMM passes.

Predicting Which Allocations Train Better. Table 11 in Appendix D.7 compares how well scores measured before training rank the final loss gap. Gradient bending ranks the masks with Spearman correlations of 0.95 under memory budgets and 0.94 under compute budgets, compared with 0.35 and 0.50 for local reconstruction error. Activation-Fisher, a baseline that uses activation loss gradients to score quantization error, also ranks the boundary masks well, with a correlation of 0.98. Bending additionally scores Dgrad and Wgrad quantization, which leaves the forward loss unchanged, so bending can compare forward and backward protection using the same criterion. These rankings allow candidate allocations to be compared before spending the time to train each one.

Appendices D.7 and E.8 provide the full calibration and training results for the memory and compute budgets, and Appendix C.1 defines the baseline scores and selection rules. Appendix D.11 reports vision fine-tuning results, and Appendix E.1 compares formats and identifies precision settings that need no additional protection, including the actual-kernel FP8 control.

7 Conclusion

We traced quantization errors from the activations, weights, and gradients of a training step to the parameter gradient. Similar local errors can produce gradient shifts differing by more than tenfold along different propagation routes. Predicting the training loss gap requires measuring gradient bending, not only the gradient norm: Norm changes can be absorbed by the training recipe, while directional distortion persists even after learning rate adjustment.

We therefore recommend a practical calibration procedure for efficient low-precision training: Using the gradient bending caused by quantization as a direct criterion for precision allocation, compare all-low bending with the BF16 floor, trace the bending–cost path when needed, and choose a mask within the available budget to freeze before training.

AI USE STATEMENT

Generative AI tools were used as assistive tools for writing and editing, mathematical proofreading and consistency checking, and code development and debugging. Any AI-assisted outputs incorporated into this work were reviewed and validated by the authors as appropriate. The authors take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies the numerical behavior of low-precision training. We are not aware of any ethical concerns specific to this study.

REPRODUCIBILITY STATEMENT

Implementation details, hyperparameters, evaluation protocols, and seed settings are described in Appendices C.2 through C.4, D.9, and E.2.

REFERENCES

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *CoRR*, abs/1607.06450, 2016.

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *NeurIPS*, pp. 5151–5159, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3432, 2013.

Roberto L. Castro, Andrei Panferov, Rush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native FP4 Training Can Be Optimal for Large Language Models. In *NeurIPS*, 2025.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. FP4 All the Way: Fully Quantized Training of Large Language Models. In *NeurIPS*, 2025.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *CoRR*, abs/1805.06085, 2018.

Musa Oguzhan Cim, Poovaiah Palangappa, Miro Hodak, Ravi Dwivedula, Meena Arunachalam, and Mahmut Taylan Kandemir. Pretraining large language models with MXFP4 on Native FP4 Hardware. *CoRR*, abs/2605.09825, 2026.

Michael P. Connolly, Nicholas J. Higham, and Théo Mary. Stochastic Rounding and Its Probabilistic Backward Error Analysis. *SIAM J. Sci. Comput.*, 43(1):A566–A585, 2021.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *NIPS*, pp. 3123–3131, 2015.

DeepSeek-AI. DeepSeek-V3 Technical Report. *CoRR*, abs/2412.19437, 2024.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare Quantization of Neural Networks With Mixed-Precision. In *ICCV*, pp. 293–302, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks. In *NeurIPS*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

Mario Drumond, Tao Lin, Martin Jaggi, and Babak Falsafi. Training DNNs with Hybrid Block Floating Point. In *NeurIPS*, pp. 451–461, 2018.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned Step Size quantization. In *ICLR*, 2020.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. *CoRR*, abs/2103.13630, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, pp. 249–256, 2010.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *CoRR*, abs/2312.00752, 2023.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *ICML*, volume 37, pp. 1737–1746, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pp. 770–778, 2016.

Nicholas J. Higham. *Accuracy and stability of numerical algorithms, Second Edition*. 2002.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *J. Mach. Learn. Res.*, 18:187:1–187:30, 2017.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, volume 37, pp. 448–456, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *CVPR*, pp. 2704–2713, 2018.

Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A Study of BFLOAT16 for Deep Learning Training. *CoRR*, abs/1905.12322, 2019.

Samir Khuller, Anna Moss, and Joseph Naor. The Budgeted Maximum Coverage Problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K. Bansal, William Constable, Oguz Elibol, Stewart Hall, Luke Hornof, Amir Khosrowshahi, Carey Kloss, Ruby J. Pai, and Naveen Rao. Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks. In *NIPS*, pp. 1742–1752, 2017.

Andrey Kuzmin, Mart van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. FP8 Quantization: The Power of the Exponent. In *NeurIPS*, 2022.

Wonyeol Lee, Rahul Sharma, and Alex Aiken. Training with Mixed-Precision Floating-Point Assignments. *Trans. Mach. Learn. Res.*, 2023.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, pp. 9992–10002, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, pp. 11966–11976, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

Naveen Mellempudi, Sudarshan Srinivasan, Dipankar Das, and Bharat Kaul. Mixed Precision Training With 8-bit Floating Point. *CoRR*, abs/1905.12334, 2019.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed Precision Training. In *ICLR*, 2018.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart F. Oberman, Mohammad Shoeybi, Michael Y. Siu, and Hao Wu. FP8 Formats for Deep Learning. *CoRR*, abs/2209.05433, 2022.

NVIDIA. Pretraining Large Language Models with NVFP4. *CoRR*, abs/2509.25149, 2025.

NVIDIA. Nemotron 3 Super: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning. *CoRR*, abs/2604.12374, 2026.

Open Compute Project. OCP Microscaling Formats (MX) Specification, Version 1.0. Technical report, Open Compute Project, September 2023. URL `https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf`.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML (3)*, volume 28, pp. 1310–1318, 2013.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. FP8-LM: Training FP8 Large Language Models. *CoRR*, abs/2310.18313, 2023.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *NIPS*, pp. 4785–4795, 2017.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Y. Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verilli, Ralph Wittig, Doug Burger, and Eric S. Chung. Microscaling Data Formats for Deep Learning. *CoRR*, abs/2310.10537, 2023.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

Noam Shazeer. GLU Variants Improve Transformer. *CoRR*, abs/2002.05202, 2020.

Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks. In *NeurIPS*, pp. 4901–4910, 2019.

Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Ultra-Low Precision 4-bit Training of Deep Neural Networks. In *NeurIPS*, 2020.

Albert Tseng, Tao Yu, and Youngsuk Park. Training LLMs with MXFP4. In *AISTATS*, volume 258, pp. 1630–1638, 2025.

Satoki Tsuji, Hiroshi Kawaguchi, Atsuki Inoue, Yasufumi Sakai, and Fuyuka Yamada. Greedy Search Algorithm for Mixed Precision in Post-Training Quantization of Convolutional Neural Network Inspired by Submodular Optimization. In *ACML*, volume 157, pp. 886–901, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NIPS*, pp. 5998–6008, 2017.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-Aware Automated Quantization With Mixed Precision. In *CVPR*, pp. 8612–8620, 2019.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training Deep Neural Networks with 8-bit Floating Point Numbers. In *NeurIPS*, pp. 7686–7695, 2018.

James Hardy Wilkinson. Rounding errors in algebraic processes. In *IFIP Congress*, pp. 44–53, 1959.

Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. Stable and low-precision training for large-scale vision-language models. In *NeurIPS*, 2023.

Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and Inference with Integers in Deep Neural Networks. In *ICLR*, 2018.

Haocheng Xi, Yuxiang Chen, Kang Zhao, Kai Jun Teh, Jianfei Chen, and Jun Zhu. Jetfire: Efficient and Accurate Transformer Pretraining with INT8 Data Flow and Per-Block Quantization. In *ICML*, volume 235, pp. 54049–54063, 2024.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. COAT: Compressing Optimizer states and Activations for Memory-Efficient FP8 Training. In *ICLR*, 2025.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, and Jeffrey Pennington. Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10, 000-Layer Vanilla Convolutional Neural Networks. In *ICML*, volume 80, pp. 5389–5398, 2018.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On Layer Normalization in the Transformer Architecture. In *ICML*, volume 119, pp. 10524–10533, 2020.

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V3: Dyadic Neural Network Quantization. In *ICML*, volume 139, pp. 11875–11886, 2021.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *NeurIPS*, pp. 12360–12371, 2019.

Tianyi Zhang, Zhiqiu Lin, Guandao Yang, and Christopher De Sa. QPyTorch: A Low-Precision Arithmetic Simulation Framework. In *EMC2@NeurIPS*, pp. 10–13, 2019.

Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *CoRR*, abs/1606.06160, 2016.

APPENDIX TABLE OF CONTENTS

Appendix B.3 extends the layer-level error analysis of Sec. 2 to the full parameter gradient. Appendices C.1 through C.4 list eligible sites and provide implementation, evaluation, and seed-pairing details. Boundary-quantization results appear in Sec. 3 and Appendices D.11 and D.7.1; GEMM-operand results appear in Appendices D.14 and E.

A Related Work **15**
A.1 Overview . . . 15
A.2 Extended Related Work . . . 15

B Formalism, Diagnostics, and Proofs **17**
B.1 List of Notation . . . 17
B.2 From One Layer to the Network . . . 17
B.3 From Activation Gradients to the Parameter Gradient . . . 20
B.4 Realized-Pass Identity, Shared Parameters, and Terminal and Forward-Induced Perturbations . . . 21
B.5 Mechanism Taxonomy . . . 23
B.6 Derived Diagnostics, Metric Identities, and One-Step Bound Details . . . 24
B.7 Update Geometry under Coordinate-Wise Optimizers . . . 27
B.8 Implications for Precision Allocation . . . 29
B.9 Protected-Set Curvature under the First-Order Surrogate . . . 29
B.10 Proofs . . . 30

C Implementation, Protocols, and Code **34**
C.1 Selection-Rule and Baseline Details . . . 34
C.2 Implementation and Protocol Details . . . 38
C.3 Evaluation Protocols . . . 39
C.4 Seed Protocol and Pairing Details . . . 41
C.5 Core Python Source Code . . . 42

D Extended Boundary-Quantizer Results and Actual-Kernel Checks **43**
D.1 ImageNet Precision Sweep . . . 43
D.2 Direct Visualization of Gradient Bending . . . 44
D.3 Site-Level Factorization and Pairwise Alignment . . . 44
D.4 Mechanistic Ablations . . . 46
D.5 Longer-Horizon Decoder-LM Continued Pretraining . . . 48
D.6 Decoder-LM Protection Diagnostics . . . 49
D.7 Budget-Matched Decoder Allocation . . . 50
D.8 Decoder-LM Decomposition and Trajectory . . . 56
D.9 Learning Rate Tuning Cannot Undo Bending . . . 59
D.10 Quantization Error Repeats across Batches . . . 60
D.11 Transfer to Vision Fine-Tuning . . . 62
D.12 ResNet-18 Transfer Fine-Tuning Details . . . 62
D.13 Modern Backbone Fine-Tuning Details . . . 63
D.14 Actual-Kernel FP8 GEMM Microbenchmarks . . . 63

E Controlled FP4 Cores and Optimizer-Update Geometry **64**
E.1 Diagnosing Formats and Identifying Well-Conditioned Precision Settings . . . 65
E.2 Hardware Scope and Experimental Controls . . . 65
E.3 Controlled Block–Scale Factorial . . . 67
E.4 Fprop, Dgrad, Wgrad, and Direct Readout Error . . . 68
E.5 Incoming Load, Local Directional Error, and Propagation . . . 71
E.6 Allocating Precision to Individual GEMM Passes at Equal Work . . . 72
E.7 Protected GEMM Work, Modeled Time, and Update Bending . . . 74
E.8 Compute-Budget Allocation in Continued Pretraining . . . 76
E.9 Full-Budget From-Scratch Trajectories . . . 79
E.10 Scale-Up to 355M and 1.56B Parameters . . . 80
E.11 Matched Actual-Kernel FP8 Accuracy Baseline . . . 81
E.12 Statistical Protocol, Artifact Audit, and Next Tests . . . 82

F Discussion **83**
F.1 Evidence and Practical Tradeoffs . . . 83
F.2 Systems Coverage and Generalization . . . 84

G Limitations **85**

Overview of the Analysis and Allocation

Our analysis starts with the three GEMMs of a linear layer $Y = XW$: One computes the forward output, and two compute the input and weight gradients (Sec. 2). Quantization in these passes affects three types of operands: activations, weights, and incoming gradients. We trace the resulting errors to the parameter gradient (Sec. 3). Along this route, the network can amplify, attenuate, or reorient an error, and errors from different sites can reinforce or cancel one another.

We describe the resulting propagated quantization error through two coordinates: the norm ratio, which compares the low- and high-precision gradient norms, and gradient bending, which measures the change in direction (Sec. 4). Our measurements show that quantization primarily produces bending. Forward quantization can flip rectified linear unit (ReLU) masks, and AdamW amplifies the effect of sign changes in small gradient coordinates. Bending-guided precision allocation uses these measurements to select the operation groups or individual passes to keep in high precision (Sec. 5). Measuring the gradient shows how much protecting a forward or backward computation restores the high-precision direction, even when the forward loss does not change.

Our proposed rule uses a small number of calibration batches to trace a bending–cost path, showing how much bending remains as more computations receive high precision. Tracing this path takes seconds for module groups and under an hour for individual passes. This path predicts the target-level budget, where bending approaches the level introduced by BF16 arithmetic, and, when one site dominates, the minimum viable budget needed for substantial repair. Both predictions are made before training begins. Figure 1 summarizes this progression from quantization to propagation, gradient geometry, and precision allocation.

A Related Work

A.1 Overview

Precision Allocation and Sensitivity. Mixed-precision allocation is usually posed as a search or sensitivity problem. Hardware-aware search uses device feedback, Hessian-aware methods weight quantization residuals by local loss curvature, greedy inference-time searches update accuracy-based sensitivity after each choice, and training-time allocation trades memory against accuracy over tensors (Wang et al., 2019; Dong et al., 2019; 2020; Yao et al., 2021; Tsuji et al., 2021; Lee et al., 2023). These criteria supply our reconstruction-error, loss-sensitivity, activation-Fisher, finite-difference Hessian, and budgeted knapsack baselines. Our selector differs in the scored quantity: Prior criteria use tensor reconstruction, loss or curvature sensitivity, end-to-end accuracy, or hardware cost, whereas bending-guided precision allocation measures the marginal distortion of the realized concatenated optimizer gradient under the currently executed mask. Budgeted Hessian-aware allocation solves an integer program over modular sensitivity scores (Yao et al., 2021); we use the same budget formulation but score executed masks and select protected masks with the cost-normalized greedy rule of Khuller et al. (1999).

Appendix A.2 reviews the low-precision format and systems literature and analyses of gradient propagation through architecture, and gives the full comparison with the sensitivity literature summarized above.

A.2 Extended Related Work

This subsection expands the overview above with the full format, systems, and sensitivity comparison.

Low-Precision Training Formats and Methods. Low-precision training spans stochastic rounding, fixed-point and block floating-point arithmetic, 16-bit floating-point (FP16) and BF16 mixed precision, FP8, and 4-bit formats (Connolly et al., 2021; Gupta et al., 2015; Köster et al., 2017; Banner et al., 2018; Wang et al., 2018; Micikevicius et al., 2018; Kalamkar et al., 2019; Sun et al., 2019; Mellempudi et al., 2019; Micikevicius et al., 2022; Rouhani et al., 2023; Peng et al., 2023; Sun et al., 2020). Across these formats, the recurring design problem is how to allocate precision across tensors and operations without sacrificing task performance.

Quantized and Mixed-Precision Training. Early quantized-training methods showed that weights, activations, parameter gradients, and layer errors can be represented with few bits during optimization, typically while retaining higher-precision state and using straight-through estimators for nondifferentiable quantizers (Bengio et al., 2013; Courbariaux et al., 2015; Zhou et al., 2016; Hubara et al., 2017; Wu et al., 2018). Learned clipping thresholds and quantizer step sizes subsequently enabled scales to be learned during training (Choi et al., 2018; Esser et al., 2020). Fixed-point, shared-exponent, hybrid block-floating-point, FP16, and BF16 training methods established the importance of careful rounding, dynamic range, loss scaling, and selective high-precision accumulation or state (Gupta et al., 2015; Köster et al., 2017; Drumond et al., 2018; Micikevicius et al., 2018; Kalamkar et al., 2019). Quantization-aware training and efficient-inference surveys systematize fake-quantized graphs, calibration, scaling, and hardware-facing accuracy–efficiency tradeoffs (Jacob et al., 2018; Gholami et al., 2021). This literature establishes the viability of low-precision training and the need to treat different tensors and operations differently.

FP8, Microscaling, and Large-Model Systems. FP8 work makes exponent allocation, scaling, outlier handling, and operation-specific formats explicit design variables, including in LLM training (Kuzmin et al., 2022; Micikevicius et al., 2022; Peng et al., 2023). Microscaling and ultra-low-precision studies further expose the effects of block size, shared scales, gradient scaling, and quantization bias (Sun et al., 2020; Rouhani et al., 2023). Recent FP4 training methods combine a block-scaled core, whose FP4 values have 2 exponent bits and 1 mantissa bit (E2M1), with stabilization mechanisms such as stochastic rounding, Hadamard transforms, two-dimensional scaling, selective high precision, or a late quantization-aware phase (Tseng et al., 2025; Castro et al., 2025; Chmiel et al., 2025; NVIDIA, 2025; Cim et al., 2026). Systems including SwitchBack, Jetfire, and large-scale FP8 mixture-of-experts training emphasize kernel and data-flow coverage alongside numerical stability (Wortsman et al., 2023; Xi et al., 2024; DeepSeek-AI, 2024), and COAT compresses activations and optimizer states to FP8 to reduce training memory (Xi et al., 2025); simulation frameworks such as QPyTorch support controlled studies before matching kernels are available (Zhang et al., 2019). These works provide the format and systems context for our actual-kernel FP8 evaluation and controlled simulated-FP4 comparisons. We treat a format and scaling policy as given and ask where the resulting error matters.

Mixed-Precision Allocation and Sensitivity. Mixed-precision allocation is commonly posed as a search or sensitivity problem. Hardware-aware automated quantization uses device feedback, whereas Hessian-aware methods weight quantization residuals by local loss curvature (Wang et al., 2019; Dong et al., 2019; 2020; Yao et al., 2021). Greedy inference-time searches update accuracy-based sensitivity after each choice (Tsuji et al., 2021), and training-time floating-point allocation explicitly trades memory against accuracy over tensors produced during optimization (Lee et al., 2023). These methods motivate our reconstruction-error, loss-sensitivity, activation-Fisher, finite-difference Hessian, and budgeted baselines. The distinction is the scored object: Prior criteria use tensor reconstruction, loss or curvature sensitivity, end-to-end accuracy, memory cost, or hardware feedback, whereas our selector measures the marginal distortion of the realized concatenated optimizer gradient under the currently executed precision mask.

Architecture and Gradient Propagation. Residual paths, normalization, attention, gated feed-forward layers, and recurrent or state-space dynamics impose different local Jacobian structures and backward paths (He et al., 2016; Ioffe & Szegedy, 2015; Ba et al., 2016; Zhang & Sennrich, 2019; Xiong et al., 2020; Vaswani et al., 2017; Shazeer, 2020; Gu & Dao, 2023). Analyses of initialization, deep-linear dynamics, vanishing or exploding gradients, and dynamical isometry characterize how products of these Jacobians preserve, amplify, or attenuate signals across depth or time (Glorot & Bengio, 2010; Saxe et al., 2014; Pascanu et al., 2013; Pennington et al., 2017; Xiao et al., 2018). This line of work explains why architecture shapes gradient propagation, but existing analyses have generally not injected realized finite-precision perturbations at internal computation-graph boundaries and traced the resulting readout into the concatenated parameter gradient.

Temporal Switching and Spatial Protection. Large-scale FP4 pretraining is now practical (Chmiel et al., 2025; NVIDIA, 2026), and Chmiel et al. (2025) close the remaining gap with a brief quantization-aware phase that uses FP4 for forward computation while switching the backward and update GEMMs to BF16. That switch is temporal and protects the whole backward path; the

protection studied here is spatial and covers the few sites and passes whose error survives propagation, including forward passes, which Sec. 2 identifies as the route that changes the backward computation. In our continued-pretraining study with an NVFP4-style GEMM core, executing the backward path in BF16 throughout training closes 2.7% of the loss gap, whereas executing the forward passes in BF16 closes 99.3% (Appendix E.8.1).

Recent large-scale results demonstrate the practical viability of FP4 training, including fully quantized pretraining at the 7-billion-parameter scale and Nemotron 3 Super's 25-trillion-token NVFP4 pretraining (Chmiel et al., 2025; NVIDIA, 2026). Chmiel et al. (2025) further show that a brief quantization-aware phase can close the remaining BF16 loss gap by retaining FP4 forward computation while switching the backward and update GEMMs to BF16. The quantization-aware intervention is temporal and protects the full backward path. Our protection is spatial: Bending-guided precision allocation locates where propagation through the architecture rotates the optimizer gradient and allocates high precision to the selected FP8 or NVFP4 sites or individual GEMM passes. The decoder-LM experiments of Appendices D.7.1 and E.8.1 and the study of individual GEMM passes of Appendix E examine this complementary allocation axis.

Positioning of This Study. This study connects the preceding lines without proposing another numerical format. A perturbation is local only by measurement location; the perturbation's optimizer-level importance depends jointly on the incoming adjoint, the directional local error, and propagation through the architecture into parameter-gradient readouts. We decompose the resulting gradient distortion into angular bending and the norm ratio, measure interactions under each executed mask, and select small high-precision protected sets according to the measured repair produced by each set. The update-space analysis extends the same view to coordinate-wise maps such as common-state AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). Relative to format and systems work, the contribution is a diagnostic and selection objective; relative to sensitivity-based allocation, the contribution acts directly on the realized optimizer-gradient vector; relative to gradient-propagation analyses, the contribution ties Jacobian propagation to finite-precision error and parameter-level optimization. The resulting signal complements accuracy, memory, and throughput when choosing a deployable precision policy.

B Formalism, Diagnostics, and Proofs

This appendix collects the notation, the first-order expansions and exact identities supporting Sec. 3, the mechanism taxonomy, the derived diagnostics associated with Sec. 4, and all proofs.

B.1 List of Notation

Table 1 collects the principal symbols used in the main text and the appendix, grouped by role: the linear layer of Sec. 2 and the corresponding operand errors, quantization sites and Jacobians, adjoints and local errors, propagation and readout operators, optimizer-gradient diagnostics, update geometry, protected sets, and the FP4 scale factors of Appendix E.

B.2 From One Layer to the Network

Write $\delta X \coloneqq Q(X) - X$ for the quantization error of X and define δW and δG_Y in the same way. When every operand is quantized, expanding the three GEMMs gives the exact errors

$$\begin{aligned} \text{Fprop:} \quad & \widetilde{Y} - Y = \delta X W + X \delta W + \delta X \delta W, \\ \text{Dgrad:} \quad & \widetilde{G}_X - G_X = \delta G_Y W^\top + G_Y \delta W^\top + \delta G_Y \delta W^\top, \\ \text{Wgrad:} \quad & \widetilde{G}_W - G_W = \delta X^\top G_Y + X^\top \delta G_Y + \delta X^\top \delta G_Y; \end{aligned} \tag{1}$$

Under boundary quantizers, $\delta W = 0$ and only the terms in δX and δG_Y remain. Under GEMM-operand quantization, each pass quantizes a separate copy of an operand along the reduction dimension of that GEMM, so the Fprop and Wgrad copies of δX differ, as do the two copies of δW and of δG_Y; under boundary quantizers, a single quantized copy of each operand, X or G_Y, serves both passes that read that operand.

Table 1: Principal notation used throughout the paper. Closely related symbols are grouped.

Symbol	Meaning
X, W, Y; G_Z	Linear-layer input, weight, and output; gradient $\partial\mathcal{L}/\partial Z$
$Q(\cdot)$; $\delta X, \delta W, \delta G_Y$	Quantize–dequantize map; operand quantization errors
ℓ, k; L	Quantization-site indices; total number of sites
$\theta, \mathcal{L}$	Trainable parameters; reference loss
$h_\ell, \widetilde{h}_\ell$	Reference and precision-policy activations
f_ℓ, J_ℓ	Local forward map and activation Jacobian
$g_\ell, \widetilde{g}_\ell$	Reference and precision-policy adjoints
$\mathcal{G}_\ell$	Activation-adjoint space at site ℓ
$\pi_\ell, \widetilde{B}_\ell$	Local precision policy and backward map
$v_k^{\mathrm{ref}}, v_k^{\pi}$	Reference and realized incoming adjoints
$d_k(v; \pi_k)$	Realized local backward error
$\Delta J_k^\top$	Fixed-forward backward-map perturbation
$\delta h_k, \delta g_L$	Forward-state and loss-gradient shifts
$\Delta B_k^{\mathrm{prec}}, E_k$	Precision-only and combined operator errors
$U_k, \mathcal{T}_k$	Upstream and readout propagation
$P_\ell, \Delta P_\ell$	Parameter Jacobian and readout perturbation
e_k	Local error $E_k v_k^{\mathrm{ref}}$ added to the arriving gradient
$t_{k,\mathrm{act}}^\theta, t_{k,\mathrm{par}}^\theta$	Propagated and direct-readout contributions
$t_k^\theta, t_{\mathrm{term}}^\theta$	Combined site and loss-gradient contributions
$R_{\geq 2}, R_{\geq 2}^\theta$	Higher-order activation and gradient terms
$g_\theta, \widetilde{g}_\theta$	Reference and precision-policy gradients
D_{g}, ϕ	Gradient bending and inter-gradient angle
$R_{\mathrm{norm}}, S_{\mathrm{norm}}$	Norm ratio and log-scale distortion
P_{desc}	First-order descent preservation
$E_{\mathrm{rel}}^{\mathrm{param}}, \rho_q^{\mathrm{param}}$	Global and group relative gradient error
$\kappa_k^{\parallel}, \kappa_k^{\perp}$	Parallel and orthogonal site criticality
$\Pi_{g_\theta^\perp}$	Projector orthogonal to g_θ
η, β	Step size and gradient-smoothness constant
Υ; u^{ref}, u	Coordinate-wise rule; reference and policy updates
$\varpi_i, \zeta_i, \lambda_0$	Update weight, gain, and zero-reference mass
D_{u}	Common-state AdamW update bending
$S, \mathcal{Q}, \mathcal{R}, B_{\mathrm{grp}}$	Low groups, candidates, protected set, and group budget
$c_q, c(\mathcal{R}), B$	Action cost, set cost, and cost budget
$D(\mathcal{R}), \tau$	Calibration bending of a mask and target level
$m_q(\mathcal{R}), C_q$	Marginal protection vector and cache cost
u_q, Γ_{ij}	Orthogonal group contribution and alignment
$\mathcal{E}_\perp(S), V(\mathcal{R})$	Bending energy and protection benefit
I_{ij}^{int}	Pairwise precision-mask interaction
y_m, y_q	Module and group outputs
$S_{\mathrm{Fisher}}, S_{\mathrm{Hess}}$	Fisher and finite-difference curvature scores
$\epsilon_{\mathrm{guard}}, \epsilon_{\mathrm{norm}}$	Numerical guard constants
e_k^{D}; L_k, Q_k, A_k	Dgrad residual; load, quantization error, and amplification
$s_{\mathrm{MX},b}$	MXFP4 block scale
$s_{\mathrm{global}}, \widehat{s}_b, s_{\mathrm{NV},b}^{\mathrm{eff}}$	NVFP4 global, block, and effective scales

The linear Dgrad map $G_Y \mapsto G_Y W^\top$ does not use the input activation X. Forward errors affect later Wgrad GEMMs through the saved activations used to compute weight gradients. ReLU masks and the derivatives used to backpropagate through softmax and normalization also depend on the forward activations. At the output, errors in the logits change the loss gradient that starts backpropagation.

A Two-Layer Example. Consider a linear layer, a ReLU, and a second linear layer, $H = XW_1$, $A = \mathrm{ReLU}(H)$, and $Y = AW_2$, with the ReLU mask $M \coloneqq \mathbf{1}[H > 0]$. Backpropagation computes $G_A = G_Y W_2^\top$ in the Dgrad GEMM of the second layer, $G_H = M \odot G_A$ at the ReLU, and $G_{W_1} = X^\top G_H$ in the Wgrad GEMM of the first layer. The three routes of Sec. 2 appear as three terms of the error in G_{W_1}. The Dgrad error δG_A of the second layer reaches G_{W_1} as $X^\top(M \odot \delta G_A)$,

carried upstream by the mask. The quantization error δX of the Wgrad operand enters directly as $\delta X^\top G_H$.

Quantizing X in the forward pass changes H by $\delta X W_1$, which can flip entries of M and then changes G_H by $\delta M \odot G_A$: The changed mask alters the backward operator, and each flipped entry moves gradient mass between coordinates, which rotates the gradient (Sec. 4). The general expansion below includes these three kinds of term for every site of a network.

Notation. We compare the implemented backward pass with an exact reference, following classical finite-precision error analysis (Wilkinson, 1959; Higham, 2002). Let a network be written as

$$h_{\ell+1} \coloneqq f_\ell(h_\ell), \qquad J_\ell \coloneqq \frac{\partial h_{\ell+1}}{\partial h_\ell},$$

where f_ℓ is the computation between two consecutive quantization sites, such as a module between two boundary quantizers or a single linear layer. Indices increase in the forward direction, so the loss side has larger indices, and the backward pass propagates gradients toward smaller indices. Throughout this section, $g_\ell \coloneqq \partial \mathcal{L}/\partial h_\ell$ denotes the activation gradient, or adjoint, at h_ℓ, and $\mathcal{G}_\ell$ denotes the space of such gradients. Full-precision backpropagation is

$$g_0 = J_0^\top J_1^\top \cdots J_{L-1}^\top g_L,$$

and for the linear layer of Sec. 2, $J_\ell^\top$ is the Dgrad map $G_Y \mapsto G_Y W^\top$.

Local Error at a Site. Under a precision policy, the implemented backward map of site ℓ is

$$\widetilde{B}_\ell(\cdot;\pi_\ell) : \mathcal{G}_{\ell+1} \longrightarrow \mathcal{G}_\ell, \qquad v \longmapsto \widetilde{B}_\ell(v;\pi_\ell),$$

where π_ℓ is the policy at site ℓ and $v \in \mathcal{G}_{\ell+1}$ is the gradient arriving from the loss side. For a linear layer under GEMM-operand quantization, $\widetilde{B}_\ell(v) = Q(v)Q(W)^\top$; for a boundary quantizer at the output of f_ℓ, $\widetilde{B}_\ell(v) = J_\ell^\top Q(v)$. The local backward error

$$d_\ell(v;\pi_\ell) \coloneqq \widetilde{B}_\ell(v;\pi_\ell) - J_\ell^\top v$$

evaluates the implemented and reference maps on the same arriving gradient, so d_ℓ isolates the error created at site ℓ from error already present in v; the word local identifies the boundary at which the error is measured before propagation. For the linear layer, $d_\ell(v)$ is the Dgrad error of Eq. 1. Dynamic scaling, rounding, and clipping can make $\widetilde{B}_\ell$ nonlinear in v.

Forward quantization adds a second source of local error. If upstream quantization changes the input of site k from h_k to $\widetilde{h}_k \coloneqq h_k + \delta h_k$, the backward pass evaluates J_k at the perturbed input $\widetilde{h}_k$, for example a ReLU mask with flipped units, the δM of the two-layer example. In the frozen or linearized operator view of Appendix B.4, the combined first-order perturbation of the backward operator of site k is

$$E_k \coloneqq \Delta B_k^{\mathrm{prec}}(\widetilde{h}_k) + [J_k(\widetilde{h}_k) - J_k(h_k)]^\top,$$

where $\Delta B_k^{\mathrm{prec}}(\widetilde{h}_k)$ is the operator perturbation from quantizing the backward operands, measured against $J_k(\widetilde{h}_k)^\top$ at the realized input, and the bracket is the Jacobian shift induced by forward quantization. Without forward quantization, E_k reduces to the fixed-forward operator perturbation $\Delta J_k^\top \coloneqq \widetilde{B}_k(\cdot;\pi_k) - J_k^\top$ of a zero-offset linear backward map, for which $d_k(v;\pi_k) = \Delta J_k^\top v$.

Propagation to the Input. For the full-precision trajectory, define the reference gradient arriving at site k and the upstream propagation operator by

$$v_k^{\mathrm{ref}} \coloneqq J_{k+1}^\top \cdots J_{L-1}^\top g_L = g_{k+1}, \qquad U_k \coloneqq J_0^\top \cdots J_{k-1}^\top = \left(\frac{\partial h_k}{\partial h_0}\right)^\top.$$

Empty products are the identity, so $U_0 = I$ and $v_{L-1}^{\mathrm{ref}} = g_L$. Read from right to left, v_k^{ref} arrives at site k, the local error $E_k v_k^{\mathrm{ref}}$ is created there, and U_k carries that error upstream to h_0. Expanding the implemented backward pass to first order in the site perturbations gives

$$\widetilde{g}_0 - g_0 = U_L \delta g_L + \sum_{k=0}^{L-1} \underbrace{U_k}_{\text{upstream propagation}} \underbrace{E_k v_k^{\mathrm{ref}}}_{\text{local error}} + R_{\geq 2}, \tag{2}$$

where $U_L \coloneqq J_0^\top \cdots J_{L-1}^\top$ carries the perturbation $\delta g_L \coloneqq \widetilde{g}_L - g_L$ of the loss gradient, for example from quantized logits, and $R_{\geq 2}$ collects the terms with two or more perturbation factors together with the remainders of the local linearizations. Dropping $R_{\geq 2}$ gives the first-order attribution across sites. Each singleton term shows that precision sensitivity depends jointly on the arriving direction, the local perturbation, and the upstream propagation; the propagation can amplify, attenuate, mix, or reorient the error.

With fixed forward state, $E_k = \Delta J_k^\top$ and $\delta g_L = 0$, Eq. 2 is the first-order part of the ordered product $(J_0^\top + \Delta J_0^\top) \cdots (J_{L-1}^\top + \Delta J_{L-1}^\top) g_L$; for a general, possibly nonlinear backward pass, the realized recursion then telescopes to an exact identity in which each site contributes the path-conditioned term $U_k d_k(v_k^\pi; \pi_k)$ with $v_k^\pi \coloneqq \widetilde{g}_{k+1}$ (Appendix B.4).

B.3 From Activation Gradients to the Parameter Gradient

The optimizer receives the gradient with respect to all trainable parameters. For a parameter block θ_ℓ used only in module ℓ, the parameter gradient is the Wgrad readout $\nabla_{\theta_\ell} \mathcal{L} = P_\ell^\top g_{\ell+1}$ with $P_\ell \coloneqq \partial h_{\ell+1} / \partial \theta_\ell$, which for the linear layer of Sec. 2 is $G_W = X^\top G_Y$. A precision policy perturbs this readout through the arriving gradient, $P_\ell^\top \delta g_{\ell+1}$, and through the readout operator, $\Delta P_\ell^\top g_{\ell+1}$; for the linear layer, the two terms are $X^\top \delta G_Y$ and $\delta X^\top G_Y$ in the Wgrad row of Eq. 1, and Appendix B.4 gives the first-order derivation and the case of shared parameters.

To connect a site to the optimizer-gradient vector, let $\mathcal{T}_k$ denote the reference readout propagation: $\mathcal{T}_k$ injects a gradient perturbation at h_k, carries the perturbation to every upstream parameter use, applies the corresponding $P_\ell^\top$, and concatenates the resulting parameter-gradient blocks. Let $e_k \coloneqq E_k v_k^{\mathrm{ref}}$ denote the local error that site k adds to the arriving gradient. The propagated part of the site's parameter-gradient contribution is $t_{k,\mathrm{act}}^\theta \coloneqq \mathcal{T}_k e_k = \mathcal{T}_k E_k v_k^{\mathrm{ref}}$.

Under a single-site first-order attribution, let $t_{k,\mathrm{par}}^\theta$ collect the direct $\Delta P_\ell^\top g_{\ell+1}$ terms attributed to site k, whether the terms arise from quantized Wgrad operands or from forward-induced parameter-Jacobian shifts, and define the combined contribution $t_k^\theta \coloneqq t_{k,\mathrm{act}}^\theta + t_{k,\mathrm{par}}^\theta$. The concatenated optimizer-gradient shift therefore has the form

$$\widetilde{g}_\theta - g_\theta = t_{\mathrm{term}}^\theta + \sum_{k=0}^{L-1} t_k^\theta + R_{\geq 2}^\theta, \tag{3}$$

where t_{term}^θ is the parameter-gradient readout of the loss-gradient perturbation δg_L and $R_{\geq 2}^\theta$ collects every remaining term: multi-site products, the bilinear interactions $\Delta P_\ell^\top \delta g_{\ell+1}$, and the higher-order remainders of the local linearizations. Dropping $R_{\geq 2}^\theta$ gives the first-order attribution used below.

This expression exposes a directional criticality factorization. Whenever $\|g_\theta\|_2$, $\|v_k^{\mathrm{ref}}\|_2$, and $\|E_k v_k^{\mathrm{ref}}\|_2$ are nonzero,

$$\frac{\|t_{k,\mathrm{act}}^\theta\|_2}{\|g_\theta\|_2} = \underbrace{\frac{\|v_k^{\mathrm{ref}}\|_2}{\|g_\theta\|_2}}_{\text{incoming load}} \underbrace{\frac{\|E_k v_k^{\mathrm{ref}}\|_2}{\|v_k^{\mathrm{ref}}\|_2}}_{\text{directional local error}} \underbrace{\frac{\|\mathcal{T}_k E_k v_k^{\mathrm{ref}}\|_2}{\|E_k v_k^{\mathrm{ref}}\|_2}}_{\text{propagation and readout gain}} . \tag{4}$$

The incoming load is the size of the gradient arriving at the site relative to the parameter gradient, $\|G_Y\|_2 / \|g_\theta\|_2$ for a linear layer. The directional local error is the relative size of the error that the site adds to that gradient; when the error comes from quantizing the arriving gradient with per-tensor scaling, this factor is nearly the same at every site, between 0.022 and 0.027 across ResNet-18 (Appendix D.3). The propagation and readout gain measures how strongly the layers between the site and the Wgrad readouts amplify the error.

On ResNet-18 with backward-only quantizers, the incoming load falls about 70-fold from the stem to the last residual block while the propagation-and-readout gain rises about 250-fold, so no single factor orders the sites (Figure 5(a) in Appendix D.3).

For the activation-side mechanism, a site is optimizer-critical when a loaded incoming direction produces a substantial directional local error and the network propagates that error strongly into parameter space. A large value of $\|E_k v_k^{\mathrm{ref}}\|_2 / \|v_k^{\mathrm{ref}}\|_2$ establishes only the middle factor; a forward-output reconstruction error is an even more indirect proxy and need not determine that factor. The

factorization in Eq. 4 covers only the propagated term $t^{\theta}_{k,\mathrm{act}}$; Appendix D.3 notes how the direct readout contribution $t^{\theta}_{k,\mathrm{par}}$ enters.

B.4 Realized-Pass Identity, Shared Parameters, and Terminal and Forward-Induced Perturbations

Exact Realized-Pass Identity. Appendix B.2 expands the frozen or linearized operator product to first order. For a general, possibly nonlinear backward pass under the same fixed-forward-state setup, the realized site error is $d_k(v_k^{\pi};\pi_k)$, and $U_k d_k(v_k^{\pi};\pi_k)$ is the corresponding propagated site contribution. Because $U_{k+1} = U_k J_k^{\top}$, the realized recursion $\widetilde{g}_k = J_k^{\top} v_k^{\pi} + d_k(v_k^{\pi};\pi_k)$ telescopes. When $\widetilde{g}_L = g_L$, the telescoped recursion gives the exact identity

$$\widetilde{g}_0 - g_0 = \sum_{k=0}^{L-1} U_k d_k(v_k^{\pi};\pi_k). \tag{5}$$

The propagated site contributions in Eq. 5 are exact path-conditioned terms: Dependence on the realized incoming adjoints incorporates the downstream interactions represented by $R_{\geq 2}$ in Eq. 2.

Readout Perturbation. Appendix B.3 writes the parameter gradient of an unshared block θ_ℓ as the Wgrad readout $\nabla_{\theta_\ell}\mathcal{L} = P_\ell^{\top} g_{\ell+1}$ with $P_\ell \coloneqq \partial h_{\ell+1}/\partial\theta_\ell$. In a frozen or linearized readout view, define $\Delta P_\ell \coloneqq \widetilde{P}_\ell - P_\ell$. The term $\Delta P_\ell^{\top} g_{\ell+1}$ denotes the policy-induced first-order perturbation of the readout; the term includes both direct finite-precision error in a Wgrad computation at fixed operands and changes of the exact parameter Jacobian induced by the perturbed forward state. Define also $\delta g_{\ell+1} \coloneqq \widetilde{g}_{\ell+1} - g_{\ell+1}$ and the precision-policy gradient shift by $\delta(\nabla_{\theta_\ell}\mathcal{L}) \coloneqq \widetilde{P}_\ell^{\top}\widetilde{g}_{\ell+1} - P_\ell^{\top} g_{\ell+1}$. Expanding this difference and omitting the bilinear interaction $\Delta P_\ell^{\top}\delta g_{\ell+1}$, which is second order when both shifts are first order, gives

$$\delta(\nabla_{\theta_\ell}\mathcal{L}) \approx P_\ell^{\top}\delta g_{\ell+1} + \Delta P_\ell^{\top} g_{\ell+1};$$

for the linear layer, the two terms are $X^{\top}\delta G_Y$ and $\delta X^{\top} G_Y$ in the Wgrad row of Eq. 1.

Parameter-Gradient Readout Propagation. In a chain with unshared parameters,

$$[\mathcal{T}_k e]_{\theta_\ell} \coloneqq \begin{cases} P_\ell^{\top} J_{\ell+1}^{\top}\cdots J_{k-1}^{\top} e, & \ell < k, \\ 0, & \ell \geq k, \end{cases}$$

with the middle product equal to the identity when $\ell = k-1$. For shared-parameter blocks, the readout sums across all parameter uses, and the same definition extends to a directed acyclic graph by summing over reference backward paths; the following paragraph states the shared-block parameter gradient and the first-order perturbation of that gradient explicitly.

Shared Parameter Blocks. With P_ℓ, ΔP_ℓ, and $\delta g_{\ell+1}$ as above, if a parameter block θ is shared across modules $\mathcal{I}(\theta)$, the parameter gradient and the corresponding perturbation are given, respectively, by

$$\nabla_\theta\mathcal{L} = \sum_{\ell\in\mathcal{I}(\theta)} P_\ell^{\top} g_{\ell+1}, \qquad \delta(\nabla_\theta\mathcal{L}) \approx \sum_{\ell\in\mathcal{I}(\theta)} (P_\ell^{\top}\delta g_{\ell+1} + \Delta P_\ell^{\top} g_{\ell+1}).$$

Terminal and Forward-Induced Perturbations. The expansion in Eq. 2 includes the loss-gradient perturbation δg_L and the combined perturbation E_k; this paragraph derives both from the fixed-forward specialization. First-order activation-gradient distortion has three conceptually distinct sources: perturbation of the loss gradient, perturbation of the implemented local backward map at a fixed activation, and shifts of the exact local Jacobians induced by perturbed forward states. The three sources can interact: The backward-policy perturbation may depend directly on the perturbed activation, and forward perturbations can also shift parameter Jacobians, as described in the readout perturbation above.

If the loss gradient under the precision policy is $\widetilde{g}_L$, possibly because the loss-gradient computation is performed in low precision or evaluated at a perturbed terminal activation, define $\delta g_L \coloneqq \widetilde{g}_L - g_L$.

Equivalently, $\widetilde{g}_L = g_L + \delta g_L$. After propagation through the reference Jacobian chain, the loss-gradient perturbation contributes the additional first-order term $J_0^\top \cdots J_{L-1}^\top \delta g_L$ to $\widetilde{g}_0 - g_0$.

Writing $J_k(x) \coloneqq \mathrm{D}f_k(x)$, if forward quantization changes the realized input to site k from h_k to $\widetilde{h}_k \coloneqq h_k + \delta h_k$, then the exact local Jacobian is evaluated at the perturbed input $\widetilde{h}_k$. The forward-state perturbation can induce the shift $[J_k(\widetilde{h}_k) - J_k(h_k)]^\top$, distinct from finite-precision perturbation of the backward map at a fixed activation.

In the frozen or linearized operator view, we write $\Delta B_k^{\mathrm{prec}}(x)$ for the precision-policy perturbation of the local backward operator, measured against $J_k(x)^\top$ at the same realized activation x and on the same incoming adjoint. The precision-policy perturbation includes effects of the specified scaling, rounding, and clipping rules, but excludes the change in the exact Jacobian caused by perturbing the forward activation. More explicitly, if δh_k is the activation perturbation, then the combined local backward-operator perturbation relative to $J_k(h_k)^\top$ is

$$\Delta B_k \coloneqq \Delta B_k^{\mathrm{prec}}(h_k + \delta h_k) + [J_k(h_k + \delta h_k) - J_k(h_k)]^\top.$$

The first term includes the entire precision-policy perturbation evaluated at the realized activation and can therefore depend directly on δh_k; the second term isolates the change in the exact Jacobian caused by the forward perturbation.

In the product-expansion notation, this combined perturbation ΔB_k plays the role of E_k. When forward activation perturbations are absent, the fixed-activation precision-policy component of ΔB_k corresponds to the earlier $\Delta J_k^\top$ term. Thus, operator-level first-order expressions use E_k, represented here by ΔB_k, in place of $\Delta J_k^\top$ when forward-induced shifts are included. Our experiments explicitly separate forward-only, backward-only, and joint forward-and-backward simulated-precision interventions.

The exact combined perturbation ΔB_k defined above admits the following first-order expansion of the Jacobian-shift component, while the activation dependence of the precision-policy perturbation remains explicit.

Denoting the Fréchet derivative by D, we write $\mathrm{D}J_k(h_k)[\delta h_k]$ for the first-order variation of the Jacobian map in the direction δh_k. If J_k is differentiable and $\mathrm{D}J_k$ is locally Lipschitz near h_k, as in the case where J_k is C^2 with a locally bounded second derivative, then

$$\begin{aligned}\Delta B_k = \Delta B_k^{\mathrm{prec}}(h_k) + (\mathrm{D}J_k(h_k)[\delta h_k])^\top \\ + [\Delta B_k^{\mathrm{prec}}(h_k + \delta h_k) - \Delta B_k^{\mathrm{prec}}(h_k)] + O(\|\delta h_k\|^2).\end{aligned}$$

The first two terms are the fixed-activation precision-policy perturbation in the backward operator and the forward-induced shift. The $O(\|\delta h_k\|^2)$ term is the transposed Taylor remainder for $J_k(h_k + \delta h_k) - J_k(h_k)$. We keep the bracketed term explicit to show how the precision-policy perturbation changes with the activation. The bracketed term can be absorbed into a second-order remainder only when additional regularity is imposed on that dependence.

The decomposition above is justified as follows. By the definition of the combined perturbation above,

$$\Delta B_k = \Delta B_k^{\mathrm{prec}}(h_k + \delta h_k) + [J_k(h_k + \delta h_k) - J_k(h_k)]^\top.$$

Adding and subtracting $\Delta B_k^{\mathrm{prec}}(h_k)$ reproduces the first and bracketed terms above exactly, so the only remaining task is to expand the transposed Jacobian difference to first order.

The stated regularity provides a convex neighborhood N of h_k on which J_k is differentiable and $\mathrm{D}J_k$ is Lipschitz with a constant Λ_k; in particular, $\mathrm{D}J_k$ is continuous on N, so J_k is continuously differentiable there. For δh_k small enough that the segment from h_k to $h_k + \delta h_k$ lies in N, the fundamental theorem of calculus along that segment gives

$$J_k(h_k + \delta h_k) - J_k(h_k) - \mathrm{D}J_k(h_k)[\delta h_k] = \int_0^1 (\mathrm{D}J_k(h_k + s\delta h_k) - \mathrm{D}J_k(h_k))[\delta h_k]\mathrm{d}s.$$

The Lipschitz bound $\|\mathrm{D}J_k(h_k + s\delta h_k) - \mathrm{D}J_k(h_k)\| \le \Lambda_k s\|\delta h_k\|$ bounds the integrand in norm by $\Lambda_k s\|\delta h_k\|^2$ and hence the integral by $(1/2)\Lambda_k\|\delta h_k\|^2$. Transposition preserves the operator norm, so the transposed remainder obeys the same bound, which is exactly the stated $O(\|\delta h_k\|^2)$ estimate.

B.5 Mechanism Taxonomy

The following mechanisms specify different structures of the frozen local operator perturbation $\Delta J_k^\top$ or the combined product-expansion perturbation E_k when forward-induced shifts are present. In the realized-error view, the corresponding object is $d_k(v;\pi_k)$. The factorization in Eq. 4 gives the shared test across these cases: Determine the loaded directional error $E_k v_k^{\mathrm{ref}}$, the propagation and readout gain of the directional error, and the parallel and orthogonal components of the resulting t_k^θ. The mechanisms below predict which of these factors becomes large or discontinuous.

B.5.1 Residual Update Evaporation

Residual networks carry each branch contribution alongside an identity path (He et al., 2016). For $h_{\ell+1} \coloneqq h_\ell + \alpha F_\ell(h_\ell)$, where J_{F_ℓ} denotes the Jacobian of the residual branch F_ℓ, α denotes the branch scale, and I denotes the identity,

$$g_\ell = (I + \alpha J_{F_\ell})^\top g_{\ell+1}.$$

Let Q_b denote the quantizer at the relevant backward boundary. If

$$Q_b(g_{\ell+1} + \alpha J_{F_\ell}^\top g_{\ell+1}) = Q_b(g_{\ell+1}),$$

then that boundary contains no resolvable residual-branch increment. A residual contribution small relative to the local representable spacing makes this event more likely, but is not sufficient alone because quantization-bin boundaries matter.

B.5.2 Normalization-Induced Coherent Distortion

Normalization layers are standard cases where shared statistics enter the backward operator coherently across coordinates, from batch normalization in residual convolutional neural networks to layer normalization (LN) in transformers (Ioffe & Szegedy, 2015; Ba et al., 2016; Xiong et al., 2020). For the non-affine root-mean-square (RMS) normalization core (Zhang & Sennrich, 2019), with $r(x) \coloneqq (\|x\|_2^2/d + \epsilon_{\mathrm{norm}})^{1/2}$,

$$J_{\mathrm{RMS}}(x) \coloneqq \frac{1}{r(x)} I - \frac{1}{d r(x)^3} x x^\top.$$

For the non-affine LN core, with $P_c \coloneqq I - \mathbf{1}_d \mathbf{1}_d^\top / d$, $x_c \coloneqq P_c x$, and $r_{\mathrm{LN}}(x) \coloneqq (\|x_c\|_2^2/d + \epsilon_{\mathrm{norm}})^{1/2}$,

$$J_{\mathrm{LN}}(x) \coloneqq \frac{1}{r_{\mathrm{LN}}(x)} P_c - \frac{1}{d r_{\mathrm{LN}}(x)^3} x_c x_c^\top.$$

If an affine scale γ is included, the forward Jacobian is $\mathrm{diag}(\gamma) J_{\mathrm{core}}(x)$, and the corresponding exact local backward operator is $J_{\mathrm{core}}(x)^\top \mathrm{diag}(\gamma)$. Low precision perturbs shared scalar invariants and rank-one projection terms, so the error is coherent across coordinates.

B.5.3 Attention-Weight Redistribution

For the row-wise attention map in a transformer attention block (Vaswani et al., 2017), define

$$Z \coloneqq \frac{Q_{\mathrm{att}} K_{\mathrm{att}}^\top}{\sqrt{d_{\mathrm{qk}}}}, \qquad A \coloneqq \mathrm{softmax}_{\mathrm{row}}(Z).$$

Here, d_{qk} is the query-key dimension. Let z_i be row i of Z, written as a column vector, and let $p_i \coloneqq \mathrm{softmax}(z_i)$, so that $p_i^\top = A_{i,:}$. The softmax Jacobian at z_i, which relates query i's logits to the corresponding row of the attention map, is

$$J_{\mathrm{softmax}}(z_i) \coloneqq \frac{\partial \,\mathrm{softmax}(z_i)}{\partial z_i} = \mathrm{diag}(p_i) - p_i p_i^\top,$$
$$\delta p_i \approx J_{\mathrm{softmax}}(z_i) \delta z_i.$$

Thus, a low-precision perturbation of Q_{att} or K_{att} produces a logit perturbation δz_i, and $J_{\mathrm{softmax}}(z_i)$ maps the logit perturbation to a first-order change δp_i in the attention weights. This change perturbs the attention block's forward and backward maps and can therefore contribute to the forward-induced Jacobian shift in the propagated-error decomposition.

A small gap between the largest attention logits can allow a perturbation δz_i to swap which key has the largest attention weight. Standard dense softmax attention nevertheless remains continuous in the logits, so an argmax swap alone does not imply a large change in the weight vector; a genuinely discrete rerouting effect requires an additional hard or top-k routing operation.

B.5.4 Gated Feed-Forward Multiplicative Error

Gated feed-forward variants, including gated linear units in transformer MLP blocks, use a componentwise product between the gate and value branches (Shazeer, 2020). For $y \coloneqq \phi(a) \odot b$,

$$g_a = \phi'(a) \odot b \odot g_y, \qquad g_b = \phi(a) \odot g_y.$$

Each branch enters multiplicatively in the other branch's local derivative, so perturbing either branch changes the corresponding backward coefficient; simultaneous branch perturbations also create higher-order cross terms. Consequently, branch-wise local reconstruction error can underestimate propagated gradient error.

B.5.5 State-Space and Recurrent Drift

Selective state-space and recurrent sequence models expose long products of transition Jacobians along the sequence dimension, the classical source of vanishing and exploding gradients in recurrent training (Pascanu et al., 2013; Gu & Dao, 2023). For recurrent backward dynamics, let $F_t \coloneqq \partial h_{t+1}/\partial h_t$ be the forward Jacobian. Then

$$\delta g_t \approx F_t^\top \delta g_{t+1} + (\Delta F_t)^\top g_{t+1}.$$

Over a segment $[a, b]$, writing superscripts full and low for the full- and low-precision recurrent products, a useful summary is

$$\Delta_{\log}(a, b, u) \coloneqq \log \frac{\|\Phi_{a:b}^{\mathrm{low}} u\|_2 + \epsilon_{\mathrm{guard}}}{\|\Phi_{a:b}^{\mathrm{full}} u\|_2 + \epsilon_{\mathrm{guard}}}.$$

Here, $\Phi_{a:b}^{\mathrm{full}} \coloneqq F_a^\top F_{a+1}^\top \cdots F_b^\top$ and $\Phi_{a:b}^{\mathrm{low}}$ is the analogous low-precision product. Thus, $\Delta_{\log}(a, b, u)$ measures the logarithmic change in the backward gain in direction u when the precision of the recurrent Jacobian chain over $[a, b]$ changes from full to low precision. Positive values indicate extra amplification, whereas negative values indicate attenuation.

B.6 Derived Diagnostics, Metric Identities, and One-Step Bound Details

This subsection collects the derived gradient diagnostics, the two-coordinate geometry of a singleton propagated contribution, and the one-step bound details that supplement Sec. 4. Descent preservation measures how much of the reference gradient step's predicted loss decrease the low-precision step retains. Descent preservation and total relative error are derived from the two coordinates, as shown below. Precision-mask interaction is a separate set-level diagnostic, defined in Appendix D.4.

Numerical Guards. The cosine-distance definition of D_{g} requires both gradient norms to be nonzero. The implementation uses a small denominator guard on degenerate batches, as shown in Listing 1.

Derived Diagnostics. Two derived summaries appear in the analysis and tables. First-order descent preservation $P_{\mathrm{desc}} \coloneqq \langle g_\theta, \widetilde{g}_\theta \rangle / \|g_\theta\|_2^2$ is the candidate step's projection onto the full-precision negative-gradient direction $-g_\theta$; P_{desc} is positive exactly when $-\widetilde{g}_\theta$ remains a first-order descent direction for the reference loss. The composite relative parameter-gradient error is $E_{\mathrm{rel}}^{\mathrm{param}} \coloneqq \|\widetilde{g}_\theta - g_\theta\|_2 / (\|g_\theta\|_2 + \epsilon_{\mathrm{guard}})$. Neither adds an independent degree of freedom: Away from degenerate zero-gradient cases and ignoring numerical guards, D_{g} and R_{norm} determine both through exact identities, including $P_{\mathrm{desc}} = R_{\mathrm{norm}}(1 - D_{\mathrm{g}})$. This subsection gives the full definitions and derivations, resolves a singleton propagated contribution into components parallel and orthogonal to g_θ (Proposition 1), and proves the one-step descent bound and the associated step-size corollary.

First-Order Descent Preservation. Descent preservation quantifies the candidate step's projection onto the full-precision negative-gradient direction $-g_\theta$ and thus indicates whether low precision preserves the intended first-order loss decrease.

$$P_{\mathrm{desc}} \coloneqq \frac{\langle g_\theta, \widetilde{g}_\theta \rangle}{\|g_\theta\|_2^2}.$$

Here $P_{\mathrm{desc}} > 0$ means that $-\widetilde{g}_\theta$ remains a descent direction for the reference loss, $P_{\mathrm{desc}} < 0$ gives first-order ascent, and $P_{\mathrm{desc}} = 0$ gives no first-order loss change. The first-order expansion in Eq. 6 below provides this interpretation. In this context, descent denotes the local loss-decreasing property of a proposed update direction, a criterion that applies across optimization algorithms. This diagnostic tests the raw mixed-precision gradient step $-\eta\widetilde{g}_\theta$; for an optimizer such as AdamW that transforms the raw gradient, the same directional test can be applied to the realized update vector. If $\mathcal{L}$ is differentiable at θ with full-precision parameter gradient g_θ, $\|g_\theta\|_2 > 0$, and $\widetilde{g}_\theta$ is held fixed as $\eta \to 0$, then differentiability gives, for the SGD step $\theta^+ \coloneqq \theta - \eta\widetilde{g}_\theta$,

$$\mathcal{L}(\theta^+) = \mathcal{L}(\theta) - \eta\|g_\theta\|_2^2 P_{\mathrm{desc}} + o(\eta). \tag{6}$$

When $P_{\mathrm{desc}} = 1$, the candidate step matches the reference negative-gradient step's first-order loss decrease, although $\widetilde{g}_\theta$ may still contain an orthogonal error component. Large deviations from one indicate scale or projection mismatch.

Composite Relative Parameter-Gradient Error. The global relative parameter-gradient error reported in tables is

$$E_{\mathrm{rel}}^{\mathrm{param}} \coloneqq \frac{\|\widetilde{g}_\theta - g_\theta\|_2}{\|g_\theta\|_2 + \epsilon_{\mathrm{guard}}}.$$

Group-Wise Relative Error. For any explicitly specified reporting group q of parameter coordinates, let G_q denote the corresponding concatenated parameter-gradient block; such a reporting partition need not coincide with the groups of quantization sites used for protection. The group-wise counterpart of $E_{\mathrm{rel}}^{\mathrm{param}}$ is

$$\rho_q^{\mathrm{param}} \coloneqq \frac{\|\widetilde{G}_q - G_q\|_2}{\|G_q\|_2 + \epsilon_{\mathrm{guard}}}.$$

Metric Identities. Away from degenerate zero-gradient cases and ignoring numerical guards, the five global quantities D_{g}, P_{desc}, R_{norm}, S_{norm}, and $E_{\mathrm{rel}}^{\mathrm{param}}$ obey the exact identities, derived in Appendix B.10,

$$P_{\mathrm{desc}} = R_{\mathrm{norm}}(1 - D_{\mathrm{g}}), \tag{7}$$

$$E_{\mathrm{rel}}^{\mathrm{param}} = \sqrt{(R_{\mathrm{norm}} - 1)^2 + 2R_{\mathrm{norm}}D_{\mathrm{g}}}, \tag{8}$$

$$S_{\mathrm{norm}} = |\log R_{\mathrm{norm}}|. \tag{9}$$

Consequently, D_{g} and R_{norm} determine the other global gradient diagnostics. The identities hold for each single pair $(g_\theta, \widetilde{g}_\theta)$; because the identities are nonlinear, separately averaged diagnostics over several calibration batches need not satisfy the same identities exactly, and the discrepancy depends on batch-to-batch variation. Among the derived quantities, P_{desc} has a direct loss-decrease interpretation, $E_{\mathrm{rel}}^{\mathrm{param}}$ provides a composite severity score, and S_{norm} provides a reciprocal-symmetric magnitude-distortion score by assigning the same value to norm ratios r and $1/r$.

Roles of the Two Coordinates. For the examples in Sec. 4, the doubled gradient has $R_{\mathrm{norm}} = 2$ and $D_{\mathrm{g}} = 0$, whereas the rotated gradient has $R_{\mathrm{norm}} = 1$ and $D_{\mathrm{g}} = 0.5$. With ϕ the angle between the two gradients, so that $\cos\phi = 1 - D_{\mathrm{g}}$, the metric identities above split the squared relative error as

$$\frac{\|\widetilde{g}_\theta - g_\theta\|_2^2}{\|g_\theta\|_2^2} = \underbrace{(R_{\mathrm{norm}} - \cos\phi)^2}_{\text{removed by rescaling}} + \underbrace{D_{\mathrm{g}}(2 - D_{\mathrm{g}})}_{\text{irreducible}}, \tag{10}$$

and scaling $\widetilde{g}_\theta$ by $\cos\phi/R_{\mathrm{norm}}$, the best rescaling whenever $D_{\mathrm{g}} < 1$, removes exactly the first term. Learning rate adjustment therefore absorbs changes in gradient norm but never bending; Corollary 1 gives the same split for the guaranteed loss decrease, of which the best retuned step achieves the fraction $(1 - D_{\mathrm{g}})^2$. For all-low FP8 boundary quantization, rescaling could remove less than 0.1% of the squared gradient error on ResNet-18 and 31% on GPT-2, and the 4-bit all-low GPT-2 gradient has 2.1 times the reference norm but, even at the best step size, still has a relative error of 0.84 from the 57° gradient rotation (Tables 3 and 6).

The first-order expansion of P_{desc} above is asymptotic in the step size. The next lemma gives a nonasymptotic one-step descent bound in terms of D_{g} and R_{norm}, and the following step-size corollary separates the roles of the two coordinates in the smoothness guarantee.

Lemma 1 (One-step two-coordinate descent)**.** *Let $\mathcal{L}$ be differentiable with β-Lipschitz gradient for some $\beta > 0$, let θ satisfy $g_\theta = \nabla\mathcal{L}(\theta) \neq 0$, let $\widetilde{g}_\theta \neq 0$, and let D_{g}, P_{desc}, and R_{norm} be the guard-free diagnostics of the pair $(g_\theta, \widetilde{g}_\theta)$. Then, for every $\eta > 0$,*

$$\mathcal{L}(\theta - \eta\widetilde{g}_\theta) \leq \mathcal{L}(\theta) - \eta\|g_\theta\|_2^2\Big[R_{\mathrm{norm}}(1 - D_{\mathrm{g}}) - \tfrac{\beta\eta}{2}R_{\mathrm{norm}}^2\Big], \tag{11}$$

where the bracket equals $P_{\mathrm{desc}} - (\beta\eta/2)R_{\mathrm{norm}}^2$.

Optimizing the one-step bound of Lemma 1 over the step size separates the roles of the two coordinates.

Corollary 1 (Step-size retuning and the irreducible bending loss)**.** *Let $\mathcal{L}$ have β-Lipschitz gradient for some $\beta > 0$. For a step at θ_t with $g_\theta^{(t)} \neq 0$, $\widetilde{g}_\theta^{(t)} \neq 0$, and $D_{\mathrm{g}}^{(t)} < 1$, the right-hand side of Eq. 11 is smallest at $\eta_t^\star = (1 - D_{\mathrm{g}}^{(t)})/(\beta R_{\mathrm{norm}}^{(t)})$, where the bound equals $\mathcal{L}(\theta_t) - (1 - D_{\mathrm{g}}^{(t)})^2\|g_\theta^{(t)}\|_2^2/(2\beta)$. For the exact gradient, the corresponding bound is minimized at $\eta = 1/\beta$ and equals $\mathcal{L}(\theta_t) - \|g_\theta^{(t)}\|_2^2/(2\beta)$.*

Thus, within this one-step smoothness guarantee, step-size retuning removes the effect of $R_{\mathrm{norm}}^{(t)}$ entirely, while angular bending multiplies the best guaranteed decrease by $(1 - D_{\mathrm{g}}^{(t)})^2 = \cos^2\phi_t$, where ϕ_t is the angle between $g_\theta^{(t)}$ and $\widetilde{g}_\theta^{(t)}$. By the sharpness example below, this factor is attained and cannot be improved without additional assumptions on the loss.

The two-coordinate bound provides the analytical form of the division of labor used throughout the experiments: D_{g} is the quantity that protection must repair, whereas R_{norm} records the step-size change that a retuned learning rate, AdamW's normalization, or gradient clipping absorbs (Eq. 10). The plug-in illustration below applies the bound to the batch-averaged decoder-LM coordinates and notes the additional conditions required for per-step analysis in stochastic mini-batch training.

Singleton Contributions in the Two Coordinates. The factorization in Eq. 4 concerns the magnitude of a propagated contribution. The next result resolves the same contribution into the two coordinates of the optimizer gradient. For $\|g_\theta\|_2 > 0$, the normalized parallel and orthogonal components of a singleton contribution t_k^θ are

$$\kappa_k^{\parallel} \coloneqq \frac{\langle g_\theta, t_k^\theta\rangle}{\|g_\theta\|_2^2}, \qquad\qquad \kappa_k^{\perp} \coloneqq \frac{\|\Pi_{g_\theta^\perp} t_k^\theta\|_2}{\|g_\theta\|_2},$$

where $\Pi_{g_\theta^\perp} \coloneqq I - g_\theta g_\theta^\top/\|g_\theta\|_2^2$ is the orthogonal projector onto the complement of g_θ.

Proposition 1 (Site criticality and optimizer-gradient geometry)**.** *Let t_k^θ be the combined singleton contribution from Eq. 3, including any first-order parameter-gradient-readout perturbation, and suppose $\widetilde{g}_\theta = g_\theta + t_k^\theta$ with $\|g_\theta\|_2 > 0$ and $\|\widetilde{g}_\theta\|_2 > 0$. Use the guard-free forms of all diagnostics in this proposition. Then*

$$P_{\mathrm{desc}} = 1 + \kappa_k^{\parallel}, \qquad R_{\mathrm{norm}} = \sqrt{(1 + \kappa_k^{\parallel})^2 + (\kappa_k^{\perp})^2},$$
$$D_{\mathrm{g}} = 1 - \frac{1 + \kappa_k^{\parallel}}{\sqrt{(1 + \kappa_k^{\parallel})^2 + (\kappa_k^{\perp})^2}}, \qquad E_{\mathrm{rel}}^{\mathrm{param}} = \sqrt{(\kappa_k^{\parallel})^2 + (\kappa_k^{\perp})^2}.$$

In particular, the reference descent direction is preserved to first order exactly when $\kappa_k^{\parallel} > -1$; in this descent-preserving regime, gradient bending is controlled by the orthogonal component relative to the positive quantity $1 + \kappa_k^{\parallel}$.

Small-Perturbation Expansion of D_{g}. For a small singleton perturbation in the descent-preserving regime, if $1 + \kappa_k^{\parallel}$ stays bounded away from zero as $\kappa_k^{\perp} \to 0$, then

$$D_{\mathrm{g}} = \frac{(\kappa_k^{\perp})^2}{2(1+\kappa_k^{\parallel})^2} + o((\kappa_k^{\perp})^2).$$

Appendix B.10 derives this expansion with an explicit remainder bound.

Reading of the Two Coordinates. Proposition 1 makes all four diagnostics closed-form functions of the normalized parallel and orthogonal components $(\kappa_k^{\parallel}, \kappa_k^{\perp})$. The reference descent direction is preserved to first order exactly when $\kappa_k^{\parallel} > -1$, and in this regime D_{g} is second order in the orthogonal component, as the expansion above states and Appendix B.10 derives with an explicit remainder bound. Angular distortion therefore isolates bending but is deliberately insensitive to a purely parallel gain error that leaves a positive gain. This distinction motivates using D_{g} for direct reduction of gradient bending in Sec. 5 and R_{norm}, or the reciprocal-symmetric transform S_{norm}, to certify the correctable scale error and as a secondary ordering signal.

Contributions to the Norm Ratio. The norm ratio reflects both components of the error: $R_{\mathrm{norm}} = ((1+\kappa^{\parallel})^2+(\kappa^{\perp})^2)^{1/2}$. The parallel component rescales the gradient, while the orthogonal component can simultaneously increase the gradient norm and rotate the gradient. The parallel component is small at most sites and largest at the GPT-2 logits, where $\kappa^{\parallel} = 0.095$ and $\kappa^{\perp} = 0.61$. Similarly, the all-low NVFP4-style gradient has 2.1 times the reference norm, although the NVFP4-style gradient's projection onto g_θ is only 1.13 times the reference.

Norm distortion is an inexpensive diagnostic that also ranks the trained GPT-2 boundary masks well, with Spearman correlation 0.93 against final loss (Table 12). However, a norm ratio near one does not imply small bending: The MXFP4 core in Appendix E.1 produces a gradient with a norm 2% smaller than the reference but rotated by 39°. Measuring both coordinates distinguishes correctable scale changes from the directional error targeted by precision allocation.

Sharpness and Plug-In Illustration of the One-Step Bound. The bound is sharp: For $\mathcal{L}(\theta) \coloneqq (\beta/2)\|\theta - \theta_*\|_2^2$, Eq. 11 holds with equality.

At a step size tuned to the reference, $\beta\eta = 1$, the bracket in Eq. 11 is $R_{\mathrm{norm}}(1 - D_{\mathrm{g}}) - R_{\mathrm{norm}}^2/2$, against $1/2$ for the exact gradient. Plugging the batch-averaged coordinates in Table 6 into this expression gives descriptive ratios of 77% for the all-low FP8 policy, 99.8% for FP8 bending-guided precision allocation, and 98.5% for NVFP4 bending-guided precision allocation. The all-low NVFP4 means, $R_{\mathrm{norm}} = 2.10$ and $D_{\mathrm{g}} = 0.45$, give a negative plug-in bracket, while the optimally retuned one-step plug-in bound based on the mean D_{g} is $(1 - 0.45)^2 \approx 30\%$ of the reference bound. Because the table averages nonlinear diagnostics across batches, these percentages are descriptive illustrations of the one-step bound.

Per-step or trajectory analysis in stochastic mini-batch training additionally requires conditional alignment or bias control and a conditional second-moment bound for the sampled precision-policy direction.

B.7 Update Geometry under Coordinate-Wise Optimizers

Coordinate-Wise Optimizers. The main experiments train with AdamW. This subsection carries the two coordinates to coordinate-wise update rules such as the fixed-state AdamW update (Theorem 1 and Corollaries 2 and 3) and defines the update-level bending diagnostic D_{u}, which Appendix E measures alongside D_{g}.

This subsection extends the analysis of Sec. 4 to coordinate-wise optimizers. The main experiments train with AdamW, and Appendix E measures the bending of a common-state AdamW update

alongside the bending of the raw gradient. Holding the pre-update moments, parameters, step index, and hyperparameters fixed makes each update coordinate a function of the corresponding supplied-gradient coordinate.

Theorem 1 expresses the diagnostics of the reference and precision-policy update pair through the first two moments of the induced coordinate-wise gains and the update mass λ_0 on coordinates where the reference update vanishes. When $\lambda_0 = 0$, update bending is a strictly increasing function of the weighted coefficient of variation (CV) of these gains (Corollary 2). For a fixed diagonal preconditioner, the gains on nonzero-reference coordinates equal the raw coordinate-wise ratios of the precision-policy and reference gradients; the preconditioner changes only the coordinate weights and the zero-reference mass (Corollary 3). Thus, a common-state AdamW update can rotate far more than the raw gradient, making D_{u} a separate measurement of update direction.

Here $g \coloneqq g_\theta \in \mathbb{R}^d$ and $\tilde{g} \coloneqq \widetilde{g}_\theta$ denote the vectors actually supplied to that coordinate-wise rule; if global clipping is used, the theorem is applied to the clipped vectors because global clipping is not coordinate-wise. Write the coordinates of g and $\tilde{g}$ as g_i and $\tilde{g}_i$, indexed by the d parameter coordinates i. These coordinate indices are unrelated to the site indices of Appendix B.2.

A coordinate-wise update rule is a family $\Upsilon \coloneqq (\Upsilon_1, \dots, \Upsilon_d)$ of functions $\Upsilon_i : \mathbb{R} \to \mathbb{R}$; the rule produces the reference update $u^{\mathrm{ref}} \coloneqq (\Upsilon_i(g_i))_{i \le d}$ and the precision-policy update $u \coloneqq (\Upsilon_i(\tilde{g}_i))_{i \le d}$. The next result expresses the geometry of the two updates through optimizer-induced update gains and reference-update weights. These gains need not equal the raw ratios $\tilde{g}_i/g_i$ for nonlinear rules such as AdamW; equality holds for the fixed diagonal map treated in Corollary 3 below.

We write $D_{\mathrm{g}}^{\Upsilon}$, $P_{\mathrm{desc}}^{\Upsilon}$, $R_{\mathrm{norm}}^{\Upsilon}$, and $E_{\mathrm{rel}}^{\Upsilon}$ for the guard-free diagnostics of the pair (u^{ref}, u) in place of $(g_\theta, \widetilde{g}_\theta)$. In particular, $P_{\mathrm{desc}}^{\Upsilon}$ measures projection onto the reference update; $P_{\mathrm{desc}}^{\Upsilon}$ is not the first-order reference-loss descent coefficient unless u^{ref} is aligned with g. In the experimental tables, $D_{\mathrm{u}} \coloneqq D_{\mathrm{g}}^{\mathrm{AdamW}}$ denotes the angular distortion of the common-state AdamW update pair.

For any such rule with $u^{\mathrm{ref}} \neq 0$, let $I \coloneqq \{i : \Upsilon_i(g_i) \neq 0\}$. Define the update weights $\varpi_i \coloneqq \Upsilon_i(g_i)^2 / \|u^{\mathrm{ref}}\|_2^2$ and the update gains $\zeta_i \coloneqq \Upsilon_i(\tilde{g}_i)/\Upsilon_i(g_i)$ for $i \in I$, write $\mathbb{E}_\varpi[\cdot]$ for the ϖ-weighted average over I, and let

$$\lambda_0 \coloneqq \frac{\sum_{i \notin I} \Upsilon_i(\tilde{g}_i)^2}{\|u^{\mathrm{ref}}\|_2^2}$$

be the update mass on coordinates where the reference update vanishes.

Theorem 1 (Update geometry under coordinate-wise optimizers)**.** *Under the preceding notation, the diagnostics of the update pair are*

$$P_{\mathrm{desc}}^{\Upsilon} = \mathbb{E}_\varpi[\zeta], \qquad (R_{\mathrm{norm}}^{\Upsilon})^2 = \mathbb{E}_\varpi[\zeta^2] + \lambda_0, \qquad (E_{\mathrm{rel}}^{\Upsilon})^2 = \mathbb{E}_\varpi[(\zeta - 1)^2] + \lambda_0,$$
$$D_{\mathrm{g}}^{\Upsilon} = 1 - \frac{\mathbb{E}_\varpi[\zeta]}{\sqrt{\mathbb{E}_\varpi[\zeta^2] + \lambda_0}},$$

the last whenever $u \neq 0$.

The next corollary uses the weighted CV of the update gains.

Corollary 2 (Gain dispersion controls update bending)**.** *Under the setting of Theorem 1, if $\lambda_0 = 0$ and $\mathbb{E}_\varpi[\zeta] > 0$, then*

$$D_{\mathrm{g}}^{\Upsilon} = 1 - (1 + \mathrm{CV}_\varpi(\zeta)^2)^{-1/2}, \qquad \mathrm{CV}_\varpi(\zeta)^2 \coloneqq \frac{\mathbb{E}_\varpi[(\zeta - \mathbb{E}_\varpi\zeta)^2]}{(\mathbb{E}_\varpi\zeta)^2}.$$

Consequently, bending is a strictly increasing function of the nonnegative weighted CV $\mathrm{CV}_\varpi(\zeta)$.

For a diagonal preconditioner, the gains on nonzero-reference coordinates do not depend on the preconditioner. The reference weights are reweighted, and candidate mass on zero-reference coordinates can also change through λ_0.

Corollary 3 (Diagonal preconditioners reweight the same gains)**.** *Let $g \neq 0$ and let $\Upsilon_i(x) \coloneqq x/\omega_i$ with $\omega_i > 0$ be a diagonal preconditioner, written $\Omega \coloneqq \mathrm{diag}(\omega_1, \dots, \omega_d)$ and with diagnostics superscripted by Ω. Then, on coordinates with $g_i \neq 0$, the gains of Theorem 1 are $\zeta_i = \tilde{g}_i/g_i$ and do*

not depend on ω, whereas $\varpi_i \propto g_i^2/\omega_i^2$. On coordinates with $g_i = 0$, the remaining candidate mass is

$$\lambda_0^{\Omega} = \frac{\sum_{i:g_i=0} \tilde{g}_i^2/\omega_i^2}{\sum_j g_j^2/\omega_j^2}.$$

Two direct specializations recover familiar weighting schemes. The choice $\omega \equiv 1$ returns P_{desc}, R_{norm}, and $E_{\mathrm{rel}}^{\mathrm{param}}$, as well as D_{g} whenever $\tilde{g} \neq 0$, with the gradient-mass weights $\varpi_i = g_i^2/\|g\|_2^2$. If all $g_i \neq 0$, then $\lambda_0^{\Omega} = 0$ and the choice $\omega_i = |g_i|$ gives the uniform weights $\varpi_i = 1/d$.

For the reference vector, the choice $\omega_i = |g_i|$ is the Adam or RMS propagation (RMSProp) divisor when the second-moment estimate equals the current squared gradient and ϵ is neglected. Holding that divisor fixed for both candidate vectors gives the diagonal-preconditioner model above. Recomputing the divisor from each supplied vector changes the update gains and requires the general coordinate-wise formulation.

Theorem 1 explains how the common-state AdamW update can rotate far more than the raw gradient that produced the AdamW update. Raw Euclidean diagnostics weight nonzero-reference coordinates by g_i^2, strongly downweighting small coordinates. A fixed diagonal preconditioner preserves the raw gains $\tilde{g}_i/g_i$ on those coordinates but changes the coordinate weights to g_i^2/ω_i^2; if some reference coordinates vanish, any candidate update mass on those coordinates is represented separately by λ_0 and is also reweighted by the preconditioner. A general common-state AdamW map can change both the weights and the gains because the AdamW numerator and denominator depend nonlinearly on the supplied coordinate.

In Table 24, Wgrad-only quantization has raw $D_{\mathrm{g}} = 0.00049$ for the MXFP4 (MX) core but $D_{\mathrm{u}} = 0.077$, and the all-path rows show the same amplification from 0.227 to 0.679. Because raw-gradient and AdamW-update diagnostics can differ in both the coordinate weights and optimizer-induced gains used by each diagnostic, D_{u} separately measures the resulting change in update direction. Figure 8 in Appendix E.4 measures the gains and weights on GPT-2 AdamW states: The bias-corrected first step with zero moments is sign-like, so $D_{\mathrm{u}} \approx 2f$ for a sign-flip fraction f, and the all-low FP8 policy flips the sign of about half of the coordinates below 10^{-3} of the RMS gradient magnitude while barely disturbing the large coordinates that dominate D_{g}.

B.8 Implications for Precision Allocation

Eqs. 2 through 4 motivate three design choices for precision allocation. First, the propagated contribution combines incoming load, directional local error, and propagation-and-readout gain. Evaluating the parameter gradient captures the joint effect of all three factors. Second, contributions interact through pairwise alignment and the higher-order terms in $R_{\geq 2}^{\theta}$. In our measurements, sums of single-site shifts are 1.7 to 14.8 times the joint shift, so singleton benefits do not determine the benefit of a protected set (Appendices D.3 and D.4). Third, parallel and orthogonal contributions have different effects on gradient geometry: One changes scale, while the other changes direction (Sec. 4). Our allocation rule therefore evaluates the direction of the parameter gradient, or of the AdamW update, under each candidate mask.

Priorities Suggested by the Analysis. The propagation routes suggest architectural priorities for calibration. Quantizing logits directly perturbs the loss gradient, while forward errors can alter the backward operators of later layers. Errors introduced early also encounter more downstream nonlinearities. These routes explain the measured ordering: The logits boundary dominates the GPT-2 sites in Figure 2, forward operands cause more bending than backward operands, and the compute-budget path protects forward passes in early and middle layers first (Appendix E.8.1). The calibration probe measures the strength of these effects and uses the measured strengths to allocate high precision within the given precision allocation budget.

B.9 Protected-Set Curvature under the First-Order Surrogate

Bending-guided precision allocation (Sec. 5) has a direct interpretation in terms of the propagated contributions of Appendix B.3. Let $\widetilde{g}_{\theta}^{\pi,\mathcal{R}}$ be the executed optimizer gradient and define the marginal

protection vector

$$m_q(\mathcal{R}) \coloneqq \widetilde{g}_\theta^{\pi,\mathcal{R}\cup\{q\}} - \widetilde{g}_\theta^{\pi,\mathcal{R}}.$$

Under the additive first-order set model below, $m_q(\mathcal{R}) = -t_q^\theta$: Protecting q removes the combined propagated contribution of group q. In the realized pass, $m_q(\mathcal{R})$ additionally incorporates the current mask context and higher-order interactions. The measured reduction of D_{g} can arise through changes in both the orthogonal and parallel components of the executed gradient; the first-order energy surrogate below isolates the orthogonal part. The norm ratio and $E_{\mathrm{rel}}^{\mathrm{param}}$ provide the complementary scale-sensitive coordinates when candidate update magnitudes differ and when executed policies are ranked.

The greedy rule solves a combinatorial problem heuristically, so a characterization of the first-order surrogate associated with propagated contributions is useful. Under the first-order model of Eq. 3, assume that the loss gradient is unchanged, so $t_{\mathrm{term}}^\theta = 0$, and omit $R_{\geq 2}^\theta$. Each group $q \in \mathcal{Q}$ then contributes the propagated vector $t_q^\theta \coloneqq \sum_{k\in q} t_k^\theta$, and the precision-policy gradient of a low-precision set $S \subseteq \mathcal{Q}$ is $\widetilde{g}_\theta^S \coloneqq g_\theta + \sum_{q\in S} t_q^\theta$, with protected set $\mathcal{R} \coloneqq \mathcal{Q} \setminus S$.

Let $u_q \coloneqq \Pi_{g_\theta^\perp} t_q^\theta$ and $\Gamma_{ij} \coloneqq \langle u_i, u_j\rangle$, and define the orthogonal bending energy and the protection benefit

$$\mathcal{E}_\perp(S) \coloneqq \Big\|\sum_{q\in S} u_q\Big\|_2^2, \qquad V(\mathcal{R}) \coloneqq \mathcal{E}_\perp(\mathcal{Q}) - \mathcal{E}_\perp(\mathcal{Q}\setminus\mathcal{R}).$$

Proposition 2 (Alignment determines protected-set curvature)**.** *Let $\|g_\theta\|_2 > 0$ and $|\mathcal{Q}| \geq 2$. V is submodular if and only if $\Gamma_{ij} \geq 0$ for all $i \neq j$, supermodular if and only if $\Gamma_{ij} \leq 0$ for all $i \neq j$, and modular if and only if $\Gamma_{ij} = 0$ for all $i \neq j$. In the coherent case $\Gamma_{ij} \geq 0$, V is moreover nondecreasing with $V(\emptyset) = 0$.*

Proposition 2 characterizes the curvature of this first-order surrogate for protected-set selection. Positive pairwise alignment of propagated orthogonal contributions makes V monotone and submodular, whereas mixed signs allow both amplification and cancellation. The executed masks depart from the surrogate: The orthogonal components of the GPT-2 single-site contributions in Figure 5(c) are nearly all positively aligned, yet the sum of the single-site shifts is 1.7 times as long as the joint all-low shift on GPT-2 and 6.6 times as long on ResNet-18 (Appendix D.3), and protecting the twelve GPT-2 groups other than the LM head even raises the FP8 calibration bending, from 0.0880 to 0.0955 (Table 7). The executed selector of Sec. 5 minimizes the nonlinear D_{g} under the current mask. We evaluate the selector empirically and re-evaluate the executed mask after every pick to account for interactions; Appendix D.7 compares the executed rule with the exact optimum over all operation-by-depth (op-depth) protected sets.

B.10 Proofs

This subsection gives complete proofs of the formal statements, together with derivations of the metric identities in Eqs. 7, 8, and 9 and of the small-perturbation expansion of D_{g} stated in Appendix B.6. Throughout, inner products and norms are Euclidean on the relevant finite-dimensional spaces, and all diagnostics are interpreted in guard-free form, obtained by setting $\epsilon_{\mathrm{guard}} = 0$ in the diagnostic definitions.

Derivation of the metric identities in Eqs. 7, 8, and 9. Assume $\|g_\theta\|_2 > 0$ and $\|\widetilde{g}_\theta\|_2 > 0$, so that every quantity involved is defined in guard-free form. For Eq. 7, the definitions give directly

$$R_{\mathrm{norm}}(1 - D_{\mathrm{g}}) = \frac{\|\widetilde{g}_\theta\|_2}{\|g_\theta\|_2} \cdot \frac{\langle g_\theta, \widetilde{g}_\theta\rangle}{\|g_\theta\|_2\|\widetilde{g}_\theta\|_2} = \frac{\langle g_\theta, \widetilde{g}_\theta\rangle}{\|g_\theta\|_2^2} = P_{\mathrm{desc}}.$$

For Eq. 8, expanding the squared Euclidean norm,

$$\begin{aligned}(E_{\mathrm{rel}}^{\mathrm{param}})^2 = \frac{\|\widetilde{g}_\theta - g_\theta\|_2^2}{\|g_\theta\|_2^2} = \frac{\|\widetilde{g}_\theta\|_2^2 - 2\langle g_\theta, \widetilde{g}_\theta\rangle + \|g_\theta\|_2^2}{\|g_\theta\|_2^2} = R_{\mathrm{norm}}^2 - 2P_{\mathrm{desc}} + 1 \\ = R_{\mathrm{norm}}^2 + 1 - 2R_{\mathrm{norm}}(1 - D_{\mathrm{g}}) = (R_{\mathrm{norm}} - 1)^2 + 2R_{\mathrm{norm}}D_{\mathrm{g}},\end{aligned}$$

where the fourth equality substitutes Eq. 7. Both sides of the resulting identity are nonnegative. In particular, the right-hand side is nonnegative because the Cauchy–Schwarz inequality gives $D_{\mathrm{g}} \in [0, 2]$, so taking nonnegative square roots yields Eq. 8. Finally, Eq. 9 is the definition of S_{norm}. □

Proof of Proposition 1. The four identities come first. Abbreviate $g \coloneqq g_\theta$, $t \coloneqq t_k^\theta$, $\kappa^\parallel \coloneqq \kappa_k^\parallel$, and $\kappa^\perp \coloneqq \kappa_k^\perp$, and recall $\|g\|_2 > 0$. Define

$$t_\perp \coloneqq t - \frac{\langle g, t\rangle}{\|g\|_2^2} g = t - \kappa^\parallel g.$$

Then $\langle g, t_\perp\rangle = \langle g, t\rangle - \kappa^\parallel \|g\|_2^2 = 0$, and $t_\perp = \Pi_{g^\perp} t$ because $t - t_\perp = \kappa^\parallel g$ lies in $\mathrm{span}\{g\}$ while $t_\perp \perp g$; the definition of $\kappa^\perp$ therefore gives $\|t_\perp\|_2 = \kappa^\perp \|g\|_2$. The hypothesis $\widetilde{g}_\theta = g + t$ becomes the orthogonal decomposition

$$\widetilde{g}_\theta = (1 + \kappa^\parallel) g + t_\perp.$$

Three scalar quantities follow. First, by orthogonality of g and $t_\perp$,

$$\langle g, \widetilde{g}_\theta\rangle = (1 + \kappa^\parallel)\|g\|_2^2.$$

Second, by the Pythagorean theorem applied to the orthogonal decomposition of $\widetilde{g}_\theta$,

$$\|\widetilde{g}_\theta\|_2^2 = (1 + \kappa^\parallel)^2 \|g\|_2^2 + \|t_\perp\|_2^2 = [(1 + \kappa^\parallel)^2 + (\kappa^\perp)^2]\|g\|_2^2.$$

Third, $\widetilde{g}_\theta - g = t = \kappa^\parallel g + t_\perp$ is again an orthogonal decomposition, so

$$\|\widetilde{g}_\theta - g\|_2^2 = [(\kappa^\parallel)^2 + (\kappa^\perp)^2]\|g\|_2^2.$$

Substituting these three quantities into the guard-free diagnostic definitions gives the four identities:

$$P_{\mathrm{desc}} = \frac{\langle g, \widetilde{g}_\theta\rangle}{\|g\|_2^2} = 1 + \kappa^\parallel, \qquad R_{\mathrm{norm}} = \frac{\|\widetilde{g}_\theta\|_2}{\|g\|_2} = \sqrt{(1 + \kappa^\parallel)^2 + (\kappa^\perp)^2},$$
$$D_{\mathrm{g}} = 1 - \frac{\langle g, \widetilde{g}_\theta\rangle}{\|g\|_2 \|\widetilde{g}_\theta\|_2} = 1 - \frac{1 + \kappa^\parallel}{\sqrt{(1 + \kappa^\parallel)^2 + (\kappa^\perp)^2}},$$
$$E_{\mathrm{rel}}^{\mathrm{param}} = \frac{\|\widetilde{g}_\theta - g\|_2}{\|g\|_2} = \sqrt{(\kappa^\parallel)^2 + (\kappa^\perp)^2}.$$

The identity for D_{g} additionally uses $\|\widetilde{g}_\theta\|_2 > 0$, which by the second identity holds exactly when $(1 + \kappa^\parallel, \kappa^\perp) \neq (0, 0)$; in the excluded case $\widetilde{g}_\theta = 0$, the diagnostic D_{g} is undefined, while the other three identities remain valid.

The descent condition follows next. By the first identity, $P_{\mathrm{desc}} > 0$ if and only if $\kappa^\parallel > -1$, and by Eq. 6 the condition $P_{\mathrm{desc}} > 0$ is exactly first-order preservation of the reference descent direction by the step $-\eta\widetilde{g}_\theta$. Moreover, in the descent-preserving regime $1 + \kappa^\parallel > 0$, the third identity can be rewritten as

$$D_{\mathrm{g}} = 1 - (1 + (\kappa^\perp/(1 + \kappa^\parallel))^2)^{-1/2},$$

a strictly increasing function of the nonnegative ratio $\kappa^\perp/(1 + \kappa^\parallel)$, so gradient bending is controlled by the orthogonal component relative to $1 + \kappa^\parallel$. □

Derivation of the small-perturbation expansion of D_{g}. This derivation makes precise the expansion quoted after Proposition 1. For a perturbation with $1 + \kappa_k^\parallel > 0$, set

$$x \coloneqq \Big(\frac{\kappa_k^\perp}{1 + \kappa_k^\parallel}\Big)^2 \geq 0.$$

By the rewritten third identity in the proof of Proposition 1, $D_{\mathrm{g}} = 1 - (1+x)^{-1/2}$. Taylor's theorem with Lagrange remainder for $x \mapsto (1+x)^{-1/2}$ gives, for some $\xi \in (0, x)$ when $x > 0$ and trivially when $x = 0$,

$$(1+x)^{-1/2} = 1 - \tfrac{1}{2}x + \tfrac{3}{8}(1+\xi)^{-5/2}x^2,$$

and since $0 \le (3/8)(1+\xi)^{-5/2} \le 3/8$ for $\xi \ge 0$,

$$0 \le \tfrac{1}{2}x - D_{\mathrm{g}} \le \tfrac{3}{8}x^2.$$

Substituting the definition of x yields

$$D_{\mathrm{g}} = \frac{(\kappa_k^{\perp})^2}{2(1+\kappa_k^{\parallel})^2} + O((\kappa_k^{\perp})^4) = \frac{(\kappa_k^{\perp})^2}{2(1+\kappa_k^{\parallel})^2} + o((\kappa_k^{\perp})^2)$$

as $\kappa_k^{\perp} \to 0$, with the O and o statements uniform over any family of perturbations along which $1 + \kappa_k^{\parallel}$ stays bounded away from zero. In particular, the Cauchy–Schwarz inequality gives $|\kappa_k^{\parallel}| \le \|t_k^{\theta}\|_2 / \|g_\theta\|_2$; hence, in the small-singleton regime $t_k^{\theta} \to 0$, both $\kappa_k^{\parallel} \to 0$ and $\kappa_k^{\perp} \to 0$, and the uniformity condition holds. □

Proof of Lemma 1. Since $g_\theta \neq 0$ and $\widetilde{g}_\theta \neq 0$, the diagnostics D_{g}, P_{desc}, and R_{norm} are defined in guard-free form. The standard smoothness inequality for a function with β-Lipschitz gradient, applied with displacement $-\eta\widetilde{g}_\theta$, gives

$$\mathcal{L}(\theta - \eta\widetilde{g}_\theta) \le \mathcal{L}(\theta) - \eta\langle g_\theta, \widetilde{g}_\theta\rangle + \tfrac{\beta\eta^2}{2}\|\widetilde{g}_\theta\|_2^2.$$

By the definitions of the diagnostics and Eq. 7,

$$\langle g_\theta, \widetilde{g}_\theta\rangle = \|g_\theta\|_2^2 P_{\mathrm{desc}} = \|g_\theta\|_2^2 R_{\mathrm{norm}}(1 - D_{\mathrm{g}}), \qquad \|\widetilde{g}_\theta\|_2^2 = R_{\mathrm{norm}}^2 \|g_\theta\|_2^2.$$

Substituting these two identities and factoring out $\eta\|g_\theta\|_2^2$ yields Eq. 11 together with the stated form of the bracket. For the isotropic quadratic, expanding the square shows that the smoothness inequality is exact, so Eq. 11 is an equality. □

Proof of Corollary 1. Fix t with $g_\theta^{(t)} \neq 0$ and $D_{\mathrm{g}}^{(t)} < 1$, and abbreviate $D := D_{\mathrm{g}}^{(t)}$ and $R := R_{\mathrm{norm}}^{(t)} > 0$. The right-hand side of Eq. 11 equals $\mathcal{L}(\theta_t) - \|g_\theta^{(t)}\|_2^2 \psi(\eta)$ with

$$\psi(\eta) := R(1-D)\eta - \tfrac{\beta R^2}{2}\eta^2.$$

The function ψ is a strictly concave quadratic in η with $\psi'(\eta) = R(1-D) - \beta R^2 \eta$, which vanishes only at $\eta_t^\star = (1-D)/(\beta R) > 0$; hence ψ attains the maximum over $\eta > 0$ at $\eta_t^\star$, and $\psi(\eta_t^\star) = (1-D)^2/\beta - (1-D)^2/(2\beta) = (1-D)^2/(2\beta)$. Substituting gives the stated bound at $\eta_t^\star$. The exact gradient corresponds to $D = 0$ and $R = 1$, for which $\eta^\star = 1/\beta$ and $\psi(\eta^\star) = 1/(2\beta)$, so the ratio of the two guarantees is $(1-D)^2$; and $1 - D = \cos\phi_t$ by the definition of D_{g}.

Finally, for the isotropic quadratic direct expansion makes the right-hand side of Eq. 11 the exact loss after the step, so for that loss the largest one-step decrease achievable along $-\widetilde{g}_\theta^{(t)}$ over all $\eta > 0$ is exactly $(1-D)^2\|g_\theta^{(t)}\|_2^2/(2\beta)$, against $\|g_\theta^{(t)}\|_2^2/(2\beta)$ along $-g_\theta^{(t)}$. The factor $(1-D)^2$ is therefore attained and cannot be improved without further assumptions on $\mathcal{L}$. □

Proof of Theorem 1. Throughout, $u_i^{\mathrm{ref}} = \Upsilon_i(g_i)$ and $u_i = \Upsilon_i(\tilde{g}_i)$. The set $I = \{i : u_i^{\mathrm{ref}} \neq 0\}$ is nonempty because $u^{\mathrm{ref}} \neq 0$, and $\sum_{i\in I} \varpi_i = \sum_{i\in I}(u_i^{\mathrm{ref}})^2/\|u^{\mathrm{ref}}\|_2^2 = 1$, so ϖ is a probability vector on I and $\mathbb{E}_\varpi[f] := \sum_{i\in I}\varpi_i f_i$ for any family $(f_i)_{i\in I}$.

For $i \in I$, the definition of ζ_i gives $u_i = \zeta_i u_i^{\mathrm{ref}}$, while $u_i^{\mathrm{ref}} = 0$ for $i \notin I$. The three Euclidean quantities that enter the diagnostics are therefore

$$\langle u^{\mathrm{ref}}, u\rangle = \sum_{i\in I} u_i^{\mathrm{ref}}\zeta_i u_i^{\mathrm{ref}} = \|u^{\mathrm{ref}}\|_2^2 \sum_{i\in I}\varpi_i\zeta_i = \|u^{\mathrm{ref}}\|_2^2 \mathbb{E}_\varpi[\zeta],$$

$$\|u\|_2^2 = \sum_{i\in I}\zeta_i^2 (u_i^{\mathrm{ref}})^2 + \sum_{i\notin I} u_i^2 = \|u^{\mathrm{ref}}\|_2^2(\mathbb{E}_\varpi[\zeta^2] + \lambda_0),$$

$$\|u - u^{\mathrm{ref}}\|_2^2 = \sum_{i\in I}(\zeta_i - 1)^2 (u_i^{\mathrm{ref}})^2 + \sum_{i\notin I} u_i^2 = \|u^{\mathrm{ref}}\|_2^2(\mathbb{E}_\varpi[(\zeta-1)^2] + \lambda_0).$$

The guard-free definitions are $P^{\Upsilon}_{\mathrm{desc}} := \langle u^{\mathrm{ref}}, u\rangle / \|u^{\mathrm{ref}}\|_2^2$, $R^{\Upsilon}_{\mathrm{norm}} := \|u\|_2 / \|u^{\mathrm{ref}}\|_2$, $E^{\Upsilon}_{\mathrm{rel}} := \|u - u^{\mathrm{ref}}\|_2 / \|u^{\mathrm{ref}}\|_2$, and, for $u \neq 0$, $D^{\Upsilon}_{\mathrm{g}} := 1 - \langle u^{\mathrm{ref}}, u\rangle / (\|u^{\mathrm{ref}}\|_2 \|u\|_2)$; substituting the three equations above gives the four identities. □

Proof of Corollary 2. Let $m := \mathbb{E}_{\varpi}[\zeta] > 0$. When $\lambda_0 = 0$,

$$\mathbb{E}_{\varpi}[\zeta^2] = m^2 + \mathbb{E}_{\varpi}[(\zeta - m)^2] = m^2(1 + \mathrm{CV}_{\varpi}(\zeta)^2),$$
$$D^{\Upsilon}_{\mathrm{g}} = 1 - \frac{m}{\sqrt{m^2(1 + \mathrm{CV}_{\varpi}(\zeta)^2)}} = 1 - (1 + \mathrm{CV}_{\varpi}(\zeta)^2)^{-1/2},$$

where the second line follows from Theorem 1. The map $x \mapsto 1 - (1 + x^2)^{-1/2}$ is strictly increasing on $[0, \infty)$. □

Proof of Corollary 3. Retain the notation of the proof of Theorem 1. For $\Upsilon_i(x) = x/\omega_i$ one has $I = \{i : g_i \neq 0\}$, $\zeta_i = (\tilde{g}_i/\omega_i)/(g_i/\omega_i) = \tilde{g}_i/g_i$, $\varpi^{\Omega}_i = (g_i^2/\omega_i^2)/\Sigma$ with $\Sigma := \sum_j g_j^2/\omega_j^2 > 0$, and $\lambda^{\Omega}_0 = \sum_{i \notin I} (\tilde{g}_i^2/\omega_i^2)/\Sigma$.

For $\omega \equiv 1$, the quantities become $\varpi^{\mathrm{eu}}_i = g_i^2/\|g\|_2^2$ and $\lambda^{\mathrm{eu}}_0 = \sum_{i \notin I} \tilde{g}_i^2/\|g\|_2^2$, and $(u^{\mathrm{ref}}, u) = (g, \tilde{g})$. Thus, the diagnostics of Theorem 1 equal the gradient diagnostics of the pair $(g_\theta, \widetilde{g}_\theta)$ whenever the relevant diagnostics are defined; for example,

$$\sum_{i \in I} \varpi^{\mathrm{eu}}_i (\zeta_i - 1)^2 + \lambda^{\mathrm{eu}}_0 = \frac{\sum_{i \in I} (\tilde{g}_i - g_i)^2}{\|g\|_2^2} + \frac{\sum_{i \notin I} \tilde{g}_i^2}{\|g\|_2^2}$$
$$= (E^{\mathrm{param}}_{\mathrm{rel}})^2.$$

If all $g_i \neq 0$ and $\omega_i = |g_i|$, then $g_i^2/\omega_i^2 = 1$ for every i, so $\varpi_i = 1/d$.

For a reference Adam or RMSProp vector whose second-moment divisor is $\sqrt{v_i} = |g_i|$ with vanishing ϵ, the resulting map is the corresponding fixed-divisor model; recomputing the divisor from $\tilde{g}_i$ would change the gains and fall outside this specialization. □

Remark 1 (Fixed-state AdamW scope). Theorem 1 applies to a common-state AdamW comparison when the parameters, pre-update first and second moments, step index, and hyperparameters are held fixed across the two supplied vectors. The newly formed moments and the decoupled weight-decay update are then coordinate-wise functions of the corresponding supplied-gradient coordinate. If global norm clipping is part of the recipe, the theorem applies after the two vectors have been clipped; the global clipping map is not coordinate-wise.

Corollary 3 is the special map $x \mapsto x/\omega_i$ with a fixed divisor and no additive momentum or decay offset. A general AdamW update is therefore not obtained by merely reweighting the raw gains $\tilde{g}_i/g_i$.

Proof of Proposition 2. Write $n := |\mathcal{Q}| \geq 2$. Because $\Pi_{g_\theta^\perp}$ is linear and $u_q = \Pi_{g_\theta^\perp} t^\theta_q$, bilinearity of the inner product gives

$$\mathcal{E}_\perp(S) = \sum_{i \in S} \sum_{j \in S} \Gamma_{ij} = \sum_{q \in S} \Gamma_{qq} + \sum_{\substack{i,j \in S \\ i \neq j}} \Gamma_{ij}, \qquad \Gamma_{qq} = \|u_q\|_2^2. \tag{12}$$

For $\mathcal{R} \subseteq \mathcal{Q}$ and $q \notin \mathcal{R}$, define the marginal gain $\nu_q(\mathcal{R}) := V(\mathcal{R} \cup \{q\}) - V(\mathcal{R})$. Put $S := \mathcal{Q} \setminus (\mathcal{R} \cup \{q\})$, so that $\mathcal{Q} \setminus \mathcal{R} = S \cup \{q\}$ is a disjoint union. By the definition of V and Eq. 12,

$$\nu_q(\mathcal{R}) = \mathcal{E}_\perp(S \cup \{q\}) - \mathcal{E}_\perp(S) = \Gamma_{qq} + 2 \sum_{j \in S} \Gamma_{qj} = \Gamma_{qq} + 2 \sum_{j \in \mathcal{Q} \setminus (\mathcal{R} \cup \{q\})} \Gamma_{qj}. \tag{13}$$

Recall that a set function V on the subsets of $\mathcal{Q}$ is submodular if $\nu_q(\mathcal{R}) \geq \nu_q(\mathcal{R}')$ whenever $\mathcal{R} \subseteq \mathcal{R}' \subseteq \mathcal{Q} \setminus \{q\}$, supermodular if the reverse inequality always holds, and modular if equality always holds. For such $\mathcal{R} \subseteq \mathcal{R}'$, Eq. 13 gives

$$\nu_q(\mathcal{R}) - \nu_q(\mathcal{R}') = 2 \sum_{j \in \mathcal{R}' \setminus \mathcal{R}} \Gamma_{qj}.$$

If $\Gamma_{ij} \geq 0$ for all $i \neq j$, the right-hand side is nonnegative for every admissible pair, so V is submodular. Conversely, if V is submodular, then for any $q \neq j$ the choice $\mathcal{R} = \emptyset$, $\mathcal{R}' = \{j\}$ is admissible because $n \geq 2$ and gives $2\Gamma_{qj} \geq 0$. The supermodular statement follows by reversing both inequalities, and the modular statement by combining the two. In the coherent case, Eq. 13 gives $\nu_q(\mathcal{R}) \geq \Gamma_{qq} = \|u_q\|_2^2 \geq 0$ for every $\mathcal{R}$ and $q \notin \mathcal{R}$, so V is nondecreasing, and $V(\emptyset) = \mathcal{E}_\perp(\mathcal{Q}) - \mathcal{E}_\perp(\mathcal{Q}) = 0$. □

C Implementation, Protocols, and Code

This appendix defines every selection rule and baseline score, the simulated-quantizer and actual-kernel implementations, the evaluation protocols, the seed and pairing conventions, and the core selector source.

C.1 Selection-Rule and Baseline Details

Candidate Groups. The eligible sites are the outputs of the following modules: for ResNet-18, every convolution, batch normalization, ReLU, residual block, and the linear classifier, 58 sites in five operation families; for ConvNeXt Tiny (ConvNeXt-T), Shifted Window Transformer Tiny (Swin-T), and Vision Transformer Base with 16×16 patches (ViT-B/16), the stem, convolution, normalization, activation, attention, MLP, residual, and classifier modules; and for GPT-2, the token and position embeddings, every LN, attention block, and MLP block, and the LM head, whose output is the logits, 52 sites. GEMM-operand quantization covers the four linear layers of every GPT-2 block, whose Fprop, Dgrad, and Wgrad passes, grouped by layer and operation, form the pass units of Appendix E; the embeddings, normalization, attention scores, softmax, loss, and LM head stay in the reference precision. Our vision experiments instantiate $\mathcal{Q}$ mainly as operation-family groups. The same calibration objective also supports depth bands, op-depth intersections, and individual modules, so the candidate space can express depth-dependent quantization sensitivity. In these group labels, activation denotes activation-function modules such as ReLU, the Gaussian error linear unit, and the sigmoid linear unit; every eligible operation family can still produce a forward activation at the corresponding quantizer.

Bending-Guided Precision Allocation. Given a precision allocation budget B on the total cost $c(\mathcal{R})$, set $\mathcal{R}_0 \coloneqq \emptyset$ and, while some feasible action lowers D, add

$$r_t \in \operatorname*{arg\,max}_{q \in \mathcal{Q} \setminus \mathcal{R}_{t-1} : c(\mathcal{R}_{t-1}) + c_q \leq B} \frac{D(\mathcal{R}_{t-1}) - D(\mathcal{R}_{t-1} \cup \{q\})}{c_q}, \qquad \mathcal{R}_t \coloneqq \mathcal{R}_{t-1} \cup \{r_t\};$$

the rule then selects the better of the final set and the best single feasible action, the standard safeguard for knapsack budgets (Khuller et al., 1999). Each pick re-evaluates the executed mask to capture interactions among protected actions. The score is the reduction in gradient or update bending under that mask per unit cost. We call this rule bending-guided precision allocation. With $c_q \equiv 1$, a group budget B_{grp}, and $D = D_{\mathrm{g}}$, the rule reduces to the unit-cost selection $r_t \in \arg\min_{r \in \mathcal{Q} \setminus \mathcal{R}_{t-1}} D_{\mathrm{g}}(\pi, \mathcal{R}_{t-1} \cup \{r\})$ for $t = 1, \ldots, B_{\mathrm{grp}}$; the cache-cost variant charges activation-cache elements, and the variant that selects individual GEMM passes charges GEMM work and scores D_{u}, which counts the sign flips of small coordinates that the AdamW update amplifies (Sec. 4).

Measured BF16 Reference Floors. For the GPT-2 boundary-quantizer experiments, $\tau = 0.0034$, corresponding to a 4.7° rotation (Appendix D.7). The NVFP4-style GEMM-core experiments use $\tau = 0.0036$, measured for the BF16 reference path of the GEMM core (Appendix E.8). For ResNet-18, quantizing every module output and gradient to BF16 produces gradient bending of $D_{\mathrm{g}} = 0.0040$ and matches FP32 fine-tuning performance (Appendices D.1 and D.12).

Reading the Path. The bending–cost path gives two operational budget predictions for the candidate action space. When one action removes most of the bending, as the logits boundary does under boundary quantization, the cost of that action defines a minimum viable budget for substantial repair. Below this threshold, the available actions leave most of the bending intact. When many actions each remove a smaller amount, as in allocation to individual GEMM passes, the path decreases gradually.

The first point at which raw-gradient bending reaches τ defines the target-level budget, at which we predict learned weights with loss close to BF16 training. These predictions are made entirely from calibration; Sec. 6 compares both budget predictions with training outcomes.

Selector Roles. In the experiments, unit-cost bending-guided precision allocation targets direct angular repair of the optimizer gradient under a group-count budget, while the cache-cost variant and the budgeted allocation of Sec. 5 prioritize the same repair per unit cost. Reconstruction-error selection is the primary comparator that ignores propagation. Loss-sensitivity, activation-Fisher, and finite-difference Hessian selectors provide sensitivity-aware comparators, while random and fixed selectors represent simple policy priors. Together, these baselines separate direct angular repair from local, objective-aware, and cost-normalized alternatives.

A Common Criterion for Boundaries and Passes. Because D is measured on the vector that the optimizer receives, the rule places unlike actions on one scale: a module boundary, a group of boundaries, or a single Fprop, Dgrad, or Wgrad pass, under a memory, GEMM-work, or GEMM-time cost. Scoring all three passes on one scale is what shows, in Appendix E.8.1, that the backward path carries a small share of the bending while taking two thirds of the GEMM work.

Protected Local Backward Map. Let $h_k^{\pi,\mathcal{R}}$ denote the activation reaching module k under the current precision policy and protected set. At a protected site, the backward map uses reference precision at that realized activation point. Writing $\widetilde{B}_k(v; h, \pi_k)$ when the realized activation is explicit, define the protected local backward map by

$$\widetilde{B}_k^{\pi,\mathcal{R}}(v) := \begin{cases} J_k(h_k^{\pi,\mathcal{R}})^\top v, & k \in \mathcal{R}, \\ \widetilde{B}_k(v; h_k^{\pi,\mathcal{R}}, \pi_k), & k \notin \mathcal{R}. \end{cases}$$

The protected map agrees with $v \mapsto J_k(h_k)^\top v$ whenever $J_k(h_k^{\pi,\mathcal{R}}) = J_k(h_k)$. Equality of the realized and full-precision inputs is sufficient for this agreement, but not necessary; distinct inputs can have the same Jacobian, *e.g.*, in an affine module or within one linear region of a piecewise-affine module.

Baselines. Reconstruction-error selection scores the local relative quantization error of the concatenated module outputs without following the error to the parameter gradient or separating rescaling from rotation. Loss sensitivity measures the calibration-loss increase under one group's mask, while activation-Fisher and finite-difference Hessian scores use loss-gradient and curvature information along the quantization residual. These loss-based scores do not score quantization confined to Dgrad or Wgrad, which leaves the forward loss unchanged, or measure how AdamW amplifies small gradient errors. Under a precision allocation budget, each score-based baseline computes the exact knapsack optimum of the summed per-group scores, the integer-program form of budgeted Hessian-aware allocation (Yao et al., 2021). Bending-guided precision allocation measures the executed parameter-gradient or AdamW-update direction, distinguishes rotation from rescaling through bending and the norm ratio, and re-evaluates the mask after every pick to account for interactions. Random and fixed selection rules provide additional comparators. The following paragraphs define every baseline score, the cost models, and the implementation; Table 11 compares how well the scores predict training outcomes.

Reconstruction-Error Baseline. For the reconstruction-error selection baseline, let y_m be the output activation of module m, and let $y_q := (y_m)_{m \in q}$ be the concatenated output of group q. We write $Q(y_q) := (Q(y_m))_{m \in q}$ for modulewise quantization. The baseline selects the groups with the largest

$$\frac{\|Q(y_q) - y_q\|_2}{\|y_q\|_2 + \epsilon_{\text{guard}}},$$

where Q is the corresponding simulated quantizer. This score measures reconstruction error at the candidate group's module-output boundaries and does not account for downstream loss sensitivity or backward propagation.

Loss-Sensitivity, Curvature, and Prior Baselines. A loss-sensitivity baseline protects groups whose individual masks most increase calibration loss. Additional baselines include activation-Fisher and finite-difference Hessian scores, with the Hessian scores motivated by Hessian-aware mixed-precision quantization (Dong et al., 2019; 2020). We also compare random and fixed selection rules. These comparisons show how the bending signal differs from output-reconstruction, loss, curvature, and Fisher heuristics, and identify regimes in which the resulting candidate orderings diverge.

Activation-Fisher Baseline. For the diagonal empirical activation-Fisher proxy, we compute a full-precision forward and backward pass on the calibration batch and score a group q by

$$S_{\mathrm{Fisher}}(q) \coloneqq \sum_{m \in q} \|(Q(y_m) - y_m) \odot \nabla_{y_m} \mathcal{L}\|_2^2.$$

Finite-Difference Curvature Baseline. For the finite-difference Hessian proxy, we perturb the group output along the quantization residual $\delta_m \coloneqq Q(y_m) - y_m$. Let $\mathcal{L}_q(\cdot)$ denote the downstream loss as a function of a replacement group output, and let $\delta_q \coloneqq (\delta_m)_{m \in q}$ be the concatenated residual vector. We use the symmetric curvature score

$$S_{\mathrm{Hess}}(q) \coloneqq \max\{\mathcal{L}_q(y_q + \delta_q) + \mathcal{L}_q(y_q - \delta_q) - 2\mathcal{L}_q(y_q), 0\}.$$

For a quadratic Taylor model around y_q, the untruncated symmetric second difference is $\delta_q^\top H_q \delta_q$, where H_q is the Hessian of $\mathcal{L}_q$ with respect to the group output at y_q; the defined score is therefore $\max\{\delta_q^\top H_q \delta_q, 0\}$. The curvature contribution in the one-sided second-order Taylor model along $+\delta_q$ is $(1/2)\delta_q^\top H_q \delta_q$, but this constant factor does not change the top-B_{grp} group selection when the same nonnegative clipping is used. The activation-Fisher and finite-difference baselines choose the top-B_{grp} groups by score. Both use task-loss or loss-gradient information from the calibration batch in addition to the quantization residual.

Cache-Cost Bending-Guided Precision Allocation. Because groups have different activation-cache footprints, the cache-cost variant is the bending-guided rule of Sec. 5 with $c_q = C_q$. At greedy step t, the variant selects

$$r_t \in \operatorname*{arg\,max}_{\substack{q \in \mathcal{Q} \setminus \mathcal{R}_{t-1} \\ C_q > 0}} \frac{D_{\mathrm{g}}(\pi, \mathcal{R}_{t-1}) - D_{\mathrm{g}}(\pi, \mathcal{R}_{t-1} \cup \{q\})}{C_q},$$

where C_q is the activation-cache cost of group q defined below, so the resulting score measures bending removed per unit cache cost. The ResNet-18 and modern-backbone cache-cost runs apply this ratio for three picks. The budget-matched decoder study of Appendix D.7 fixes a footprint budget, admits only feasible groups, stops when no feasible group lowers the bending, and applies the single-group safeguard of Sec. 5.

Activation-Cache Cost. The memory cost is the extra cache space required to store protected module outputs in high precision. We calculate this memory cost from tensor shapes and the bytes per element of the two formats. GEMM time follows from measured kernel times or throughput ratios of low-precision and BF16 kernels (Appendix E.7).

On a calibration batch, the cost in mebibytes (MiB) for group q is

$$C_q \coloneqq \frac{1}{2^{20}} \sum_{m \in q} |y_m|(b_{\mathrm{FP32}} - b_{\mathrm{low}}),$$

where $|y_m|$ is the number of forward-output elements and b_{FP32} and b_{low} are the nominal bytes per element of the two formats. This reporting proxy charges protected outputs as FP32 cache entries independently of the storage used by every arithmetic reference path in the experiments. Charging BF16 entries at $b_{\mathrm{BF16}} - b_{\mathrm{low}}$ extra bytes per element rescales every cost of one format by the same constant, $1/3$ for FP8 and $3/7$ for NVFP4, and therefore leaves every budget-matched comparison unchanged; the budget-matched study also reports each cost as a fraction of the eligible footprint. The cache-cost selector above divides the marginal reduction of D_{g} by C_q, yielding the bending removed per unit cache cost. The model provides a reproducible activation-cache proxy for comparing selection rules under a shared memory budget, analogous to the budgeted role of hardware-aware and training-time precision-allocation methods (Wang et al., 2019; Lee et al., 2023).

GEMM-Work and Modeled-Time Costs. For the GEMM-operand cores of Appendix E, an action protects one Fprop, Dgrad, or Wgrad pass of a layer–operation group. For a fixed token batch, the three linear passes of a weight matrix have the same leading multiply–accumulate count, so the action cost is the number of protected weight elements, one pass-equivalent unit per element, and a group action costs three units per weight. Under a throughput ratio per floating-point operation, $r := T_{\mathrm{low}}/T_{\mathrm{BF16}}$, protecting a fraction p of the pass-equivalent work gives the modeled GEMM time $p + (1-p)/r$ relative to BF16, an additive cost proportional to the protected work. Appendix E.7 instantiates r with the measured FP8 kernel throughput of Table 19 and compares the model with the measured time of native NVFP4 and FP8 kernels.

Budgeted Baselines. Under a precision allocation budget, the reconstruction-error, loss-sensitivity, activation-Fisher, and finite-difference Hessian baselines take the exact knapsack optimum of the corresponding modular scores, computed by dynamic programming over integer element costs, so that zero or negative scores never consume budget. This rule is the integer-program form of budgeted Hessian-aware allocation (Yao et al., 2021). The random baseline adds groups in one fixed random order while the groups fit.

Search Cost. Because each probe is one forward and backward pass over the calibration batches, the search is priced in training tokens and does not grow with the length of the training run. In our GPT-2 studies, a probe processes 2,048 tokens for module boundaries and 4,096 tokens for single passes. The three-group memory search of Appendix D.7.1 takes 36 probes, about 74k tokens or 5 seconds on one laptop graphics processing unit (GPU).

For single passes, a coarse-to-fine search cuts the cost more than fourfold: Every action on the allocation path over individual GEMM passes of Appendix E.8.1 is an Fprop pass, and the three uniform masks that protect all Fprop, all Dgrad, or all Wgrad passes already show that the Fprop passes carry the bending, so comparing the three pass types first and then running the greedy search over Fprop passes alone returns the identical path to the target level with at most 301 probes, about 1.2M tokens, against 1,320 probes for the search over all passes.

Search-Cost Details. The allocation is prospective. The rule uses only the starting weights and a few calibration batches, the mask is frozen before training, and the search costs at most $\sum_t |\mathcal{Q} \setminus \mathcal{R}_{t-1}|$ executed-mask evaluations, each a forward and backward pass over the calibration batches. Under a first-order surrogate that sums the contributions of the unprotected groups, the protection benefit is monotone and submodular exactly when the propagated orthogonal contributions of distinct groups are pairwise aligned (Proposition 2 in Appendix B.9). The executed masks depart from this surrogate because quantization errors interact beyond first order, so the rule re-evaluates the executed mask after every pick, and Appendix D.7.1 compares the resulting masks with the exact optimum over all protected sets.

A greedy search over $|\mathcal{Q}|$ candidates with B_{grp} picks performs at most $\sum_{t=1}^{B_{\mathrm{grp}}} (|\mathcal{Q}| - t + 1)$ executed-mask evaluations, each a forward and backward pass over the calibration batches followed by one comparison with the stored reference gradients; a singleton ranking needs $|\mathcal{Q}|$ evaluations. For the GPT-2 decoder with four 2×256-token calibration batches, one evaluation processes 2,048 tokens, the token count of one training step, and takes 0.14 seconds under FP8 and 0.41 seconds under NVFP4 on one NVIDIA GeForce RTX 5090 Laptop GPU. The three-group op-depth search costs 36 evaluations, about 5 seconds under FP8, and the three-group per-module search costs 117 evaluations, which took 18 seconds. The FP4 search over individual GEMM passes at the 5.6% budget of Appendix E.6 performs 282 evaluations over eight calibration batches, about 1.2M tokens, or 0.9% of the 134.22M-token trajectory budget, and tracing the full path to 11.1% of the work takes 517 evaluations, including the all-low evaluation (Appendix E.7).

At the pretrained weights of Appendix E.8, the corresponding path takes 642 evaluations of 1.4 seconds each, about 15 minutes, including the all-low evaluation, and extending the path to the target level at 29.6% of the work takes 678 more evaluations of 3.6 seconds each, about 40 minutes, so the complete trace costs 1,320 evaluations of 4,096 tokens, about 5.4M tokens.

Every one of the 22 actions on this path is an Fprop pass, and the three uniform masks that execute every Fprop, every Dgrad, or every Wgrad GEMM in BF16 already rank the Fprop passes first, with calibration update bending 0.123 against 0.378 and 0.386 for the other two and 0.391 for the all-low

core. A two-stage search that compares the three pass types first and then runs the greedy path over the 24 Fprop actions alone therefore returns the identical path with at most 301 evaluations, three for the pass types, 297 for the greedy steps, and one for the all-low evaluation: At every step, the maximizer over all feasible actions is an Fprop action, so the maximizer over the feasible Fprop actions is the same action.

The executed cost-normalized selector scores each mask with the nonlinear diagnostic D_{g}, or the corresponding update-level diagnostic in the controlled FP4 study. We evaluate cache-cost allocation under footprint budgets and selection over individual GEMM passes under matched protected-work budgets; Appendix D.7 compares the executed rule with the exact optimum over all op-depth protected sets. Cost normalization remains operationally useful because a group-count budget can spend most of the available footprint on one expensive group when several cheaper groups repair more measured bending per MiB.

C.2 Implementation and Protocol Details

The simulated-precision boundary quantizer can quantize module-output activations, incoming activation gradients, or both, while the surrounding computation remains on standard PyTorch paths unless an actual-kernel path is explicitly stated. The boundary quantizer does not quantize module weights. The simulated quantizer configurations include BF16, FP16, FP8-E4M3, and FP8 with 5 exponent bits and 2 mantissa bits (FP8-E5M2), each with per-tensor or block-128 scaling, and the tensor-scaled symmetric FP4-E2M1 stress proxy. The default FP8 rows use one dynamic scale per tensor, while the block-128 comparison uses independent scales for contiguous 128-value blocks. These variants are implemented as autograd-compatible simulated-precision quantizers, following the same simulation motivation as low-precision arithmetic frameworks such as QPyTorch (Zhang et al., 2019). The actual-kernel experiments separately measure scaled-matmul paths in the NVIDIA Compute Unified Device Architecture (CUDA) stack.

Controls use BF16, FP16, FP8-E5M2, block-128 FP8, and a tensor-scaled FP4-E2M1 stress proxy. The decoder experiments also use NVFP4-style boundary quantizers with E2M1 values, 16-value blocks, and one FP32 scale per block. The simulated FP4 GEMM cores use the separate scaling rules specified in Appendix E.

In the formalism, full precision denotes the exact reference operator. Experimental tables label the reference precision as FP32 or BF16.

In the experiments, all-low FP8 denotes the policy in which all eligible groups use the per-tensor FP8-E4M3 simulated-precision quantizer on both module-output activations and incoming activation gradients; thus, all-low refers to coverage of all eligible boundary groups. Model weights and optimizer states retain the reference storage precision, and post-backward parameter gradients are quantized only in explicitly labeled ablations, namely the perturbation-site decomposition. Protection returns selected groups to the corresponding reference path.

We distinguish these boundary quantizers from the actual-kernel GEMM-operand setting. Actual-kernel FP8 GEMMs apply the stated policy to both activation and weight operands in the forward GEMM; the corresponding backward Dgrad and Wgrad GEMMs likewise use low-precision operands, while the trainable master weights and optimizer states retain the reference storage precision. The actual-kernel setting isolates the hardware scaled-matmul path by replacing selected GEMMs while leaving surrounding operation classes unchanged.

Throughout the actual-kernel comparisons, actual-kernel FP8 denotes per-tensor FP8-E4M3 activation and weight operands with BF16 output, unless another output data type (dtype) is listed explicitly. The term full-parameter denotes a training scope in which every model parameter is trainable. Under an actual-kernel policy, GEMMs selected for low precision use the FP8 scaled-matmul path. Under the coverage-matched simulated policy, the activation and weight operands are separately quantized and dequantized before standard matmul. In backward, the Dgrad GEMM applies the policy to the output-gradient and weight operands, and the Wgrad GEMM applies the stated policy to the output-gradient and saved-activation operands. Trainable master weights and optimizer states retain the reference storage precision; bias addition and bias-gradient reduction use the surrounding standard path.

Unless otherwise stated, the experiments use PyTorch 2.5.1 and torchvision 0.20.1 in a CUDA environment on NVIDIA GH200 hardware. BF16 and FP16 quantizers use direct 16-bit casts. FP8 quantizers follow the stated FP8 formats. The FP4 stress row simulates tensor-scaled symmetric 4-bit quantization. Each run specifies the model, dataset, precision policy, seed, and metrics needed for paired comparisons, and the sensitivity baselines are selected before downstream evaluation. We also evaluate matched GPT-2 pretraining trajectories in which every block GEMM uses an actual FP8 kernel. The controlled FP4 core experiments are explicit numerical simulations on NVIDIA GH200 hardware, while the separately labeled FP8 runs provide the hardware-executed accuracy and geometry baseline.

For group-output baselines, y_m denotes the output of module m, and y_q denotes the concatenation $(y_m)_{m \in q}$. The activation-Fisher baseline scores quantization residuals using the corresponding output gradients on the calibration batch. For the finite-difference Hessian proxy, $\mathcal{L}_q(\cdot)$ is the network loss when the output vector for group q is replaced by the perturbed group-output value, with parameters and non-descendant inputs fixed. In the activation-cache cost model, $|y_m|$ is the number of forward-output elements for module m.

The actual-kernel FP8 GEMM microbenchmarks use PyTorch `float8_e4m3fn` for both matrix operands and scaled matmul. The supported-kernel comparisons use FP8-E4M3 $\times$ FP8-E4M3 scaled matmul.

C.3 Evaluation Protocols

The empirical evaluation covers diagnostics on ImageNet-1K validation images and transfer fine-tuning on CIFAR-100. The fine-tuning experiments use ResNet-18, ConvNeXt-T, Swin-T, and ViT-B/16. The evaluation also includes a GPT-2 continued-pretraining study on WikiText-103 with an auxiliary LAMBADA proxy. Controlled experiments compare the simulated MX core with the simulated NVFP4-style (NV) core. These controlled experiments use fixed-state probes on models with up to 1.56B parameters and independently initialized three-seed GPT-2 trajectories at 124M and 355M parameters; actual-kernel FP8 trajectories serve as accuracy and geometry anchors.

The experiments follow the theoretical chain in Eqs. 3 and 4. First, the loss-plane experiment tests whether the orthogonal propagated component rotates the optimizer direction and reduces reference-objective descent. Second, source and mask ablations test whether forward-induced E_k, directional local error, propagation context, and mask interaction determine criticality more accurately than local reconstruction error alone. Third, protection experiments test whether removing groups with large marginal propagated contributions repairs gradient geometry and downstream training. Finally, actual-kernel evaluations test whether the same geometry remains informative on hardware FP8 paths.

ImageNet Validation. We evaluate the torchvision ImageNet-pretrained ResNet-18 model introduced by He et al. (2016) on ImageNet validation images. This setting provides a canonical residual-network evaluation of the diagnostic and protection workflow and a reference point for the ConvNeXt-T, Swin-T, and ViT-B/16 studies. The ImageNet precision-sweep gradient diagnostics use one calibration batch, and the accuracy columns of the sweep evaluate a 5,120-image random ImageNet validation subset, while the loss-plane visualization uses 16 fixed 16-image batches.

Gradient-Rotation Visualization. Figure 4 uses ImageNet-pretrained ResNet-18 in evaluation mode and 256 validation images drawn without replacement, divided into 16 consecutive 16-image batches. The deterministic preprocessing follows `ResNet18_Weights.IMAGENET1K_V1`. Forward hooks place a quantizer after every `Conv2d`, `BatchNorm2d`, `ReLU`, ResNet `BasicBlock`, and `Linear` output, instantiating the same five operation-family groups used by the main ResNet-18 study. A custom autograd operation also quantizes the incoming adjoint at each quantizer. BF16 uses a direct cast. FP8-E4M3 and FP8-E5M2 use the corresponding codebooks with one dynamic scale per realized tensor. The tensor-scaled FP4 stress uses the signed E2M1 magnitude codebook $\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$. All diagnostics compare the resulting concatenated parameter gradient with the unquantized FP32 gradient on the same batch. All-low policies cover five operation groups. The FP4-E2M1 row is a tensor-scaled fake-quantization stress proxy; Appendix E analyzes the controlled MX and NV cores separately.

For Figure 4(a), the representative batch is fixed as the batch with the upper-median FP8-E4M3 angle among the 16 measured batches, *i.e.*, the ninth-smallest angle at $23.60°$; the sample median of the 16 angles marked in Figure 4(b) is $23.15°$. Define the reference, low-precision, and orthogonal in-plane directions by

$$u_{\mathrm{ref}} \coloneqq -\frac{g_\theta}{\|g_\theta\|_2}, \qquad u_{\mathrm{low}} \coloneqq -\frac{\widetilde{g}_\theta}{\|\widetilde{g}_\theta\|_2},$$
$$u_\perp \coloneqq \frac{u_{\mathrm{low}} - \langle u_{\mathrm{ref}}, u_{\mathrm{low}}\rangle u_{\mathrm{ref}}}{\|u_{\mathrm{low}} - \langle u_{\mathrm{ref}}, u_{\mathrm{low}}\rangle u_{\mathrm{ref}}\|_2}.$$

The rendered surface evaluates the unquantized FP32 loss at $\theta + au_{\mathrm{ref}} + bu_\perp$ on a 29×29 grid. The common arrow length $\rho = 0.01049$ follows a separate plane-radius rule chosen for visible contour variation: the displacement whose linear FP32 loss reduction is 22% of the loss, clipped to $[2 \times 10^{-4}, 4 \times 10^{-2}]$ and then adjusted, if necessary, so that the realized FP32 loss decrease along u_{ref} is finite and at least 10% of the loss. On the representative batch, this rule needs no adjustment and yields the reported endpoint losses of 1.576 and 1.597.

For Figure 4(c), each precision direction and the FP32 direction receive the same, much smaller, per-batch displacement ρ_{c}, chosen so that the linear FP32 loss reduction is 3% of the loss and clipped to $[2 \times 10^{-5}, 10^{-2}]$; on the representative batch, $\rho_{\mathrm{c}} = 0.00143$. Endpoint losses are always evaluated with the quantizers disabled, so that comparison tests all directions against one shared FP32 objective. The run used PyTorch 2.9.1, torchvision 0.24.1, CUDA 12.8, and an NVIDIA GeForce RTX 5090 Laptop GPU. The experiment script records the per-batch metrics, aggregate summary, and evaluated loss grid used to render the figure.

Gradient-Rotation Measurements. Across the 16 batches, the mean angle between the full-precision and precision-policy gradients increases from $4.54°$ for BF16 to $23.60°$ for FP8-E4M3, $37.18°$ for FP8-E5M2, and $87.41°$ for the FP4 stress proxy. The corresponding mean D_{g} values are 0.0032, 0.0844, 0.2065, and 0.9548. The representative FP8-E4M3 batch in Figure 4(a) is the ninth-smallest angle of the 16, the upper median, at $23.6°$, corresponding to $D_{\mathrm{g}} = 0.0836$. Starting from a full-precision loss of 1.978, equal-length FP32 and FP8-E4M3 steps reach losses of 1.576 and 1.597, respectively, on the reference surface in Figure 4(a). Across batches, the measured equal-length loss-decrease ratios are 0.999, 0.919, 0.796, and 0.047 in the same precision order and closely track $\cos\phi$. Equalizing the candidate step lengths isolates the directional effect measured by $D_{\mathrm{g}} = 1 - \cos\phi$. Under this tensor-scaled FP4-E2M1 stress proxy, two of the 16 batches cross the $90°$ first-order-descent boundary; the most distorted batch produces an actual loss increase.

CIFAR-100 Transfer Fine-Tuning. We fine-tune all parameters of ImageNet-pretrained ResNet-18 on CIFAR-100. The baseline recipe runs for four epochs and uses 224×224 random resized crops, RandAugment, horizontal flips, AdamW (Loshchilov & Hutter, 2019), cosine decay, label smoothing of 0.1, gradient clipping at 1.0, and a batch size of 256. The backbone and classifier learning rates are 3×10^{-5} and 3×10^{-4}, respectively, with weight decay 0.05. We report five runs per policy on the full 50,000-image training set and the 10,000-image test set. This full-dataset transfer experiment tests whether low-precision gradient geometry affects an actual training outcome.

Modern Vision-Backbone CIFAR-100 Fine-Tuning. For additional modern vision backbones, we fine-tune all parameters of ImageNet-pretrained ConvNeXt-T, Swin-T, and ViT-B/16 (Liu et al., 2022; 2021; Dosovitskiy et al., 2021) on CIFAR-100 for six epochs. The recipe uses the same 224×224 ImageNet-style augmentation family as the ResNet-18 transfer run, together with AdamW, cosine decay, label smoothing of 0.1, gradient clipping at 1.0, and a batch size of 128. The backbone and classification-head learning rates are 3×10^{-5} and 3×10^{-4}, respectively, with weight decay 0.05. We report five runs per policy for ConvNeXt-T and three for Swin-T and ViT-B/16 on the full 50,000-image training set and the 10,000-image test set. Protected sets are selected from a held-out CIFAR-100 calibration gradient batch, and we compare FP32, all-low FP8, and cache-cost bending-guided precision allocation.

Decoder-LM Continued Pretraining. For the decoder-only LM study, we perform continued pretraining of the 124-million-parameter GPT-2 model on WikiText-103 from shared pretrained weights. We concatenate tokenized nonempty entries from the WikiText-103 training split, insert an

end-of-sequence token after each entry, and divide the resulting token stream into 8,192 nonoverlapping 256-token sequences. We construct 512 validation sequences of the same length in the same way from the validation split. The optimizer is AdamW with a learning rate of 5×10^{-5}, a weight decay of 10^{-2}, a linear warmup ratio of 0.03, and a batch size of 8. We apply gradient clipping at 1.0 and make one full pass over the training sequences. The BF16 baseline uses autocast.

Continued pretraining starts each policy from the weights used for calibration, controls initialization effects, and corresponds to the late phase in which FP4 training systems change precision (Chmiel et al., 2025). Appendix E adds from-scratch trajectories at 124M and 355M parameters.

A longer-horizon evaluation uses four times as many training sequences for the strongest policy subset. As an auxiliary downstream proxy, we evaluate the final weights on 512 examples from the LAMBADA `plain_text` test split by predicting only the final GPT-2 token, without LAMBADA-specific training. When the target word spans multiple tokens, the word's earlier tokens remain in the input context; therefore, this last-token proxy is not directly comparable to commonly reported LAMBADA full-word accuracy.

Within each decoder comparison, policies are matched on initial pretrained weights, tokenization, sequence construction, training schedule, validation split, and downstream proxy. Validation loss and validation perplexity (PPL) are evaluated under the quantizers of the corresponding policy.

The FP8 decoder rows follow the simulated-precision quantizer definition above. The decoder coverage and the NVFP4-style boundary quantization are specific to this decoder-LM experiment. This 4-bit boundary quantization uses E2M1 values in 16-value blocks, each block scaled by one FP32 factor set by the largest magnitude in that block, without the E4M3 block-scale encoding of the NV core in Appendix E. The NVFP4-style codebook is $\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$ with sign restored. Rows modeled on the quantization-aware fine-tuning (QAF) phase of Chmiel et al. (2025) apply the forward perturbation but disable backward-gradient quantization, making the QAF rows backward-high-precision analogues.

C.4 Seed Protocol and Pairing Details

Entries of the form $x \pm y$ report the mean and standard deviation over the runs, batches, seeds, or held-out splits identified in the corresponding caption. Bracketed ranges report bootstrap 95% confidence intervals where shown. The seed counts and pairing rules behind these entries are as follows.

The gradient-rotation visualization uses one fixed sampling seed. The main budget-matched decoder boundary and GEMM-core comparisons use three paired training seeds (Appendices D.7 and E.8). The earlier granularity comparison in Table 8 and the instrumented decoder trajectories use single-seed realizations, as specified in the corresponding captions and protocols.

The five-run CIFAR-100 settings use five experiment seeds. This protocol applies to the ResNet-18 full-parameter fine-tuning setting and the ConvNeXt-T runs. The Swin-T and ViT-B/16 simulated-precision quantizer runs use a three-seed subset.

Simulated-precision ImageNet gradient diagnostics use the calibration batches stated in each table. The calibration batch used for bending-guided precision allocation is separate from the batch used for the precision sweep. The 5,120-image ImageNet subset comparison uses the corresponding precision-sweep calibration setting.

For CIFAR-100 fine-tuning, all policies compared within a setting share the same seed set. Seed-paired deltas are computed by comparing a policy against the FP32 or all-low FP8 run with the same backbone and seed. For the modern-backbone comparison in Table 18, ConvNeXt-T uses the five-seed set. Swin-T and ViT-B/16 use the three-seed subset. All reported deltas in that table use these within-backbone paired runs. Protected sets for the modern-backbone fine-tuning runs are selected separately for each seed from a held-out CIFAR-100 calibration gradient batch. The reported activation-cache costs of the selected protected sets are averaged over seeds.

The controlled 124M and 355M from-scratch trajectories use three initialization seeds per policy. Policies compared within a trajectory setting share initialization and data order, and the actual-kernel FP8 baselines use the same three-way pairing. The 355M WikiText-103 learning rate sweep uses two tuning seeds. The intervention on individual GEMM passes uses one initialization for selection and

held-out evaluation and one independent initialization for frozen-mask transfer, with eight evaluation batches in each state.

Table summaries use matched splits, seed sets, horizons, and recipes within each experimental regime.

C.5 Core Python Source Code

Listing 1 gives the core selection routine used by the unit-cost and cache-cost bending-guided experiments. The helper `gradient_for_policy` denotes the experiment-specific routine that returns the concatenated parameter gradient under a precision policy, and `policy_cls` is the precision-policy class of the experiment code, whose `rescued_groups` argument lists the protected groups. A full-precision calibration gradient is compared with gradients obtained under candidate low-precision masks, and the next protected group is chosen by the decrease in D_{g} per unit activation-cache cost. Here `budget` is the number of groups selected; setting all costs to one recovers the unit-cost rule. The resource-budgeted variant is given separately in Listing 2.

Listing 1: Bending-guided selection with a per-group cost.

```python
import torch
EPS = 1.0e-12

def d_grad(ref_grad, test_grad):
    ref = ref_grad.detach().float().reshape(-1)
    test = test_grad.detach().float().reshape(-1)
    dot = torch.dot(ref, test).item()
    denom = max(ref.norm().item() * test.norm().item(), EPS)
    cosine = min(1.0, max(-1.0, dot / denom))
    return 1.0 - cosine

def bending_guided_selection(model, batch, groups, low_precision,
                             costs, budget, gradient_for_policy,
                             policy_cls):
    groups = frozenset(groups)
    selected = set()
    ref_grad = gradient_for_policy(
        model, batch, policy_cls("fp32", frozenset()))
    current = d_grad(
        ref_grad,
        gradient_for_policy(model, batch,
                            policy_cls(low_precision, groups)))
    for _step in range(min(budget, len(groups))):
        candidates = []
        for group in sorted(groups - frozenset(selected)):
            protected = frozenset(selected | {group})
            policy = policy_cls(low_precision, groups,
                                rescued_groups=protected)
            candidate_grad = gradient_for_policy(model, batch, policy)
            candidate_d = d_grad(ref_grad, candidate_grad)
            cost = max(float(costs[group]), EPS)
            score = (current - candidate_d) / cost
            candidates.append((score, -candidate_d, protected))
        _, neg_d, protected = max(candidates)
        selected = set(protected)
        current = -neg_d
    return frozenset(selected)
```

Listing 2 gives the budgeted variant of Sec. 5 used in Appendices D.7 and E.7. The callable `bending` returns the calibration-batch bending of an executed protected mask, D_{g} for the decoder boundary quantizers and D_{u} for the FP4 cores, and caches every evaluated mask; `costs` holds the additive action costs in elements.

Listing 2: Budgeted bending-guided precision allocation.

Table 2: ImageNet precision sweep on pretrained ResNet-18. Gradient diagnostics use one calibration batch. Accuracy uses the shared 5,120-image validation subset.

Setup	Gradient			Accuracy	
Precision	D_{g}	P_{desc}	R_{norm}	Top-1 (%)	Top-5 (%)
BF16	0.0040	0.9972	1.0012	69.51	89.59
FP16	0.0005	0.9997	1.0002	69.61	89.61
FP8-E4M3 (per-tensor)	0.0952	0.8092	0.8943	67.97	89.28
FP8-E4M3 (block-128)	0.0957	0.8561	0.9467	69.08	89.43
FP8-E5M2 (per-tensor)	0.2614	0.6720	0.9098	65.06	88.28
FP8-E5M2 (block-128)	0.2780	0.6708	0.9290	66.70	88.81
FP4 stress	0.9905	0.0041	0.4263	0.16	0.53

```python
def budgeted_assignment(bending, costs, budget):
    selected, used = frozenset(), 0.0
    current = bending(selected)
    while True:
        best = None
        for action in sorted(set(costs) - selected):
            if used + costs[action] > budget:
                continue
            gain = current - bending(selected | {action})
            ratio = gain / costs[action]
            if best is None or ratio > best[0]:
                best = (ratio, action)
        if best is None or best[0] <= 0.0:
            break
        selected = selected | {best[1]}
        used += costs[best[1]]
        current = bending(selected)
    singles = [(bending(frozenset({a})), a)
               for a in sorted(costs) if costs[a] <= budget]
    if singles and min(singles)[0] < current:
        selected = frozenset({min(singles)[1]})
    return selected
```

D Extended Boundary-Quantizer Results and Actual-Kernel Checks

This appendix reports the full boundary-quantizer results: the ImageNet precision sweep, the site-level and mechanistic analyses behind Figure 2, the complete decoder-LM outcome and diagnostic tables, the vision transfer details, and the auxiliary actual-kernel checks. Each results subsection of this appendix and of Appendix E closes with a short summary paragraph.

D.1 ImageNet Precision Sweep

The FP4 setting in Table 2 is an extreme tensor-scaled fake-quantization stress test designed to probe near-total descent collapse. Appendix E analyzes the controlled MX and NV cores. The BF16 and FP16 settings serve as separate 16-bit controls and remain close to FP32, so the main degradation studied in the experiments is specific to FP8-level precision. Block-128 scaling substantially recovers subset top-1 relative to the corresponding per-tensor FP8 quantizer, with a nonmonotonic change in D_{g}. The sweep shows that format and block-scaling choices affect forward accuracy, descent projection, norm ratio, and angular gradient distortion in different proportions. The primary low-precision setting throughout the experiments is the all-low FP8 policy, *i.e.*, the per-tensor FP8-E4M3 quantizer applied to all eligible groups.

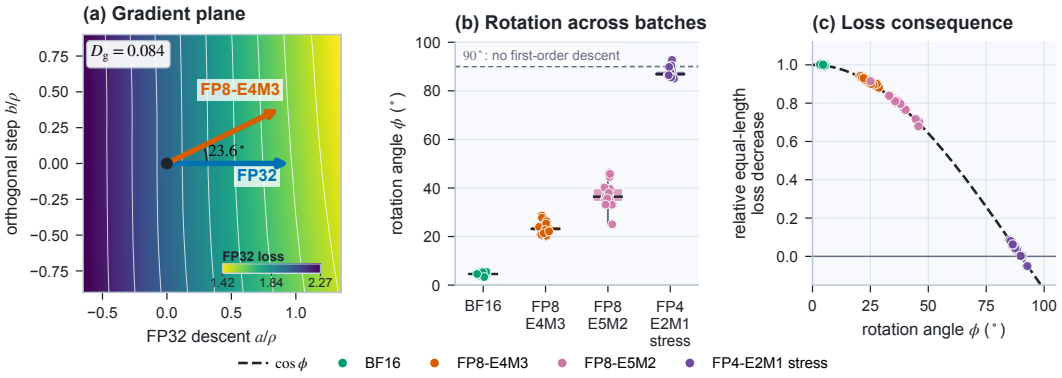


Figure 4: Measured gradient bending on ImageNet-pretrained ResNet-18. (a) FP32 loss surface with equal-length FP32 and FP8-E4M3 directions on the upper-median-angle FP8 batch. (b) Rotation angles over 16 fixed batches, shown as medians with interquartile ranges. (c) Equal-length reference-loss decrease relative to the FP32 direction; the dashed curve is $\cos\phi$.

Summary. The 16-bit quantizers are nearly neutral, per-tensor FP8-E4M3 and FP8-E5M2 bend the ResNet-18 gradient by $D_g = 0.095$ and 0.26, and the FP4 stress proxy collapses descent. Block-128 scaling raises subset top-1 accuracy and moves the norm ratio toward one without lowering D_g, so a scaling choice can repair forward accuracy and gradient scale while leaving the gradient bending in place.

D.2 Direct Visualization of Gradient Bending

Figure 4 visualizes gradient bending on the reference loss surface. Across the 16 batches, the mean angle between the full-precision and precision-policy gradients increases from $4.54°$ for BF16 through $23.60°$ for FP8-E4M3 to $87.41°$ for the FP4 stress proxy. The equal-length loss-decrease ratios closely track $\cos\phi$, and two FP4 batches cross the $90°$ first-order-descent boundary; Appendix C.3 lists the per-format values.

Summary. Lower precision rotates the ResNet-18 gradient further, and the reference-loss decrease of an equal-length step closely follows $\cos\phi$, so the gradient bending $D_g = 1 - \cos\phi$ measures the fraction of first-order descent that a precision policy loses at a fixed step length. Under the FP4 stress proxy, two of the 16 batches cross the $90°$ descent boundary.

D.3 Site-Level Factorization and Pairwise Alignment

Similar Local Errors, Different Routes. With dynamic scaling, relative operand error depends strongly on the format. Per-tensor FP8-E4M3 produces about 2.6% relative error at all 58 ResNet-18 module-output sites and at 50 of the 52 GPT-2 sites (Sec. 3). Quantizing one operand at a time in every GPT-2 block GEMM to the NVFP4-style format likewise gives similar relative errors across the six operand roles in Eq. 1, between 0.077 and 0.099 (Table 25). The sites nevertheless differ in propagation routes, incoming gradient magnitudes, and amplification through the network.

The expansion motivates three empirical comparisons in Secs. 3 and 6: whether local reconstruction error ranks sites by the resulting parameter-gradient shift; whether forward-induced changes dominate gradient distortion; and whether protecting the forward computation repairs more distortion than protecting the backward path.

The factorization in Eq. 4 applies only to the activation-propagation term $t^{\theta}_{k,\mathrm{act}}$. A direct Wgrad or parameter-gradient-readout contribution $t^{\theta}_{k,\mathrm{par}}$ is not covered by this activation-side factorization and must be added vectorially before the parallel and orthogonal geometry of the combined t^{θ}_{k} is evaluated.

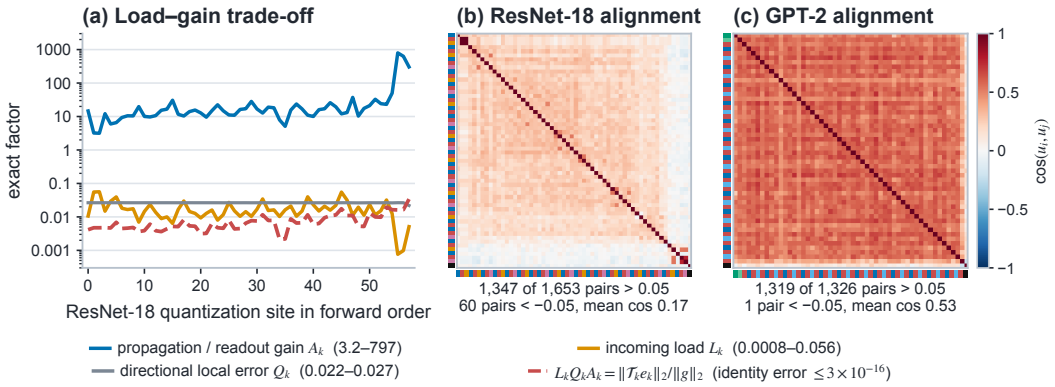


Figure 5: Site-level factorization and pairwise alignment for the single-site quantizers of Figure 2. (a) The three factors of Eq. 4 and the product of those factors for the backward-only quantizer on every ResNet-18 site. (b, c) Cosines between the orthogonal components u_i of the single-site contributions on the first calibration batch; edge strips mark site families as in Figure 2.

The single-site quantizers of Figure 2 use four fixed calibration batches, 64 ImageNet validation images per batch for ResNet-18 and eight protocol-built 256-token WikiText-103 validation blocks per batch for GPT-2. In Figure 2(a), on ResNet-18 the local relative reconstruction error of every site lies between 0.0264 and 0.0267, whereas the parameter-gradient shift spans 0.017 at the last residual block to 0.293 at the stem BatchNorm and ReLU, a 17× range that decays with depth and rises again to 0.128 at the classifier; the all-low policy gives 0.43.

In Figure 2(b), on GPT-2, 51 of 52 sites shift the gradient by between 0.019 and 0.029, while the logits site alone shifts the gradient by 0.621 with the same local error of 0.026 and raises the calibration loss from 3.82 to 5.76; the all-low policy gives 0.593 on these batches and 0.48 on the selection batches of Table 6. For every single-site shift, the dominant component is orthogonal to g_θ: The ratio $\kappa_k^\perp / |\kappa_k^\parallel|$ ranges from 7.9 to over 1,000 across the ResNet-18 sites and from 6.4 to over 2,000 across the GPT-2 sites, with medians of 22 and 30, and the GPT-2 logits site carries the largest parallel component, $\kappa^\parallel = 0.095$ against $\kappa^\perp = 0.61$.

On ResNet-18, the stem sites are most critical because noise from the stem crosses every downstream ReLU boundary (Appendix D.4); the classifier is also critical because quantized logits perturb the loss gradient. On GPT-2, the logits site dominates because the large dynamic range of the logits makes per-tensor E4M3 spacing span several logit units, and the softmax turns the 2.6% reconstruction error into a parameter-gradient shift above 60%. Every other site produces a much smaller shift when quantized alone. The labels in Figure 2(b) anticipate Appendix D.7.1: Bending-guided precision allocation protects the LM head first, whereas reconstruction-error selection protects the final LayerNorm, whose local error is among the largest but whose gradient shift is about twenty times smaller than the gradient shift of the LM head.

Figure 5 continues the single-site analysis of Sec. 3 with the exact factorization of Eq. 4 and the pairwise alignment of the propagated orthogonal components. Figure 5(a) shows that incoming load and propagation-and-readout gain trade off across depth while directional local error stays flat, so no single factor orders the sites. For the backward-only quantizer, the factorization is exact on every ResNet-18 site to a relative residual below 3×10^{-16}: L_k falls from 0.056 to 0.0008 toward the head, A_k rises from 3.2 to 797, and Q_k stays between 0.022 and 0.027.

Figures 5(b) and 5(c) measure the pairwise alignment Γ_{ij} of Proposition 2. Of the site pairs, 1,347 of 1,653 for ResNet-18 and 1,319 of 1,326 for GPT-2 have cosine above 0.05; only 60 and 1, respectively, fall below -0.05. The first-order surrogate of Proposition 2 is therefore close to the coherent regime on both networks, although the negative ResNet-18 pairs make the approximation imperfect. The classifier and LM-head rows are nearly orthogonal to the other sites, giving the readout sites distinct directions and explaining the inclusion of the readout sites in both greedy protected sets. Nevertheless, the single-site shifts sum to 6.6× the joint all-low shift on ResNet-18 and 1.7× on GPT-2, so the

Table 3: Forward-activation and backward-activation-gradient decomposition on the ImageNet calibration batch. FP8 is the per-tensor FP8-E4M3 simulated-precision boundary quantizer.

Setup	Gradient			
Policy	D_{g}	P_{desc}	R_{norm}	$E_{\mathrm{rel}}^{\mathrm{param}}$
FP8 output activations, FP32 activation gradients	0.0941	0.8113	0.8956	0.4237
FP32 output activations, FP8 activation gradients	0.0016	1.0004	1.0020	0.0567
FP8 output activations and activation gradients	0.0952	0.8092	0.8943	0.4259
FP8 outputs and activation gradients, normalization protection	0.0844	0.9766	1.0666	0.4295
FP8 outputs and activation gradients, normalization and residual protection	0.0861	0.9767	1.0687	0.4345
FP8 outputs and activation gradients, activation, normalization, and residual protection	0.0697	0.9556	1.0272	0.3793

executed masks depart from the additive surrogate. The selector of Sec. 5 therefore re-evaluates the executed mask after every pick to account for interactions among sites.

Summary. Similar local errors of about 2.6% produce single-site gradient shifts that span 17-fold across ResNet-18 sites and 32-fold across GPT-2 sites, where the logits dominate, and every single-site shift is mostly orthogonal to the reference gradient. No single factor of Eq. 4 orders the sites: Incoming load and propagation-and-readout gain trade off across depth while directional local error stays flat. Nearly all site pairs are positively aligned, yet the single-site shifts sum to 6.6× the joint all-low shift on ResNet-18 and 1.7× on GPT-2, so the selector re-evaluates the executed mask after every pick.

D.4 Mechanistic Ablations

In ResNet-18, flipped ReLU masks carry 76% of the gradient shift caused by forward quantization, the perturbed loss gradient 21%, and the quantized Wgrad inputs 3% (Figure 6).

Where the Rotation Comes From. Proposition 1 in Appendix B.6 splits a site's propagated contribution into a component along g_θ, $\kappa^{\parallel}$, which rescales the optimizer gradient, and an orthogonal component, $\kappa^{\perp}$, which rotates the optimizer gradient. At every one of the 58 ResNet-18 and 52 GPT-2 sites in Figure 2, $\kappa^{\perp}$ exceeds $|\kappa^{\parallel}|$, by median factors of 22 and 30, so the propagated quantization error mostly rotates the gradient. Forward quantization can produce this rotation through flipped ReLU masks; Sec. 4.2 discusses sign flips under AdamW.

Flipped ReLU Masks. Forward quantization can flip ReLU masks. If the gradient arriving at three ReLU units is (a, a, a) and the mask of those units is $(1, 1, 0)$, the gradient at the pre-activations is $(a, a, 0)$, and quantization that switches the second unit off and the third on turns that gradient into $(a, 0, a)$, a vector of the same norm at 60° from the original, the rotated example in Sec. 4. In ResNet-18, flipped masks carry 76% of the forward-induced shift, with $\kappa^{\perp} = 0.37$ against $\kappa^{\parallel} = -0.06$ (Figure 6).

Forward and Backward Perturbation. Table 3 provides the forward and backward decomposition and shows that, for this ResNet-18 setting, forward-induced activation perturbation is the dominant source of parameter-gradient distortion. In the calibration setting, forward-only FP8 quantizers give $D_{\mathrm{g}} = 0.0941$, or 0.090 averaged over the four 64-image batches of Figure 6, compared with 0.0016 for backward-only adjoint quantization and 0.0952 for the joint intervention. Forward quantizers therefore account for almost all of the all-low FP8 distortion. The dominant error enters through the combined E_k and the corresponding parameter-Jacobian readout.

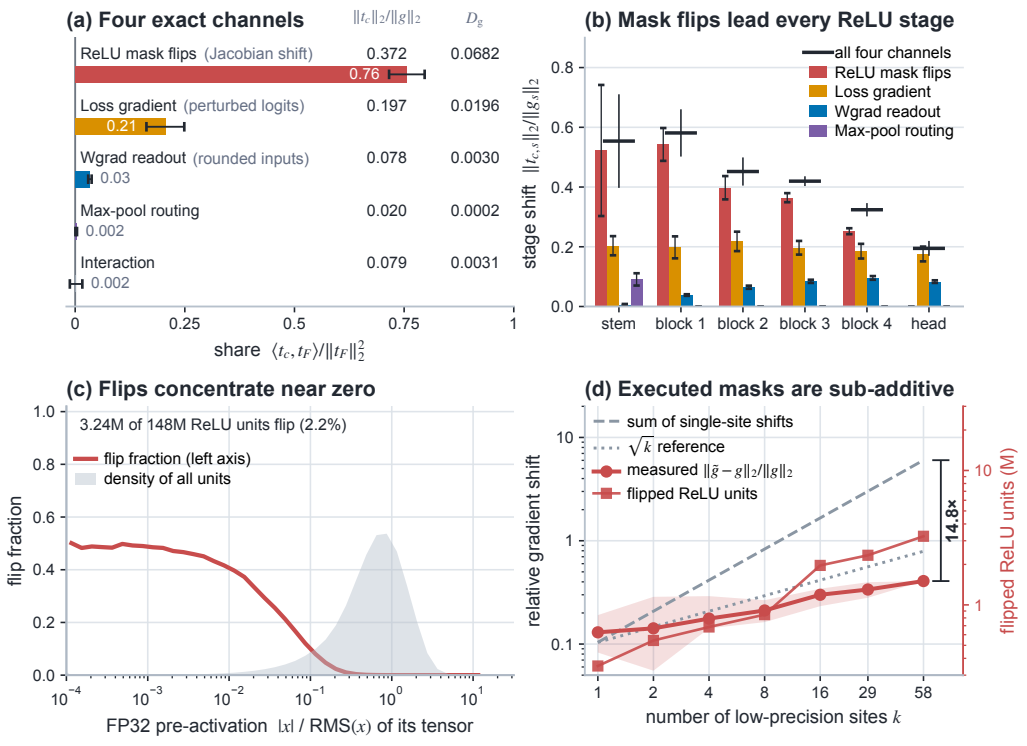


Figure 6: Four-channel replay of the forward-induced ResNet-18 rotation. Discrete rerouting of the backward pass dominates the FP8 rotation. (a) Channel shares of the FP8 gradient shift t_F. (b) Stage-wise shifts. (c) ReLU flip fraction against pre-activation magnitude. (d) Joint shift and flipped units over random subsets of k sites. Bars give the standard deviation over four batches, and the band in (d) spans the range over eight subsets per k.

Figure 6 resolves the forward-induced term into four replay channels. Flipped ReLU masks, the discrete part of the Jacobian shift $[J_k(\widetilde{h}_k) - J_k(h_k)]^\top$ of Appendix B.2, carry 76% of the shift and alone produce $D_g = 0.068$; the perturbed loss gradient carries 21%; the quantized saved inputs entering the Wgrad readouts carry 3%; and max-pool routing carries 0.2%. A local reconstruction score most directly reflects the third channel. Because flips are triggered by the total propagated noise, which grows sublinearly with the number of low-precision sites, single-site shifts over-count the joint effect by 14.8× in Figure 6(d). Because noise from the stem crosses every downstream ReLU boundary, this flip mechanism makes the stem sites of Figure 2 the most critical. The same mechanism explains why the selector must re-evaluate the executed mask after every pick.

Channel Replay Details. The channel shares quoted above arise as follows. In evaluation-mode ResNet-18, the FP32 backward graph depends on the realized forward state through four sets of objects: ReLU masks; max-pool routing indices; saved inputs used by convolution, BatchNorm, and classifier Wgrad readouts; and the terminal loss gradient. We record these objects from a forward-only all-low FP8-E4M3 quantizer on all 58 module sites, run on four fixed 64-image ImageNet batches with $D_g = 0.0902$, and replay each on the FP32 graph. Replaying all four together reproduces the FP8 gradient shift t_F to a relative error of 1.3×10^{-4}. Figure 6(a) reports the channel shares with the corresponding four-batch standard deviations, 0.76 ± 0.04 for mask flips and 0.21 ± 0.04 for the loss gradient, together with an interaction share of 0.002; alone, the mask channel gives $D_g = 0.068$ and the readout channel 0.003. Every channel is nearly orthogonal to the reference gradient: The mask channel has $\kappa^\perp = 0.37$ against $\kappa^\parallel = -0.06$, and the loss-gradient channel $\kappa^\perp = 0.20$ against $\kappa^\parallel = -0.02$, so the flips primarily rotate the gradient.

Figure 6(b) shows that mask flips dominate every stage with a downstream ReLU and vanish at the classifier. Quantization noise created upstream is propagated to ReLU boundaries, where the

noise flips units whose pre-activation is smaller than the arriving noise; each flip reroutes the adjoint. In Figure 6(c), 3.24 million of the 148 million ReLU units in a batch, or 2.2%, change state. The flip fraction is about 0.5 for pre-activations below 10^{-3} of the tensor RMS, decreases through the next two decades, and is below 0.01 in aggregate above 0.1 RMS. Thus, units with $|x| < 0.1$ RMS constitute fewer than 12% of the units but 74% of the flips.

In Figure 6(d), over random subsets of k of the 58 sites, the joint shift grows from 0.13 at $k = 1$ to 0.41 at $k = 58$, whereas the 58 single-site shift norms sum to 6.0, or 14.8× the joint value, and the single-site flip counts sum to 4.2× the joint count. The mask-flip channel is specific to networks with hard activation masks; in GPT-2, the forward-induced shift enters through the softmax, LayerNorm, and logits nonlinearities, and Figure 2(b) shows that the logits site dominates.

Precision-Mask Interactions. Precision-mask interaction measures whether two low-precision group choices distort the optimizer gradient additively or create extra coupled harm beyond the separate effects of the two choices. For a distortion metric D and a set S of low-precision groups,

$$I_{ij}^{\text{int}} \coloneqq D(\{i,j\}) - D(\{i\}) - D(\{j\}) + D(\emptyset).$$

Positive I_{ij}^{int} indicates super-additive policy harm. This scalar records the operational nonadditivity of the executed masks. When $D = D_{\text{g}}$, the scalar value can reflect parallel-component changes, the nonlinear cosine normalization, and the higher-order computational coupling recorded by $R_{>2}^{\theta}$; consequently, the sign of a D_{g} interaction alone does not determine the sign of the inner product Γ_{ij} of propagated contributions.

Single-mask distortions on the calibration batch are largest for normalization, activation, and convolution. The corresponding D_{g} values are 0.0542 for normalization, 0.0491 for activation, 0.0482 for convolution, 0.0173 for residual, and 0.0133 for classifier. The largest positive pairwise interaction occurs between the classifier and residual groups, with $I_{\text{classifier,residual}}^{\text{int}} = 0.0100$; the next largest are 0.0082 for classifier–convolution and 0.0074 for activation–classifier. Several other interactions are negative on this batch, indicating sub-additive angular distortion in the scalar D_{g} metric. This mask dependence helps explain why the final greedy set includes the inexpensive, strongly coupled classifier readout despite the smaller singleton distortion of the classifier readout.

These scalar interactions establish operational mask nonadditivity. Because these interactions are measured with the nonlinear D_{g} metric, the interaction signs do not determine the signs of Γ_{ij} or establish or refute submodularity of V: A negative D_{g} interaction can arise from vector cancellation, changes in the parallel component, or cosine normalization. This distinction motivates re-evaluating the executed mask after each pick.

Summary. Forward quantization causes almost all of the ResNet-18 FP8 bending, $D_{\text{g}} = 0.094$ against 0.0016 for backward-only quantization, and flipped ReLU masks carry 76% of the forward-induced shift. Because flips respond to the total propagated noise, the single-site shifts over-count the joint shift 14.8-fold, and mask pairs exhibit both super- and sub-additive angular distortion, so the benefit of a protected set cannot be inferred additively from single-mask scores.

D.5 Longer-Horizon Decoder-LM Continued Pretraining

Table 4 trains the three-group bending-guided masks of Table 10, at 70.2% and 60.8% of the eligible footprint under FP8 and NVFP4-style boundary quantizers, and the backward-high-precision QAF analogue with four times as many 256-token training sequences from WikiText-103. This experiment tests whether the main conclusion persists beyond the 8,192-sequence setting. The BF16 baseline improves to a validation loss of 3.169. The all-low FP8 policy remains severely degraded and reaches a loss of 7.228. Bending-guided precision allocation returns the validation loss to 3.170 and recovers essentially the whole gap. Under NVFP4-style boundary quantization, bending-guided precision allocation recovers 97.0% of the loss gap and reaches 60.2% last-token accuracy on the LAMBADA proxy. The backward-high-precision QAF analogue recovers only 58.9%.

Summary. With four times the training data of the decoder protocol, all-low FP8 and NVFP4-style boundary quantizers still degrade GPT-2 continued pretraining. The three-group bending-guided

Table 4: GPT-2 continued pretraining with four times the training data of the decoder protocol, 32,768 sequences. The QAF analogue applies the forward quantizers and returns every backward activation gradient to high precision. Last token is the LAMBADA last-token accuracy proxy of Appendix C.3, and gap closed is normalized within each precision setting; both are percentages.

Setup		Cost	Outcome			
Policy	Protected groups	Extra MiB	Val. loss	PPL	Last token	Gap closed
BF16 baseline	none	0.0	3.169	23.8	60.5	–
All-low FP8	none	0.0	7.228	1377.6	46.7	0.0
FP8 bending-guided	early norm, middle norm, LM head	91.6	3.170	23.8	60.5	100.0
All-low NVFP4	none	0.0	4.440	84.7	56.4	0.0
NVFP4 bending-guided	early attention, final norm, LM head	92.5	3.208	24.7	60.2	97.0
NVFP4 backward-high-precision QAF analogue	none	full backward	3.691	40.1	59.2	58.9

Table 5: FP8 and NVFP4 protection-budget curves at the pretrained GPT-2 weights, before continued pretraining. Gradient diagnostics are means over four selection calibration batches, and validation loss uses the 512-sequence validation split. Extra MiB follows the activation-cache cost model with $b_{\text{low}} = 1$ byte for FP8 and 0.5 byte for NVFP4, so the same group costs more under NVFP4.

Setup		Cost	Gradient	Outcome
Precision	Budget	Extra MiB	D_{g}	Val. loss
FP8	0	0.0	0.0826	5.921
	1	73.6	0.0029	3.954
	2	82.6	0.0016	3.948
	3	91.6	0.0012	3.945
	5	101.7	0.0008	3.944
NVFP4	0	0.0	0.4547	19.585
	1	85.9	0.0166	4.078
	2	87.2	0.0091	3.993
	3	92.5	0.0071	3.990
	5	113.5	0.0029	3.969

masks close 100.0% of the FP8 loss gap and 97.0% of the NVFP4-style gap, whereas the backward-high-precision QAF analogue closes 58.9% of the NVFP4-style gap, so selective protection remains more effective than a uniformly high-precision backward path.

D.6 Decoder-LM Protection Diagnostics

Table 5 gives the protection-budget curve measured before continued pretraining. A one-group protection already removes most of the low-precision gradient distortion, while larger budgets further reduce D_{g} and validation loss with diminishing returns after three groups. The validation losses in this table evaluate the policies at the pretrained weights, before further optimization. Table 10 reports the continued-pretraining results of the trained masks.

Summary. At the pretrained GPT-2 weights, protecting one group, the LM-head boundary, removes about 96% of the bending under both FP8 and NVFP4-style quantizers and lowers the validation loss from 5.92 to 3.95 under FP8 and from 19.6 to 4.08 under NVFP4-style quantizers. Larger budgets lower the bending further with diminishing returns after three groups, and the three-group masks also return the norm ratio and the descent preservation to within 2% of one.

Table 6: Full gradient diagnostics for the all-low and budget-3 bending-guided rows of Table 5.

Setup		Gradient			
Precision	Policy	D_{g}	P_{desc}	R_{norm}	$E_{\mathrm{rel}}^{\mathrm{param}}$
FP8	All-low	0.0826	1.0869	1.1849	0.4794
	Bending-guided	0.0012	1.0001	1.0013	0.0493
NVFP4	All-low	0.4547	1.1334	2.0990	1.7694
	Bending-guided	0.0071	1.0081	1.0153	0.1206

D.7 Budget-Matched Decoder Allocation

D.7.1 Allocation under a Memory Budget

Before training, the calibration probe predicts where protection starts to repair the gradient and where the gradient reaches the BF16 level, and training confirms both predictions; at matched memory, the masks of bending-guided precision allocation set the low-cost frontier of the final loss.

Decoder Continued Pretraining. Quantizing every eligible module-output activation and activation gradient severely degrades GPT-2 continued pretraining: Under FP8 boundary quantizers, the validation loss ends at 6.620 against 3.285 for BF16 training, and even without quantizers the learned weights reach only 4.544 (Table 10 in Appendix D.7). The candidate actions are 13 op-depth groups: the early, middle, and late four-block depth bands crossed with attention, MLP, and normalization operations, together with the token embedding, the position embedding, the final normalization, and the LM head, whose output is the logits. In calibration, every selector chooses groups under a budget stated as a share of the eligible activation footprint, and the score-based selectors take the exact knapsack optimum of the corresponding scores (Table 9); the trained masks of Table 7 are each rule's selections and masks along the bending-guided path, compared by the high-precision activation footprints the masks protect.

Before Training: Two Predicted Budgets. Group count does not reflect memory cost: The LM-head logits boundary alone is 56.4% of the eligible boundary footprint, because the 50,257-token vocabulary far exceeds the 768-dimensional model width. With the same site definition, the logits share falls to about 33% for GPT-2 Medium and 14% for GPT-2 XL and to 6% to 19% for Llama-shaped 7B to 8B models with 32k to 128k vocabularies, so this minimum viable budget shrinks with model width and depth. The calibration probe shows that protecting the logits boundary removes 96.8% of the all-low FP8 bending and 96.4% under NVFP4-style quantizers, whereas no FP8 protected set that omits the boundary, among all 8,192 op-depth sets, removes more than 12.3%; the minimum viable budget is therefore the LM-head cost. Local reconstruction error ranks the boundary 10th of 13, so a reconstruction-error knapsack spends every budget from 55% to 90% on the other twelve groups. Rules that weight quantization errors by the resulting downstream loss changes, namely loss-sensitivity, activation-Fisher, and finite-difference Hessian selection, also protect the LM-head boundary, as Eq. 4 predicts, and allocate different amounts of memory to the remaining groups.

Along the bending-guided path, the bending reaches the reference floor $\tau = 0.0034$, a rotation by 4.7°, at the LM-head cost under FP8 and at 74.6% of the footprint under NVFP4-style quantizers, with $D_{\mathrm{g}} = 0.0028$ in both cases, whereas under NVFP4-style quantizers the production-style prior of the token embedding, output head, and every normalization, at 78.9%, stays slightly above the floor with $D_{\mathrm{g}} = 0.0037$.

Calibration is inexpensive: One evaluation takes 0.14 seconds under FP8 and 0.41 seconds under NVFP4-style quantizers on one NVIDIA GeForce RTX 5090 Laptop GPU, so the three-group FP8 search, 36 evaluations, takes about 5 seconds, whereas training the same 36 masks would take about 2.5 GPU-hours.

After Training: Both Predictions Hold. Training confirms the minimum viable budget on every seed (Figures 3(a) and 3(b) and Table 7): Protecting all twelve groups other than the LM head, 43.6%

of the footprint, closes none of the FP8 loss gap and 20% of the NVFP4 gap, whereas the LM-head boundary alone closes 99.8% and 95.8%.

Training also confirms the target level (Figure 3(c)): Every trained mask whose calibration bending lies below τ learns weights whose loss without quantizers exceeds the loss of BF16 training by at most 0.003 on average over seeds, inside the 0.0037 seed-to-seed standard deviation of the BF16 runs, whereas every boundary mask above τ ends further above BF16, by 0.005 to 0.013 for the NVFP4-style masks that protect the LM head and by 0.67 to 1.26 for the masks that do not.

The probe also ranks the trained masks: Across the 18 trained boundary masks of both formats, calibration bending and the final loss gap left open have Spearman correlation $\rho = 0.95$, and over the 17 op-depth masks, bending ranks both the final gap and the unquantized loss at 0.95, and the update bending D_{u} ranks both at 0.97, whereas the summed local reconstruction error reaches only 0.35 and 0.30 and the summed loss sensitivity 0.69 and 0.70 (Table 12 in Appendix D.7). On these masks, whose bending comes mostly from forward quantization, bending agrees with the scores that weight quantization errors by the resulting downstream loss changes, activation-Fisher with correlation 0.98 and the executed-mask calibration loss with correlation 0.95; bending extends the same ranking to quantization that leaves the forward loss unchanged, which Appendix E.8.1 uses to allocate precision to individual GEMM passes.

Memory Savings over Hand-Designed Protection. Above the LM-head cost, bending-guided precision allocation attains the lowest or tied-lowest calibration bending of the six selectors at 13 of 14 budgets and matches the exact optimum at four of seven FP8 budgets. The bending-guided masks also set the low-cost frontier of the final loss: Under FP8, the module-level and two-group masks, at 60% to 63% of the footprint, achieve lower loss than the fixed op-type, loss-sensitivity, and Hessian masks, at 63% to 67%, on every seed, and the three-group mask at 70% ends 0.0007 above the BF16 loss, inside the seed-to-seed standard deviation of the BF16 runs, with 11% less footprint than the production-style prior.

The search recovers that prior from the calibration batches alone: At operation granularity, bending-guided precision allocation protects exactly the embedding, output head, and normalization boundaries (Table 8), and finer granularity trades more probe evaluations, still seconds of search, for cheaper masks, down to the module-level mask that closes 99.9% of the gap with 24% less footprint than that prior.

Under NVFP4-style quantizers, closure rises monotonically with footprint on every seed above the LM-head cost, and the bending-guided masks occupy the low-cost end of that frontier; the mask at the target level, at 74.6% of the footprint, learns weights within 0.003 of BF16 training, with 5% less footprint than the prior, whose learned weights end 0.005 above BF16.

Backward-Path Protection. Returning the whole backward path to high precision for the entire run while applying the forward quantizers, an analogue of the QAF phase of Chmiel et al. (2025), leaves the forward route, where the bending originates, in low precision. Under NVFP4-style boundary quantizers, backward-path protection leaves the calibration bending at 0.450 against 0.454 for the all-low policy, closes 61.7% and 45.5% of the loss gap on two seeds, and diverges on the third. With four times the token budget, the QAF analogue closes 58.9% of the gap, whereas the bending-guided mask at 60.8% of the footprint closes 97.0% (Table 4 in Appendix D.5), consistent with the forward-error mechanism in Sec. 2.

D.7.2 Full Calibration and Training Details

Groups differ widely in activation-cache cost, so this subsection compares selectors under a shared activation-cache budget. Every selector chooses op-depth groups under a budget stated as a fraction of the eligible boundary footprint, 130.4 MiB under FP8 and 152.2 MiB under NVFP4, using the four selection calibration batches of Table 5. Bending-guided precision allocation follows Listing 2 with $D = D_{\mathrm{g}}$, and the score-based baselines take the exact knapsack optima of the corresponding scores (Appendix C.1). Except for Table 8 in Appendix D.7.1, which comes from an earlier single-seed realization on the default GH200 setup of Appendix C.2, all results in this subsection and in Table 7 come from an independent realization on one NVIDIA GeForce RTX 5090 Laptop GPU with PyTorch 2.9.1 and Transformers 4.57.6. The laptop-GPU realization yields calibration D_{g} values that match

Table 7: GPT-2 continued pretraining under memory budgets, three seeds per mask. Share is the high-precision fraction of the eligible boundary footprint. Gap closure is mean ± standard deviation; asterisks mark calibration $D_{\mathrm{g}} < \tau = 0.0034$. ΔL_{unq} is the seed-paired excess unquantized loss over BF16, whose seed-to-seed standard deviation is 0.0037. †Ranges over seeds 17 and 31; seed 23 diverged.

Policy	Protected groups	Share (%)	Calib. D_{g}	Gap closed (%)	ΔL_{unq}
FP8 boundary quantizers					
All-low	none	0.0	0.0880	0.00	1.259
Reconstruction error	all but LM head	43.6	0.0955	-0.25 ± 1.16	1.260
Bending-guided	LM head	56.4	0.0028*	99.79 ± 0.02	0.002
Bending-guided	layer-2 norm, layer-3 norm, LM head	59.9	0.0015*	99.88 ± 0.02	0.000
Fixed op-type	late MLP, late attention, LM head	63.3	0.0026*	99.82 ± 0.01	0.002
Bending-guided	early norm, LM head	63.3	0.0015*	99.88 ± 0.05	0.000
Loss sensitivity	late norm, middle MLP, LM head	66.8	0.0025*	99.82 ± 0.03	0.001
Hessian	early attention, late norm, LM head	66.8	0.0030*	99.83 ± 0.02	0.001
Bending-guided	early norm, middle norm, LM head	70.2	0.0011*	99.98 ± 0.02	-0.001
Production-style prior	embedding, every norm, LM head	78.9	0.0006*	100.03 ± 0.03	0.000
NVFP4-style boundary quantizers					
All-low	none	0.0	0.454	0.00	0.673
Reconstruction error	all but LM head	43.6	0.414	20.29 ± 12.32	0.801
Bending-guided	LM head	56.4	0.0164	95.83 ± 0.63	0.013
Bending-guided	final norm, LM head	57.3	0.0089	96.78 ± 0.51	0.011
Bending-guided	early attention, final norm, LM head	60.8	0.0069	97.21 ± 0.41	0.008
Hessian	final norm, middle norm, LM head	64.2	0.0080	97.42 ± 0.27	0.009
Bending-guided	early attention, early norm, final norm, middle norm, LM head	74.6	0.0028*	98.53 ± 0.24	0.003
Production-style prior	embedding, every norm, LM head	78.9	0.0037	98.77 ± 0.23	0.005
Backward-path protection	whole backward path	–	0.450	45.5 to 61.7†	0.23 to 0.38†

Table 5 to within 3×10^{-4} for the shared protected masks and to within 7% for the all-low policies, and the greedy searches in this realization at the operation, op-depth, and module granularities return the protected sets of Table 8.

Minimum Viable Budget. The LM-head logits boundary alone is 56.4% of the eligible footprint, because the 50,257-token vocabulary far exceeds the 768-dimensional width of the other boundaries. At the pretrained weights, protecting this one boundary lowers the all-low calibration bending by 96.8% under FP8, from 0.0880 to 0.0028, and by 96.4% under NVFP4, from 0.454 to 0.016. The best other single group lowers the bending by only 1.3% under FP8 and 3.6% under NVFP4, and protecting all twelve other groups together, 43.6% of the footprint, leaves the bending at 0.0955 and 0.414. The exhaustive FP8 enumeration makes the threshold exact: Every one of the 8,192 op-depth protected sets that contains the LM-head boundary has $D_{\mathrm{g}} \leq 0.0147$, below every set that omits the

Table 8: Bending-guided FP8 protection at four grouping granularities, three greedy steps each, trained in a single-seed realization separate from Table 7, in which BF16 training reaches a validation loss of 3.240 and all-low FP8 6.708. Extra MiB is the added high-precision activation cache, and search evaluations count the masks that the greedy search executes.

Setup	Cost			Gradient	Outcome	
Granularity (candidates)	Search evaluations	Extra MiB	Share (%)	Calib. D_{g}	Val. loss	Gap closed (%)
Operation (6)	15	102.9	78.9	0.0007	3.241	100.0
Depth band (7)	18	109.6	84.0	0.0011	3.243	99.9
Op-depth (13)	36	91.6	70.2	0.0012	3.242	99.9
Module (40)	117	78.1	59.9	0.0016	3.245	99.9

boundary, for which $D_{\mathrm{g}} \geq 0.0771$. The calibration probe therefore identifies before training that no budget below the LM-head cost can repair the bending. The local reconstruction error ranks the LM-head boundary 10th of 13 under both formats, so a reconstruction-error knapsack spends any budget below the full footprint on the other twelve groups.

Reference Floor and Target-Level Budget. The reference floor τ of Sec. 5 is measured on the same four selection calibration batches: The FP32 gradient is computed without autocast and with TensorFloat-32 GEMMs disabled, and the BF16 gradient is the mixed-precision reference of the budget study. The two gradients differ by $D_{\mathrm{g}} = 0.0034$ on average, between 0.0031 and 0.0041 on single batches, with $R_{\mathrm{norm}} = 0.98$ and a zero-moment update bending of $D_{\mathrm{u}} = 0.035$ under the decoder recipe. Along the bending-guided path of Table 9, the FP8 bending falls below τ with the LM-head boundary alone, which gives $D_{\mathrm{g}} = 0.0028$ (Table 7), whereas the NVFP4-style bending stays above τ up to the 70% budget, with $D_{\mathrm{g}} = 0.0042$, and falls below the reference floor at 75%, with $D_{\mathrm{g}} = 0.0028$; the NVFP4-style production-style prior, at 78.9% of the footprint, produces gradient bending of $D_{\mathrm{g}} = 0.0037$. Every trained mask below τ, namely the eight FP8 masks that protect the LM head and the 74.6% NVFP4-style mask, learns weights whose loss without quantizers exceeds the seed-paired BF16 loss by at most 0.0032 on average over seeds, inside the 0.0037 seed-to-seed standard deviation of BF16 training (Table 10); the five NVFP4-style masks above τ that protect the LM head end 0.0046 to 0.0129 above BF16, and the masks without the LM head 0.67 to 1.26 above.

Allocation at Matched Budgets. At matched activation-cache budgets above the LM-head cost, bending-guided precision allocation attains the lowest or tied-lowest gradient bending on the calibration batches among the six selectors at 13 of the 14 budgets; at 90% of the FP8 footprint, activation-Fisher is lower by 3.4×10^{-5}. Under FP8, the allocation matches the minimum calibration D_{g} over all 8,192 op-depth protected sets at four of the seven feasible budgets and stays within 1.1×10^{-4} of that minimum at the other three. The greedy search thus nearly matches exhaustive selection in this comparison.

Activation-Fisher gives highest priority to the LM head and the normalization groups, so activation-Fisher matches the bending of our allocation at several budgets. The budgeted Hessian and loss-sensitivity baselines stop adding groups when the remaining groups have zero sensitivity scores. Under FP8, the calibration gradient bending D_{g} of the masks selected by these two baselines remains up to 4.3 and 2.7 times the minimum achievable at the same budget, respectively. The reconstruction-error baseline and the fixed random order allocate high precision to other groups first. Both selectors choose the LM-head output for high-precision protection only when the budget covers the entire eligible activation-cache footprint, leaving the logits in low precision at the smaller budgets (Table 9).

Budget Grid. Table 9 lists the full calibration grid behind this matched-budget comparison, at budgets from 40% to 90% of the eligible footprint. At the two budgets below the LM-head cost, 40% and 55%, every selector leaves the bending within 13% of the all-low value. On two unused held-out batches, the selected FP8 masks produce less held-out gradient bending than the mask that minimizes calibration bending at two of the three budgets where the two masks differ. At 80% of the

Table 9: Calibration D_g of every selector at matched footprint budgets on GPT-2. Budgets are percentages of the eligible boundary footprint, and the optimum, computed for FP8 only, is the exact minimum over all 8,192 op-depth protected sets. Values have three significant digits; for budgets above the LM-head cost of 56.4%, bold marks the lowest selector value, compared before rounding.

Setup			Selector						Oracle
Format	Budget (%)	Budget (MiB)	Bending-guided	Activation-Fisher	Hessian	Loss sensitivity	Recon. error	Random	Optimum
FP8	40	52.2	0.0860	0.0993	0.0880	0.0978	0.0982	0.0923	0.0772
	55	71.7	0.0860	0.0955	0.0880	0.0978	0.0955	0.0955	0.0772
	58	75.7	**0.00230**	**0.00230**	0.00281	**0.00230**	0.0955	0.0955	0.00230
	62	80.9	**0.00220**	0.00279	0.00281	0.00252	0.0955	0.0955	0.00220
	65	84.8	**0.00122**	**0.00122**	0.00252	0.00232	0.0955	0.0955	0.00122
	70	91.3	**0.00110**	**0.00110**	0.00252	0.00208	0.0955	0.0955	0.00107
	75	97.8	**0.000755**	0.000794	0.00252	0.00206	0.0955	0.0955	0.000755
	80	104.3	**0.000734**	0.000742	0.00252	0.00107	0.0955	0.0955	0.000627
	90	117.4	0.000641	**0.000606**	0.00252	0.00115	0.0955	0.0955	0.000590
NVFP4	40	60.9	0.423	0.417	0.417	0.425	0.448	0.416	–
	55	83.7	0.423	0.414	0.417	0.425	0.414	0.414	–
	58	88.3	**0.00869**	**0.00869**	0.00892	**0.00869**	0.414	0.414	–
	62	94.3	**0.00669**	**0.00669**	0.00926	0.00732	0.414	0.414	–
	65	98.9	**0.00669**	0.00780	0.00721	0.00756	0.414	0.414	–
	70	106.5	**0.00418**	0.00426	0.00640	0.00569	0.414	0.414	–
	75	114.1	**0.00280**	0.00313	0.00484	0.00515	0.414	0.414	–
	80	121.7	**0.00230**	0.00231	0.00246	0.00268	0.414	0.414	–
	90	137.0	**0.00106**	0.00128	0.00138	0.00250	0.414	0.414	–

footprint, the exact optimum is the production-style prior of the token embedding, output head, and every normalization boundary, which the operation-level search of Table 8 also returns.

Granularity and Search Cost. Table 8 repeats the three-action bending-guided search at four grouping granularities under the decoder continued-pretraining protocol of Appendix C.3. Every granularity protects the LM-head boundary and reaches near-BF16 loss. The operation-level search selects the embedding, output head, and normalization boundaries, a subset of the components to which production FP8 training allocates higher precision (DeepSeek-AI, 2024); the depth-band search selects the early and middle bands, the op-depth search the early and middle normalization groups, and the module-level search the normalizations of layers 2 and 3, each together with the LM head. The module-level mask closes 99.9% of the gap, against 100.0% for the operation-level mask, with 24% less high-precision footprint. Finer granularity costs more search evaluations, yet even the 40-candidate module-level search needs only 117 evaluations, fewer tokens than 12% of this short continued-pretraining run and a smaller relative cost for longer pretraining runs. The search also needs little calibration data: In the realization of Table 8, the three-group op-depth FP8 search returns the same mask from 1, 2, 4, 8, or 16 calibration batches.

Training at Matched Footprint. Table 10 trains the masks that separate the selectors at matched footprint, three seeds each, under the decoder continued-pretraining protocol of Appendix C.3; loss-gap closure is seed-paired against the BF16 baseline and the all-low policy of the same seed. Under FP8, protecting every group except the LM head leaves the gap open, whereas every mask that contains the LM-head boundary closes at least 99.7% of the gap on every seed, so the calibration probe predicts the minimum viable budget exactly.

Among the successful masks, the seed-paired loss differences are small but consistent. The module-level and two-group bending-guided masks, at 59.9% and 63.3% of the footprint, achieve lower loss than the fixed op-type, loss-sensitivity, and Hessian masks, at 63.3% to 66.8%, on all three seeds, by 0.0014 to 0.0021 on average. The three-group bending-guided mask, at 70.2%, achieves lower loss than the loss-sensitivity and Hessian masks by about 0.005 on every seed and ends 0.0007 above the BF16 loss; the production-style prior, at 78.9%, ends 0.0016 lower still, both inside the 0.0037 seed-to-seed standard deviation of the BF16 runs.

Under NVFP4, the calibration probe again predicts the minimum viable budget: Protecting every group except the LM head closes $20.3 \pm 12.3\%$ of the gap, whereas the LM-head boundary alone closes $95.8 \pm 0.6\%$, and every mask that contains the boundary closes at least 95% of the gap on every seed. Above that budget, the final loss falls monotonically with footprint on every seed across the bending-guided, Hessian, and production-style masks, so no trained mask is dominated in both footprint and loss; the bending-guided masks occupy the low-cost end of this frontier, closing 96.8% and 97.2% of the gap at 57.3% and 60.8% of the footprint and 98.5% at 74.6%.

The backward-high-precision QAF analogue, which leaves every forward boundary in NVFP4, closes 61.7% and 45.5% of the gap on seeds 17 and 31 and diverges on seed 23, where the final loss is 8.01 and the learned weights end 1.76 above BF16 training without quantizers; protecting the whole backward path therefore does not reliably repair the forward bending that the calibration probe measures, $D_{\mathrm{g}} = 0.450$ against 0.454 for the all-low policy.

The unquantized column separates the damage that training stores in the weights from the quantization applied at evaluation. In the seed means, every mask that contains the LM-head boundary ends within 0.013 of the BF16 unquantized loss, so most of each mask's remaining loss gap is evaluation-time quantization. The all-low FP8 and all-but-LM-head runs end at 4.54 without quantizers, above the pretrained weights at 3.944, and the quantized loss of the all-low FP8 run rose during training, from 5.92 at the pretrained weights (Table 5) to 6.62: Training along the rotated gradient undid part of the pretrained solution. The all-low NVFP4 run adapts to the NVFP4 quantizers, lowering the quantized loss from 19.6 to 4.71, yet the resulting learned weights end at 3.958 without quantizers, no better than the pretrained weights.

Prediction before Training. Figure 3(c) plots the calibration D_{g} of every trained boundary mask, measured before the first training step, against the excess loss of the learned weights evaluated without quantizers, with which calibration D_{g} has Spearman correlation 0.96 across the 18 masks of both formats. Across the same masks, the Spearman correlation between calibration D_{g} and the final gap left open is 0.95, and adding the median-seed result of the QAF analogue leaves the correlation at 0.95. The probe also ranks masks that all succeed: Among the masks that close more than 90% of the gap, calibration D_{g} and the final excess loss over BF16 have Spearman correlations of 0.83 for the eight FP8 masks and 0.89 for the six NVFP4 masks. One calibration evaluation takes 0.14 seconds under FP8 and 0.41 seconds under NVFP4, under 0.1% of one training run on the same GPU, so the probe can rank every candidate allocation before training.

Table 12 compares the probe with the other scores that are available before training, over the 17 trained op-depth masks on which every summed score is defined. Two outcomes are ranked: the final loss gap left open and the unquantized loss of the learned weights, which excludes the quantization applied at evaluation. The summed local reconstruction error, the objective of the reconstruction-error knapsack, ranks the LM-head boundary 10th of 13 groups and orders the outcomes with Spearman correlations of only 0.35 and 0.30, with intervals that include zero; the summed loss sensitivity reaches 0.69 and 0.70. Scores that account for the downstream effect of quantization errors order the outcomes at 0.92 to 0.98: bending, the relative gradient error and norm distortion, the executed-mask calibration loss, and the summed activation-Fisher and Hessian scores. In paired resamples of the masks, bending ranks the final gap better than the summed loss sensitivity in 99% of resamples and better than the summed local reconstruction error in 96%. The zero-moment AdamW update bending D_{u}, the quantity that the selector over individual GEMM passes of Appendix E.6 scores, ranks both outcomes at 0.97, so the gradient and update forms of the score agree on these masks. Unlike the loss- and curvature-based scores, which see only the forward quantization at each group, bending also scores Dgrad and Wgrad passes and the AdamW update (Appendix E.7), whose quantization leaves the forward loss unchanged.

Summary. Calibration identifies the LM-head cost, 56.4% of the eligible footprint, as the minimum viable budget, and training confirms this threshold on every seed: Protecting the other twelve groups closes none of the FP8 loss gap and about 20% of the NVFP4-style gap, whereas every mask with the LM head closes at least 99.7% and 95%. Masks below the BF16 reference floor learn weights within 0.0032 of the BF16 unquantized loss on average, and the three-group FP8 mask and the NVFP4-style target-level mask reach this level with 11% and 5% less high-precision footprint than the production-style prior. Calibration bending ranks the 17 trained op-depth masks by final gap with Spearman correlation 0.95, against 0.35 for the summed local reconstruction error.

Table 10: Budget-matched GPT-2 continued pretraining, three seeds per mask; share and calibration D_{g} are as in Table 7. Validation loss, under the policy's quantizers, and unquantized loss are means over seeds; gap closed is mean ± standard deviation, and the backward-path run diverges on one seed. Hessian denotes finite-difference Hessian selection.

Setup		Cost	Gradient	Outcome		
Policy	Protected groups	Share (%)	Calib. D_{g}	Val. loss	Unquantized loss	Gap closed (%)
FP8 boundary quantizers						
BF16 baseline	none	0.0	0	3.285	3.285	–
All-low	none	0.0	0.0880	6.620	4.544	0.00
Reconstruction error	all but LM head, 12 of 13 groups	43.6	0.0955	6.628	4.545	-0.25 ± 1.16
Bending-guided	LM head	56.4	0.0028	3.292	3.286	99.79 ± 0.02
Bending-guided	layer-2 norm, layer-3 norm, LM head	59.9	0.0015	3.289	3.285	99.88 ± 0.02
Fixed op-type	late MLP, late attention, LM head	63.3	0.0026	3.291	3.286	99.82 ± 0.01
Bending-guided	early norm, LM head	63.3	0.0015	3.289	3.285	99.88 ± 0.05
Loss sensitivity	late norm, middle MLP, LM head	66.8	0.0025	3.291	3.286	99.82 ± 0.03
Hessian	early attention, late norm, LM head	66.8	0.0030	3.290	3.286	99.83 ± 0.02
Bending-guided	early norm, middle norm, LM head	70.2	0.0011	3.285	3.284	99.98 ± 0.02
Production-style prior	embedding, every norm, LM head	78.9	0.0006	3.284	3.284	100.03 ± 0.03
NVFP4-style boundary quantizers						
All-low	none	0.0	0.454	4.714	3.958	0.00
Reconstruction error	all but LM head, 12 of 13 groups	43.6	0.414	4.414	4.086	20.29 ± 12.32
Bending-guided	LM head	56.4	0.0164	3.344	3.298	95.83 ± 0.63
Bending-guided	final norm, LM head	57.3	0.0089	3.330	3.296	96.78 ± 0.51
Bending-guided	early attention, final norm, LM head	60.8	0.0069	3.324	3.293	97.21 ± 0.41
Hessian	final norm, middle norm, LM head	64.2	0.0080	3.321	3.294	97.42 ± 0.27
Bending-guided	early attention, early norm, final norm, middle norm, LM head	74.6	0.0028	3.305	3.288	98.53 ± 0.24
Production-style prior	embedding, every norm, LM head	78.9	0.0037	3.302	3.289	98.77 ± 0.23
Backward-path protection	none; high-precision backward path	–	0.450	5.325	4.077	-55.64 ± 189.41

D.8 Decoder-LM Decomposition and Trajectory

Table 13 decomposes the initial GPT-2 calibration gradient by perturbation site. Forward activation quantization is the dominant source of gradient bending in this decoder-LM study. Quantizing only the forward activations to FP8 gives $D_{\mathrm{g}} = 0.0694$. This value is close to $D_{\mathrm{g}} = 0.0741$ when both forward activations and backward activation gradients are quantized; the 0.0741 value is measured on the calibration batches of the decomposition protocol, which differ from the selection batches that give 0.0826 in Table 6. Perturbing only the backward activation gradients or only the parameter gradients produces much smaller values.

The parameter-gradient rows are separate ablations that quantize the parameter gradients after back-propagation. The NVFP4-style boundary quantization shows the same ordering more strongly. A backward-high-precision QAF analogue is equivalent to the forward-only setting in this decomposi-

Table 11: Gradient bending accurately predicts the final-loss ranking before training. Spearman correlations between scores measured before training and the remaining validation-loss gap in GPT-2, three seeds per mask; larger values indicate closer agreement with the final-loss ranking. Memory-budget values cover the 17 op-depth boundary masks of Table 12 and compute-budget values the 16 masks of Table 28; forward-loss scores cannot distinguish interventions confined to Dgrad or Wgrad, so a dash replaces their compute-budget values.

Score before training	Memory budget	Compute budget
Summed local reconstruction error	0.35	0.50
Protected cost	0.78	0.59
Summed loss sensitivity	0.69	–
Summed activation-Fisher	0.98	–
Norm distortion S_{norm}	0.93	0.77
Gradient bending D_{g}	0.95	0.94
AdamW update bending D_{u}	0.97	0.92

Table 12: Spearman correlations, with 95% bootstrap intervals over masks, between scores measured before training and the outcomes of the 17 op-depth masks of Table 10. Summed scores add the per-group scores of the groups left in low precision, the other scores are computed from each executed mask, and D_{u} uses zero-moment AdamW updates of the decoder recipe.

Setup	Spearman correlation with outcome			
Score before training	Gap		Unquantized loss	
Summed local reconstruction error	0.35	$[-0.29, 0.96]$	0.30	$[-0.33, 0.94]$
Summed loss sensitivity	0.69	$[0.23, 0.91]$	0.70	$[0.26, 0.90]$
Summed finite-difference Hessian	0.93	$[0.75, 0.99]$	0.92	$[0.73, 0.97]$
Summed activation-Fisher	0.98	$[0.88, 1.00]$	0.97	$[0.82, 1.00]$
Executed-mask calibration loss	0.95	$[0.80, 0.98]$	0.93	$[0.77, 0.98]$
Protected footprint	0.78	$[0.41, 0.96]$	0.77	$[0.39, 0.94]$
Norm distortion S_{norm}	0.93	$[0.74, 0.98]$	0.93	$[0.74, 0.98]$
Relative gradient error $E_{\mathrm{rel}}^{\mathrm{param}}$	0.95	$[0.81, 0.99]$	0.95	$[0.80, 1.00]$
Bending D_{g}	0.95	$[0.79, 0.99]$	0.95	$[0.79, 1.00]$
Update bending D_{u}	0.97	$[0.86, 1.00]$	0.97	$[0.82, 1.00]$

tion. This equivalence explains why the QAF analogue improves the all-low NVFP4 training outcome but does not close the gap in Tables 7 and 4.

Table 14 tracks the same diagnostics during the continued pretraining trajectory. The optimizer gradients under the all-low FP8 and NVFP4 policies are badly distorted at initialization and remain so after training has moved the model. Bending-guided precision allocation yields D_{g} below 0.004 for FP8 and near 0.024 for NVFP4 at the final weights. Bending-guided precision allocation also returns the validation PPL to the BF16 baseline range. The NVFP4 backward-high-precision QAF analogue reduces final D_{g} relative to the all-low NVFP4 policy. However, the QAF analogue still leaves a much larger geometry error and validation loss than selective protection, consistent with the forward-induced decomposition above.

The Optimizer Receives Only the Direction. Common training recipes further reduce sensitivity to gradient scale. AdamW is invariant to a constant positive rescaling of the gradient history up to AdamW's ϵ term. The decoder continued-pretraining and vision fine-tuning recipes also clip gradients at unit norm, and every logged step in these recipes exceeds that threshold. Norm changes can thus be addressed through learning rate adjustment, clipping, or normalization. The allocation rule allocates precision to reduce bending, while both coordinates describe the all-low policy's gradient error.

The decoder continued-pretraining and vision fine-tuning recipes clip the global gradient norm at 1.0, and in every run of Tables 10 and 18 the pre-clip norm exceeded that threshold at every optimizer step. AdamW therefore received $\widetilde{g}_\theta / \|\widetilde{g}_\theta\|_2$, so clipping removed differences in gradient norm between policies before the update. Despite this normalization, the all-low FP8 run degraded the pretrained solution. During training under all-low FP8, the validation loss measured with the quantizers rose

Table 13: GPT-2 perturbation-site decomposition before continued pretraining. Rows isolate forward activations, backward activation gradients, post-backward parameter gradients, and a first-step AdamW update proxy. This table and Table 14 use a separate instrumented 8,192-sequence realization and calibration batches, so the values in the two tables need not match the selection results in Tables 5 and 6.

Setup		Calibration	Gradient			
Precision	Setting	Calib. loss	D_{g}	P_{desc}	R_{norm}	$E_{\mathrm{rel}}^{\mathrm{param}}$
FP8	QAF	5.644	0.0694	1.098	1.180	0.442
	Backward activation gradients only	3.652	0.0132	1.005	1.019	0.164
	Parameter gradients only	3.652	0.0004	1.003	1.004	0.027
	Forward + backward activation gradients	5.644	0.0741	1.105	1.194	0.463
	Forward + backward + parameter gradients	5.644	0.0743	1.103	1.192	0.462
	Bending-guided	3.651	0.0011	0.997	0.998	0.047
	Update proxy	3.652	0.0001	1.006	1.006	0.014
NVFP4	QAF	19.498	0.4478	1.191	2.190	1.844
	Backward activation gradients only	3.652	0.0051	1.002	1.007	0.098
	Parameter gradients only	3.652	0.0004	1.000	1.000	0.028
	Forward + backward activation gradients	19.498	0.4501	1.186	2.189	1.846
	Forward + backward + parameter gradients	19.498	0.4502	1.185	2.189	1.846
	Bending-guided	3.698	0.0066	1.003	1.010	0.115
	Update proxy	3.652	0.0006	1.007	1.008	0.036

Table 14: GPT-2 trajectory under the instrumented protocol introduced in Table 13. Initialization and final-step diagnostics use each checkpoint's FP32 gradient; $\|g_\theta^{(1024)}\|_2$ is the final reference-gradient norm. The outcome tables use independent training realizations, so the final losses in the outcome tables can differ slightly.

Setup	Gradient				Outcome	
Policy	$D_{\mathrm{g}}^{(0)}$	$D_{\mathrm{g}}^{(1024)}$	$E_{\mathrm{rel}}^{\mathrm{param},(1024)}$	$\|g_\theta^{(1024)}\|_2$	Val. loss	PPL
All-low FP8	0.0741	0.6814	1.011	21.02	6.651	773.4
FP8 bending-guided	0.0011	0.0034	0.083	4.42	3.239	25.5
All-low NVFP4	0.4501	0.6161	1.085	8.90	4.570	96.5
NVFP4 bending-guided	0.0066	0.0240	0.222	4.32	3.279	26.5
NVFP4 backward-high-precision QAF analogue	0.4478	0.3067	0.755	6.13	4.383	80.1

from 5.92 at the pretrained weights to 6.62, and the learned weights evaluated without quantizers reach 4.54, worse than the pretrained 3.944, whereas BF16 training reaches 3.285 (Tables 5 and 10); along the instrumented all-low FP8 trajectory of Table 14, bending grows from 0.074 to 0.68.

Summary. Forward activation quantization causes most of the initial GPT-2 bending, $D_{\mathrm{g}} = 0.069$ of 0.074 under FP8, so the backward-high-precision QAF analogue keeps nearly all of the all-low bending and does not close the loss gap. Clipping removes the norm differences between policies at every step, yet the all-low FP8 bending grows from 0.074 to 0.68 along the training trajectory and the learned weights end worse than the pretrained weights, whereas the bending-guided masks keep D_{g} below 0.004 under FP8 and near 0.024 under NVFP4 at the final weights.

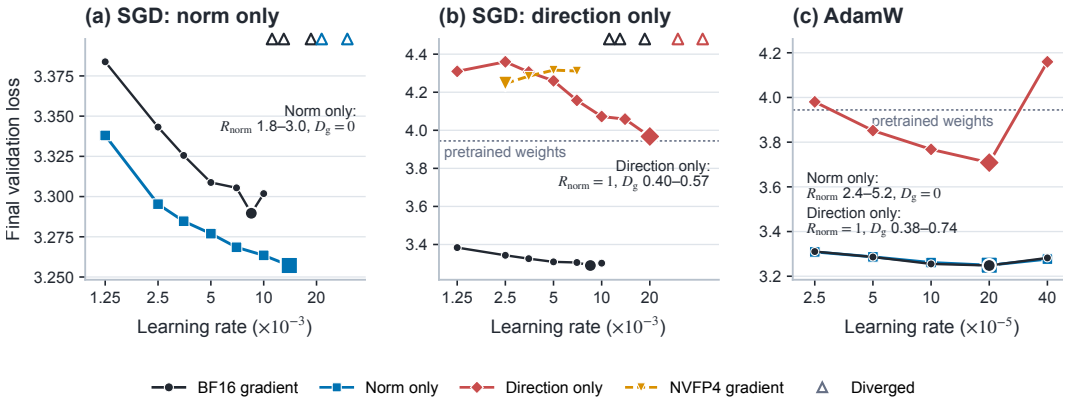


Figure 7: Learning rate tuning absorbs norm changes but not bending in GPT-2 continued pretraining. The optimizer receives the BF16 gradient, the NVFP4 gradient under all-low NVFP4-style boundary quantizers, a norm-only intervention that combines the BF16 direction with the NVFP4 gradient norm, or a direction-only intervention that combines the NVFP4 direction with the BF16 gradient norm; figure labels give the run-mean R_{norm} and D_{g} of the supplied vector. Momentum SGD (a, b) and AdamW (c) run without clipping, and final losses are unquantized. Curves show one seed; large markers mark each curve's best learning rate.

D.9 Learning Rate Tuning Cannot Undo Bending

The gradient interventions of Figure 7 test Eq. 10 in training by changing the gradient norm or direction separately. Each run starts from the pretrained GPT-2 weights and follows the decoder continued-pretraining protocol of Appendix C.3, one pass over 8,192 WikiText-103 sequences of 256 tokens with batch 8 and a linear schedule with 3% warmup, but changes the optimizer and the vector the optimizer receives.

At every step, a BF16 pass computes the reference gradient g_θ, and a second pass under the all-low NVFP4 policy computes $\widetilde{g}_\theta$ on the same batch with the same dropout masks. The optimizer then receives the BF16 gradient g_θ; the norm-only intervention $(\|\widetilde{g}_\theta\|_2/\|g_\theta\|_2)g_\theta$, which combines the reference direction with the NVFP4 gradient norm; the direction-only intervention $(\|g_\theta\|_2/\|\widetilde{g}_\theta\|_2)\widetilde{g}_\theta$, which combines the reference gradient norm with the NVFP4 direction; or the unmodified NVFP4 gradient $\widetilde{g}_\theta$. Quantizers act only in the second pass, so the final validation loss, evaluated without quantizers on the 512 validation sequences, measures only what the optimizer did with the supplied vectors.

Momentum SGD, with momentum 0.9 and no weight decay, runs on a learning rate grid from 1.25×10^{-3} to 4×10^{-2}, mostly spaced by factors of about $\sqrt{2}$ and refined near the BF16 optimum, and AdamW, with weight decay 0.01, runs at half to eight times the recipe learning rate of 5×10^{-5}; neither optimizer clips the gradient.

Seed 17 covers each grid, two further seeds repeat BF16 and the norm-only and direction-only arms at the learning rate that is best on the grid, and every intervention run records D_{g} and R_{norm} between g_θ and $\widetilde{g}_\theta$ every 32 steps along the trajectory of that run. The runs use one NVIDIA GeForce RTX 5090 Laptop GPU with PyTorch 2.9.1, and an intervention run takes about five minutes.

During stable norm-only training, the measured NVFP4 gradients have norms averaging 1.8 to 5.2 times the BF16 gradient norms and bending $D_{\text{g}} = 0.48$ to 0.80. The norm-only setting retains these large norm changes but replaces the rotated direction with the BF16 direction.

At the best rate, the AdamW norm-only loss is 3.249 ± 0.001 over three seeds, compared with 3.248 ± 0.001 for BF16. For momentum SGD, training in the norm-only setting remains stable at a learning rate where BF16 diverges and achieves 3.259 ± 0.002 at the best tuned rate, compared with 3.292 ± 0.004 for BF16. Under SGD, all final losses in the direction-only setting exceed the pretrained loss of 3.944.

Table 15 lists every grid run and the three-seed results. At the first training step, the all-low NVFP4 gradient has a norm 2.8 to 3.1 times the BF16 gradient norm and gradient bending of $D_{\mathrm{g}} = 0.53$ to 0.57 across the three seeds, measured on training batches with dropout active, whereas the calibration batches of Table 6 give 2.10 and 0.45. Along the stable norm-only runs, the NVFP4 gradient's run-mean norm ratio is 1.8 to 3.0 under SGD and 2.4 to 5.2 under AdamW, with run-mean bending of 0.48 to 0.80, and within a run the norm ratio varies with a CV of 0.10 to 0.19.

Under momentum SGD, BF16 training reaches the lowest BF16 loss, 3.290, at a learning rate of 0.0085 and diverges from 0.012. The norm-only runs supply gradients whose norms are on average 1.8 to 3.0 times the BF16 gradient norms and reach the lowest norm-only loss, 3.257, at 0.014 and diverge from 0.02. A constant gradient scaling factor would move the best rate and the divergence point of BF16 down by that factor; because the NVFP4 gradient norm varies from step to step, the norm-only runs remain stable at 0.014, where BF16 diverges, although the gradients supplied in the norm-only runs at that learning rate have norms averaging 1.8 to 1.9 times the BF16 gradient norms across the three seeds, so the norm-only intervention introduces no loss penalty at the tuned norm-only learning rate.

On the seed-17 grid, the direction-only intervention produces final losses between 3.967 and 4.360 across all stable learning rates, all above the pretrained loss of 3.944, and training with the unmodified NVFP4 gradient produces final losses between 4.246 and 4.316. Under AdamW, the norm-only intervention achieves final losses within 0.007 of BF16 at all five learning rates, although the NVFP4 gradient norm averages up to 5.2 times the BF16 gradient norm, whereas the direction-only intervention produces final losses 0.46 to 0.88 above BF16 at every rate.

Two further seeds at the best learning rate of the grid confirm the ordering: The BF16, norm-only, and direction-only settings yield final losses of 3.292 ± 0.004, 3.259 ± 0.002, and 3.973 ± 0.012 under SGD and 3.248 ± 0.001, 3.249 ± 0.001, and 3.645 ± 0.055 under AdamW. Along the stable direction-only trajectories, the NVFP4 gradient norm falls below the BF16 gradient norm, with run-mean norm ratio 0.69 to 0.88, while the NVFP4 gradient's run-mean bending stays between 0.38 and 0.74 and single steps exceed $D_{\mathrm{g}} = 1.3$, beyond a right angle, so directional error accumulates throughout these trajectories.

Summary. Training separates the two coordinates. With the BF16 direction and the norm of the all-low NVFP4 gradient, AdamW achieves a final loss within 0.007 of BF16 at every learning rate of the grid, and momentum SGD, whose supplied gradient norms then vary from step to step, achieves a lower final loss than BF16, 3.259 ± 0.002 against 3.292 ± 0.004 over three seeds. With the NVFP4 direction and the BF16 gradient norm, the final loss at the best tested learning rate is 0.68 above BF16 under SGD and 0.40 above under AdamW, and every direction-only SGD run ends above the pretrained loss.

D.10 Quantization Error Repeats across Batches

Mini-batch sampling introduces variation in the gradient even without quantization. Our probe compares reference and low-precision gradients on the same batch, isolating the additional error from the precision policy. This error can contain a component that persists across batches. Table 16 measures how much of the error repeats from batch to batch on the first 16 gradient batches of the decoder study, 2×256 tokens each at the pretrained GPT-2 weights, of which the first four are the selection calibration batches of Table 5. For the error $\xi_b \coloneqq \widetilde{g}_\theta^{(b)} - g_\theta^{(b)}$ of batch b, the repeat ratio $\mathrm{Rep} \coloneqq \|\bar{\xi}\|_2^2 / \overline{\|\xi_b\|_2^2}$, with the bars denoting means over batches, equals 1 when every batch carries the same error and about $1/16$ when the errors are independent and zero-mean; $\mathrm{Rep}_\perp$ applies the same ratio to the part of ξ_b orthogonal to $g_\theta^{(b)}$, the part that rotates the gradient, and the same ratio for the BF16 reference gradient, 0.48, measures how much of the sampled gradient is common to all batches. The last two columns compare the mean per-batch bending with the bending between the batch-averaged reference and policy gradients.

Under all-low boundary quantization, the error repeats at least as consistently as the sampled BF16 gradient, $\mathrm{Rep} = 0.52$ under FP8 and 0.79 under NVFP4-style quantizers against 0.48, and averaging over the 16 batches does not reduce the bending: The averaged gradients have bending values of 0.122 and 0.565, against per-batch means of 0.116 and 0.480. The persistent component provides a mechanism for accumulated optimization error, consistent with the direction-only runs

Table 15: Final validation loss, evaluated without quantizers, of the runs with gradient interventions in Figure 7. (a, b) Learning rate grids with seed 17; bold marks the lowest loss of each row, div. denotes a diverged run, and a dash marks a rate that was not run. (c) Three seeds, including seed 17, at the best grid learning rate.

(a) Momentum SGD

	Learning rate ($\times 10^{-3}$)											
Gradient	1.25	2.5	3.5	5	7	8.5	10	12	14	20	28	40
BF16 gradient	3.384	3.343	3.326	3.309	3.305	**3.290**	3.302	div.	div.	div.	–	–
Norm only	3.338	3.295	3.285	3.277	3.268	–	3.263	–	**3.257**	div.	div.	–
Direction only	4.310	4.360	4.306	4.260	4.157	–	4.072	–	4.059	**3.967**	div.	div.
NVFP4 gradient	–	**4.246**	4.284	4.316	4.312	–	–	–	–	–	–	–

(b) AdamW

	Learning rate ($\times 10^{-5}$)				
Gradient	2.5	5	10	20	40
BF16 gradient	3.310	3.286	3.255	**3.248**	3.282
Norm only	3.309	3.287	3.261	**3.249**	3.277
Direction only	3.980	3.852	3.768	**3.708**	4.160

(c) Three seeds at the best learning rate

Optimizer	Gradient	Best learning rate	Final loss
Momentum SGD	BF16 gradient	0.0085	3.292 ± 0.004
Momentum SGD	Norm only	0.014	3.259 ± 0.002
Momentum SGD	Direction only	0.02	3.973 ± 0.012
AdamW	BF16 gradient	2×10^{-4}	3.248 ± 0.001
AdamW	Norm only	2×10^{-4}	3.249 ± 0.001
AdamW	Direction only	2×10^{-4}	3.645 ± 0.055

Table 16: Repeat ratio Rep of the quantization error over 16 GPT-2 gradient batches at the pretrained weights, which is 1 for an error that repeats on every batch and approximately $1/16$ for independent zero-mean errors, against 0.48 for the sampled BF16 gradient. The GEMM-core measurement records Rep only.

Setting	Policy	Rep	$\mathrm{Rep}_\perp$	Mean D_g	Averaged D_g
FP8 boundary quantizers	All-low	0.52	0.52	0.116	0.122
	LM head protected	0.07	0.08	0.0028	0.0004
NVFP4-style boundary quantizers	All-low	0.79	0.79	0.480	0.565
	LM head protected	0.38	0.32	0.0187	0.0113
NV GEMM core	All-low	0.18	–	0.0265	0.0099
	Backward path in BF16	0.18	–	0.0259	0.0097
	Target-level mask	0.17	–	0.0033	0.0012
	Fprop path in BF16	0.19	–	0.0017	0.0007

of Appendix D.9. Protection removes the repeating part. With the LM-head boundary protected, the remaining FP8 error has $\mathrm{Rep} = 0.07$, close to independent noise, and averaging lowers the bending from 0.0028 to 0.0004; under NVFP4-style quantizers, part of the remaining error still repeats, $\mathrm{Rep} = 0.38$, consistent with the 4% of the loss gap that the NVFP4-style LM-head mask leaves open (Table 10).

In the NV GEMM core, the error of every mask repeats less than the sampled BF16 gradient, $\mathrm{Rep} = 0.17$ to 0.19, yet about three times more than independent noise, and the averaged gradients still exhibit 36% to 39% of the per-batch bending.

Summary. Quantization error contains a substantial component shared across batches. Under all-low boundary quantization, the error repeats across 16 GPT-2 batches at least as consistently

Table 17: Four-epoch CIFAR-100 full-parameter fine-tuning of ImageNet-pretrained ResNet-18, with five runs per policy. Share is the high-precision fraction of the eligible boundary footprint on the ImageNet calibration batch used for selection.

Setup		Cost	Outcome		
Policy	Protected groups	Share (%)	Top-1 (%)	Top-5 (%)	Loss
FP32	none	0.0	69.55 ±0.33	93.47 ±0.15	1.1955 ±0.0072
All-low BF16	none	0.0	69.53 ±0.34	93.47 ±0.12	1.1956 ±0.0073
All-low FP8	none	0.0	68.53 ±0.34	93.06 ±0.07	1.2260 ±0.0053
FP8 bending-guided, unit cost	classifier, convolution, normalization	61.9	69.35 ±0.26	93.31 ±0.05	1.2050 ±0.0071
FP8 reconstruction error	activation, classifier, residual	38.1	68.92 ±0.27	93.19 ±0.08	1.2145 ±0.0052

as the sampled gradient, and averaging the gradients does not reduce the bending; protecting the LM-head boundary leaves an FP8 error close to independent noise, whose bending averages out. In the NV GEMM core, the error repeats less than the sampled gradient but about three times more than independent noise.

D.11 Transfer to Vision Fine-Tuning

Protected sets chosen from a calibration gradient transfer to CIFAR-100 fine-tuning and recover 56% to 104% of the FP8 top-1 gap on four backbones. For ResNet-18, the sets are chosen on ImageNet: In four-epoch fine-tuning, BF16 boundary quantizers match the FP32 top-1 accuracy of 69.55%, and all-low FP8 falls 1.02 percentage points (pp) behind (Table 17 in Appendix D.12). Bending-guided precision allocation with unit cost, which allocates high precision to 62% of the eligible boundary footprint, recovers 0.82 pp, or 80% of the gap, and reconstruction-error selection, at 38% of the footprint, recovers 0.39 pp.

The pattern holds across architectures. In six-epoch fine-tuning (Table 18 in Appendix D.13), all-low FP8 leaves seed-paired top-1 gaps of 0.44, 0.29, and 0.24 pp on ConvNeXt-T, Swin-T, and ViT-B/16, respectively. Bending-guided precision allocation with cache cost recovers 59%, 56%, and 104% of those gaps while using high precision for only 14%, 25%, and 11% of the eligible boundary footprint.

D.12 ResNet-18 Transfer Fine-Tuning Details

Table 17 reports the five-run ResNet-18 transfer results. The all-low FP8 policy loses 1.02 top-1 pp relative to FP32. The ImageNet-selected unit-cost bending-guided precision allocation, at 61.9% of the eligible boundary footprint, recovers 0.82 pp, or about 80% of the top-1 drop, and the reconstruction-error selection, at 38.1%, recovers 0.39 pp. BF16 boundary quantizers match FP32, isolating FP8 precision as the source of the observed degradation relative to the BF16 boundary-quantizer control.

As a frozen-backbone linear-probe check, we also train the classifier for three epochs with the ImageNet-pretrained ResNet-18 backbone fixed. FP32 reaches a top-1 accuracy of 53.10%, the all-low FP8 policy reaches 52.98%, and a fixed protected set of activation, normalization, and residual groups reaches 53.25%. This frozen-backbone check shows that the simulated-precision quantizers can be applied while preserving a nontrivial pretrained feature baseline.

Summary. In ResNet-18 CIFAR-100 transfer, BF16 boundary quantizers match FP32, whereas all-low FP8 loses 1.02 top-1 pp. The unit-cost bending-guided precision allocation selected on ImageNet recovers 0.82 pp, about 80% of that drop, at 61.9% of the eligible boundary footprint, against 0.39 pp for reconstruction-error selection at 38.1%.

Table 18: Six-epoch CIFAR-100 fine-tuning of ImageNet-pretrained backbones, five runs per policy for ConvNeXt-T and three for Swin-T and ViT-B/16. Deltas are seed-paired and computed before rounding, recovery is the fraction of the all-low FP8 gap that protection closes, and share is the mean extra cache relative to the eligible boundary footprint of 9,937, 10,570, and 12,743 MiB.

Setup		Outcome			Cost	
Backbone	Policy	Top-1 (%)	Δ from FP32 (pp)	Recovery fraction	Extra cache (MiB)	Share (%)
ConvNeXt-T	FP32	86.71 ±0.12	0.00	–	0.00	0.0
	All-low FP8	86.27 ±0.06	−0.44 ±0.15	0.00	0.00	0.0
	FP8 bending-guided, cache cost	86.53 ±0.14	−0.18 ±0.17	0.59	1356.39	13.7
Swin-T	FP32	85.66 ±0.15	0.00	–	0.00	0.0
	All-low FP8	85.36 ±0.24	−0.29 ±0.08	0.00	0.00	0.0
	FP8 bending-guided, cache cost	85.53 ±0.19	−0.13 ±0.05	0.56	2687.38	25.4
ViT-B/16	FP32	88.72 ±0.09	0.00	–	0.00	0.0
	All-low FP8	88.48 ±0.06	−0.24 ±0.06	0.00	0.00	0.0
	FP8 bending-guided, cache cost	88.73 ±0.18	0.01 ±0.10	1.04	1366.63	10.7

D.13 Modern Backbone Fine-Tuning Details

In the modern-backbone results, protected sets are selected per seed over operation-family groups. Stem denotes the initial patch or convolutional stem, attention denotes attention blocks, MLP denotes feed-forward blocks, and residual, normalization, convolution, activation, and classifier use the same group meanings as in the ResNet-18 experiments.

Table 18 reports the six-epoch modern-backbone comparison. The FP32 baselines reach $86.71 \pm 0.12\%$ for ConvNeXt-T, $85.66 \pm 0.15\%$ for Swin-T, and $88.72 \pm 0.09\%$ for ViT-B/16. Using the rounded top-1 means, the all-low FP8 policy falls behind by about 0.44, 0.30, and 0.24 top-1 pp, respectively. The seed-paired delta column gives the corresponding all-low FP8 gaps as 0.44 ± 0.15, 0.29 ± 0.08, and 0.24 ± 0.06 pp. Cache-cost bending-guided precision allocation closes these gaps to 0.18 ± 0.17, 0.13 ± 0.05, and -0.01 ± 0.10 pp, recovering 59%, 56%, and 104% of the all-low FP8 drops while using high precision for 13.7%, 25.4%, and 10.7% of the eligible boundary footprint.

Summary. On ConvNeXt-T, Swin-T, and ViT-B/16 CIFAR-100 fine-tuning, all-low FP8 leaves small seed-paired top-1 gaps of 0.24 to 0.44 pp, and cache-cost bending-guided precision allocation recovers 56% to 104% of these accuracy gaps while using high precision for 11% to 25% of the eligible boundary footprint.

D.14 Actual-Kernel FP8 GEMM Microbenchmarks

Table 19 presents results from an actual-kernel FP8 GEMM microbenchmark. The simulated-precision FP8 row applies the diagnostic quantizer and then calls ordinary FP32 matmul, while the actual-kernel rows use the corresponding scaled-matmul path. Across the measured shapes and output dtypes, the actual-kernel FP8-E4M3 implementations achieve 26 to 32 times the throughput of the simulated path while preserving the same relative output error.

Table 19 confirms that the measured FP8 path gives the expected GEMM speedup at the same relative output error as the simulated quantizer. Appendix E.11 embeds the same actual-kernel E4M3

Table 19: Actual-kernel FP8 GEMM microbenchmark. Simulated-precision FP8 quantizes both operands before ordinary FP32 matmul, and actual-kernel rows use FP8-E4M3 operands with scaled matmul. Throughput is in tera floating-point operations per second (TFLOP s^{-1}), relative output error is $\|Y_{\mathrm{policy}} - Y_{\mathrm{FP32}}\|_F / (\|Y_{\mathrm{FP32}}\|_F + 10^{-12})$, and peak memory is the maximum allocated GPU memory.

Setup		Kernel metrics			
GEMM $M \times N \times K$	Implementation	TFLOP s^{-1}	Rel. output err.	Latency (ms)	Peak memory (MiB)
$4096 \times 4096 \times 4096$	FP32 matmul	50.9	0.0000	2.700	352
	BF16 matmul	474.5	0.0029	0.290	352
	FP16 matmul	493.7	0.0004	0.278	352
	Simulated-precision FP8 + FP32 matmul	43.0	0.0375	3.195	480
	Actual-kernel FP8-E4M3, BF16 output	1351.1	0.0375	0.102	321
	Actual-kernel FP8-E4M3, FP32 output	1134.8	0.0375	0.121	385
$8192 \times 4096 \times 4096$	FP32 matmul	51.0	0.0000	5.392	608
	BF16 matmul	566.8	0.0029	0.485	576
	FP16 matmul	550.9	0.0004	0.499	576
	Simulated-precision FP8 + FP32 matmul	44.6	0.0375	6.164	800
	Actual-kernel FP8-E4M3, BF16 output	1442.5	0.0375	0.191	529
	Actual-kernel FP8-E4M3, FP32 output	1174.0	0.0375	0.234	657

scaled-matmul path in every GPT-2 block for full pretraining trajectories, where the scaled-matmul path serves as the accuracy and geometry anchor for the simulated FP4 study.

Summary. Actual-kernel FP8-E4M3 scaled GEMMs match the relative output error of the simulated-precision FP8 quantizer, 0.0375, at 26 to 32 times the throughput of the simulated path and, with BF16 output, at 2.54 to 2.85 times the BF16 matmul throughput, the ratios that Appendix E.7 uses to model GEMM time.

E Controlled FP4 Cores and Optimizer-Update Geometry

This appendix reports controlled numerical experiments with the simulated MX and NV cores, separating four distinct experimental objects: format arithmetic, computation pass, propagated gradient error, and the AdamW update (Rouhani et al., 2023; Tseng et al., 2025; Castro et al., 2025; Chmiel et al., 2025; NVIDIA, 2025). The experiments comprise paired fixed-state interventions on models with up to 1.56B parameters, an intervention on individual GEMM passes, full-budget independently initialized trajectories at 124M and 355M parameters, and a matched actual-kernel FP8 accuracy baseline. Across the reported dense GPT-2 trajectories, the FP4 cores are explicit numerical simulations; the matched FP8 runs provide the actual-kernel baseline. Recent large-scale FP4 pretraining results (Chmiel et al., 2025; NVIDIA, 2026) establish the viability of low precision at scale, and selective protection provides a complementary way to allocate precision efficiently; Appendix A contrasts the temporal quantization-aware switch of Chmiel et al. (2025) with the spatially selective protection studied here.

Throughout this appendix, the normalized RMS error (NRMSE) is the local error metric, and D_{u} denotes common-state AdamW update distortion as defined in Appendix B.7.

Three findings qualify a scalar local-error account.

First, for an isolated Dgrad intervention the measured local GEMM residual is exactly the realized local activation-adjoint error $d_k(v_k^{\mathrm{ref}}; \pi_k)$, so the load–quantization–amplification (LQA) product $L_k Q_k A_k$ below is exact without requiring the quantizer to be linear. Fprop changes the forward

state, and Wgrad writes directly through the parameter-gradient readout, so the local GEMM-output residuals from Fprop and Wgrad do not furnish the same identity.

Second, within each of 16 endpoint–batch states, pass ranks by local NRMSE have mean Spearman correlation -0.11 with single-intervention propagated risk and reverse 53.7% of comparable pairs, as Figure 9 shows; resolved into the six operands of Eq. 1, similar quantization errors produce gradient-bending values spanning two to three orders of magnitude (Table 25).

Third, resolving the selector's actions to pass units improves on equal-work group protection in one-step held-out interventions and in continued pretraining (Appendices E.6 and E.8). These results support the selector as a causal diagnostic and useful intervention.

Naming. The selector used in this appendix is the bending-guided precision allocation of Sec. 5 with two changes: The action space is refined from boundary groups to pass units, each the Fprop, Dgrad, or Wgrad pass of one layer–operation group, and the objective is the update-level D_{u} of Appendix B.7, because the FP4 cores rotate the AdamW update far more than the raw gradient. Allocation to an operation group protects the group's Fprop, Dgrad, and Wgrad GEMMs together; allocation to individual GEMM passes selects among these three computations separately. Comparisons use matched calibration data and budgets. Bending-guided precision allocation re-evaluates the executed mask after each choice; each baseline follows the scoring and selection procedure specified for that comparison.

E.1 Diagnosing Formats and Identifying Well-Conditioned Precision Settings

The same diagnostics explain why one 4-bit format rotates the gradient far less than another at a similar local error and identify, in the initial calibration of the allocation, a well-conditioned FP8 path that needs no protection.

Diagnosing Formats. In the controlled GPT-2 124M GEMM cores at the pretrained checkpoint, with every Fprop, Dgrad, and Wgrad operand quantized, the NV core lowers the operand error of the MX core by only 25%, from 0.127 to 0.095, but lowers the gradient bending about tenfold, from $D_{\mathrm{g}} = 0.227$ to 0.023, or from 39° to 12° (Table 21). A block-size-by-scale-encoding factorial finds that E4M3 block-scale encoding drives most of this reduction by cutting the average scale error from about 19% to 2.3% (Appendix E.3). The NV core rotates both the gradient and the AdamW update less in every model and state up to 1.56B parameters (Appendix E.10), so the diagnostics recommend prioritizing scale encoding over block size.

Identifying Well-Conditioned Precision Settings. The diagnostics that flag damaging policies also identify well-conditioned policies. Actual-kernel E4M3 scaled GEMMs in every GPT-2 block, which quantize only the GEMM operands, start from-scratch training at D_{g} of 0.004 to 0.005, near the reference floor and several times below the simulated FP4 core with selected forward GEMMs in BF16, and all three runs at each of 124M and 355M parameters end within 1% of the paired BF16 loss (Table 31 in Appendix E.11). Step-0 geometry thus separates the well-conditioned FP8 path from the FP4 cores before training; Appendix D.14 isolates the kernel path with GEMM microbenchmarks.

E.2 Hardware Scope and Experimental Controls

The primary controlled-core experiments used eight NVIDIA GH200 nodes with 120 gigabytes of memory each. Each process used one SM90 GPU, for which PyTorch reported 97,280 MiB; independent seeds or policies were distributed one per node through the Message Passing Interface. PyTorch 2.5.1 exposed neither a float4 dtype nor a hardware FP4 tensor-core endpoint on this platform. Consequently, the FP4 accuracy and gradient-geometry results use numerical simulation: Operands are explicitly quantized and dequantized, then multiplied by PyTorch GEMMs with BF16 inputs and outputs whose internal accumulation is not instrumented. Wall time and peak memory characterize the simulator. These results quantify numerical accuracy and geometry. Separate native NVFP4 GEMM timings on an RTX 5090 Laptop GPU are reported in Appendix E.7; the native NVFP4 timings isolate GEMM kernel time. The same GH200 allocation exposes hardware E4M3 scaled GEMMs, which supply the separately labeled actual-kernel FP8 accuracy baseline. The largest fixed-state experiment uses a 1.56B-parameter model.

Table 20: Confirmatory low-precision protocol for the GPT-2 family; 124M, 355M, and 1.56B denote GPT-2, GPT-2 Medium, and GPT-2 XL. The statistical units are paired batches or seeds, and trajectory token counts are per policy and seed; these conventions apply to the FP4 tables below unless noted.

Experiment	Parameters	Sequence	Batch	Replicates and grid	Data and budget
Format 2×2 and 2^3 paths	124M	256	2	4 paired batches	pretrained WikiText-103 validation
Site propagation				8 paired batches $\times$ 2 cores	288 single pass interventions
Group and pass selection				8 calibration + 8 held-out + 8 transfer	random init.; equal pass-equivalent work
Format 2×2 and 2^3 paths	355M	128	1	4 paired batches	pretrained WikiText-103 validation
Endpoint confirmation	1.56B	64	1	16 pretrained + 8 random batches	paired MX and NV endpoints
Full trajectory	124M	256	64	3 policies $\times$ 3 seeds	WikiText-103; 134.22M tokens
Full trajectory	355M			3 policies $\times$ 3 seeds	WikiText-103; 134.22M tokens
Actual-kernel FP8 baseline	124M and 355M			2 models $\times$ 3 seeds	WikiText-103; 134.22M tokens

Table 20 summarizes the confirmatory protocols. WikiText-103 provides the natural-language corpus for the matched-budget studies. WikiText hyperparameter and selector calibration use validation blocks; final trajectory outcomes use the disjoint test split under the three-seed paired protocol of Appendix C.4. The matched-budget runs perform 8,192 optimizer steps with batch 64 and sequence length 256, or 134,217,728 input tokens. The WikiText packed training pool has 393,216 blocks totaling 100,663,296 tokens, so the deterministic shuffled loader is traversed approximately 1.33 times in the primary grid. WikiText final loss uses a fixed 32,768-token test slice.

The BF16 baseline uses fused AdamW, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1, cosine decay, 5% warmup, and global clipping at 1.0. Validation tuning selects learning rates of 1.1×10^{-3} at 124M and 8.0×10^{-4} at 355M. The corresponding three-seed test losses are 3.2313 ± 0.0086 and 3.1313 ± 0.0011, respectively. Thus FP4 is compared with a scale- and budget-tuned BF16 training recipe. The optimizer and FP4 core study uses a fixed implementation snapshot, and the audit independently verifies the core implementation together with the data-processing and orchestration components. These checks establish the source snapshot used for the reported results.

MX Format Implementation. Every core uses the symmetric signed E2M1 codebook whose nonnegative magnitudes are $\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$. Rounding uses round-to-nearest with ties to even (RNE). The Open Compute Project (OCP) MX format fixes E2M1 elements, an 8-bit exponent-only scale with 8 exponent bits and 0 mantissa bits (E8M0), and groups of 32. The specification permits implementation-defined conversion algorithms; we use the exponent-alignment rule in the reference conversion algorithm of Rouhani et al. (2023), which follows the specification's conversion semantics (Open Compute Project, 2023):

$$s_{\mathrm{MX},b} \coloneqq 2^{\lfloor \log_2 a_{\max,b} \rfloor}/4. \tag{14}$$

NV Hierarchical Scaling. For the NV core, groups of 16 use the two-level construction described for hierarchical NVFP4 scaling (Chmiel et al., 2025):

$$\begin{aligned} s_{\mathrm{global}} &\coloneqq a_{\max}^{\mathrm{global}}/(448 \cdot 6), \qquad \widehat{s}_b \coloneqq Q_{\mathrm{E4M3}}\left(\frac{a_{\max,b}/6}{s_{\mathrm{global}}}\right), \\ s^{\mathrm{eff}}_{\mathrm{NV},b} &\coloneqq s_{\mathrm{global}} \widehat{s}_b. \end{aligned} \tag{15}$$

Here $\widehat{s}_b$ is the represented E4M3 block-scale factor and $s^{\mathrm{eff}}_{\mathrm{NV},b}$ is the effective dequantization scale, used in the reconstruction $s^{\mathrm{eff}}_{\mathrm{NV},b} Q_{\mathrm{E2M1}}(x_b / s^{\mathrm{eff}}_{\mathrm{NV},b})$.

Edge Cases and Shared Controls. Eqs. 14 and 15 describe the case of a positive absolute maximum (amax). In the implementation, the MX exponent is clipped to the represented E8M0 range; all-zero

Table 21: GPT-2 124M block-size × scale-encoding factorial under the checkpoint-0 protocol in Table 20. All rows use E2M1 values, RNE, one-dimensional scaling, and all three low-precision GEMM paths. Lower is better for NRMSE, D_{g}, angle, D_{u}, and policy loss.

Core	Block	Scale	Operand NRMSE	Raw D_{g}	Angle (°)	R_{norm}	D_{u}	Loss
MX endpoint	32	E8M0	0.1271	0.2271	39.19	0.9779	0.6785	4.5455
Hybrid block-16	16		0.1246	0.2159	38.30	1.0384	0.6718	4.5140
Hybrid-E4M3	32	E4M3+FP32	0.1059	0.0505	18.22	1.1041	0.4870	4.1573
NV endpoint	16		**0.0952**	**0.0234**	**12.38**	1.0095	**0.3841**	**3.9372**

MX blocks use a unit computational scale, an all-zero NV tensor uses a positive safe global scale, and any represented zero NV factor $\widehat{s}_b$ is replaced by one before the effective scale is formed and used for division. Exact zero blocks therefore remain exactly zero without evaluating $\log_2 0$ or dividing by zero.

The MX exponent-alignment rule in Eq. 14 can saturate a block maximum whose normalized magnitude lies between 6 and 8; alternative OCP-compliant MX scale-selection rules need not do so. Accordingly, the MX core specified here is a fully specified numerical simulation, while bitwise hardware-kernel behavior remains implementation-dependent. The two hybrid cells independently exchange block size and scale encoding. Every core uses deterministic rounding and row-and-column weight scaling, with randomized Hadamard transforms disabled.

The custom autograd path replaces the Conv1D GEMMs inside GPT-2 blocks and separately quantizes Fprop, Dgrad, and Wgrad operands. Embeddings, normalization, softmax, loss, the LM head, optimizer state, and non-GEMM operations remain at the reference precision. Reference GEMMs use BF16 operands and BF16 outputs under PyTorch autocast. Unit and integration tests cover the E2M1 codebook and ties, block scales, pass isolation, BF16 custom autograd, AdamW geometry against an applied optimizer step, exact-budget selectors, dropout-free finite-difference scoring, frozen masks, reference linkage, trajectory grids, and locked, atomic result writes.

The actual-kernel FP8 control uses one current-amax scale per operand tensor, casts both operands to E4M3, and invokes `torch._scaled_mm` with BF16 output and the fast-accumulation option for every Fprop, Dgrad, and Wgrad GEMM in the same GPT-2 block Conv1D scope (Micikevicius et al., 2022). Master parameters and AdamW moments remain FP32; the LM head and non-GEMM operations follow the BF16 reference path. The coverage of the actual-kernel FP8 control matches the FP4 core and provides a per-tensor FP8 baseline distinct from microscaling FP8 and Transformer Engine precision configurations. Operand scaling, padding, and dispatch remain in an unfused Python wrapper, so the reported wall time and memory characterize this implementation; optimized systems performance requires a fused path.

E.3 Controlled Block–Scale Factorial

Table 21 reports the four format cores on the same four checkpoint–batch states. The policy covers Fprop, Dgrad, and Wgrad in every GPT-2 block. Operand NRMSE is accumulated from operand residual sums; the remaining columns compare the concatenated parameter gradient or common-state AdamW update with BF16. The MX endpoint reduces the gradient norm and rotates the gradient: $R_{\mathrm{norm}} = 0.978 < 1$ while the raw gradient rotates by $39.2°$. The NV endpoint reduces the angle to $12.4°$ and D_{u} from 0.679 to 0.384.

The operand diagnostics in Table 22 further distinguish the effects of block size and scale encoding. The hierarchical E4M3 scaling in Eq. 15 reduces average scale error from approximately 19% to 2.3%. Block 16 lowers underflow but raises saturation in these cells. Thus, block size trades one local failure mode against another, while scale encoding has the clearer effect on scale error; neither cell label alone captures overall accuracy.

Table 23 gives paired factorial contrasts. Effects are coded as level 2 minus level 1, so a negative value is an improvement. The interaction is the block-16-minus-block-32 contrast under E4M3 minus the corresponding contrast under E8M0. E4M3+FP32 scaling is the dominant main effect at this

Table 22: Aggregated operand diagnostics for the four factorial cells. Fractions and scale statistics are element-weighted over all instrumented operands.

Core	NRMSE	Underflow (%)	Saturation (%)	Scale error (%)	Utilization (%)
MX endpoint	0.1271	14.07	2.05	18.91	29.43
Hybrid block-16	0.1246	11.38	3.66	18.80	34.90
Hybrid-E4M3	0.1059	13.64	1.80	2.32	30.67
NV endpoint	0.0952	11.13	3.52	2.34	36.35

Table 23: Paired 2×2 factorial effects with 10,000 paired-bootstrap resamples over four paired tuples. Brackets give 95% intervals. The raw D_{g}, D_{u}, and loss rows are contrasts of the corresponding columns of Table 21; the NRMSE row reports contrasts of local GEMM-output NRMSE.

Outcome	Block 16 minus 32	E4M3 minus E8M0	Interaction
Local GEMM NRMSE	-0.00483 $[-0.00522, -0.00444]$	-0.04845 $[-0.04879, -0.04811]$	-0.01169 $[-0.01230, -0.01083]$
Raw D_{g}	-0.01915 $[-0.03219, -0.00610]$	-0.18452 $[-0.22385, -0.15119]$	-0.01584 $[-0.04282, 0.01114]$
AdamW D_{u}	-0.05482 $[-0.05877, -0.05104]$	-0.23965 $[-0.25584, -0.22346]$	-0.09623 $[-0.10773, -0.08570]$
Policy loss	-0.12578 $[-0.17005, -0.08646]$	-0.48253 $[-0.51243, -0.44489]$	-0.18863 $[-0.24019, -0.12538]$

checkpoint. A block size of 16 also helps on average, but the block–scale interaction is material, particularly after AdamW. The bootstrap resamples the four complete block–scale tuples; with four tuples, these intervals describe variation across the calibration batches.

Summary. Relative to the MX endpoint, the NV endpoint lowers operand NRMSE by 25% but raw gradient bending about tenfold. In the 124M block–scale factorial, E4M3 scale encoding is the dominant effect: E4M3 scaling cuts average scale error from about 19% to 2.3% and reduces raw D_{g}, D_{u}, and policy loss far more than the block-size change. Block 16 trades underflow for saturation, and the block–scale interaction is material after AdamW, so neither cell label alone predicts the resulting gradient and update distortion.

E.4 Fprop, Dgrad, Wgrad, and Direct Readout Error

Where the Error Enters. The forward route of Eq. 1 carries most of the measured distortion. The gradient bending D_{g}, defined in Sec. 4, is one minus the cosine between the reference and policy gradients. Under all-low FP8 boundary quantization, quantizing only the forward activations produces $D_{\mathrm{g}} = 0.094$ in ResNet-18, against 0.0016 for quantizing only the arriving gradients and 0.095 for both (Table 3 in Appendix D.4); the GPT-2 values are 0.069, 0.013, and 0.074 (Table 13).

Resolving the NV GEMM core of GPT-2 into the six operands of Eq. 1 gives the same ordering at the smallest unit. The operands carry similar relative errors, 0.077 to 0.099, yet quantizing one Fprop operand in every block GEMM produces parameter-gradient bending of $D_{\mathrm{g}} = 0.009$ to 0.011, one Dgrad operand by 0.0006 to 0.0011, and one Wgrad operand by 0.0001 (Table 25 in Appendix E.4). Quantizing the gradient operand G_Y in both backward passes produces $D_{\mathrm{g}} = 0.0007$, about 16 times less than quantizing the activation X in both Fprop and Wgrad, so protecting the backward path, which removes the gradient-operand errors but leaves the forward route in low precision, repairs little (Appendices D.7.1 and E.8.1).

In every single-operand cell and batch of the NV and MX cores, the orthogonal component of the shift exceeds the parallel component, by a factor of 3 to 60 in the batch means, so each operand error mostly rotates the gradient.

Table 24 holds layer coverage fixed and executes the complete 2^3 path factorial. Fprop is the dominant source at this pretrained state. Wgrad-only is nevertheless nonnegligible after AdamW: The Wgrad-

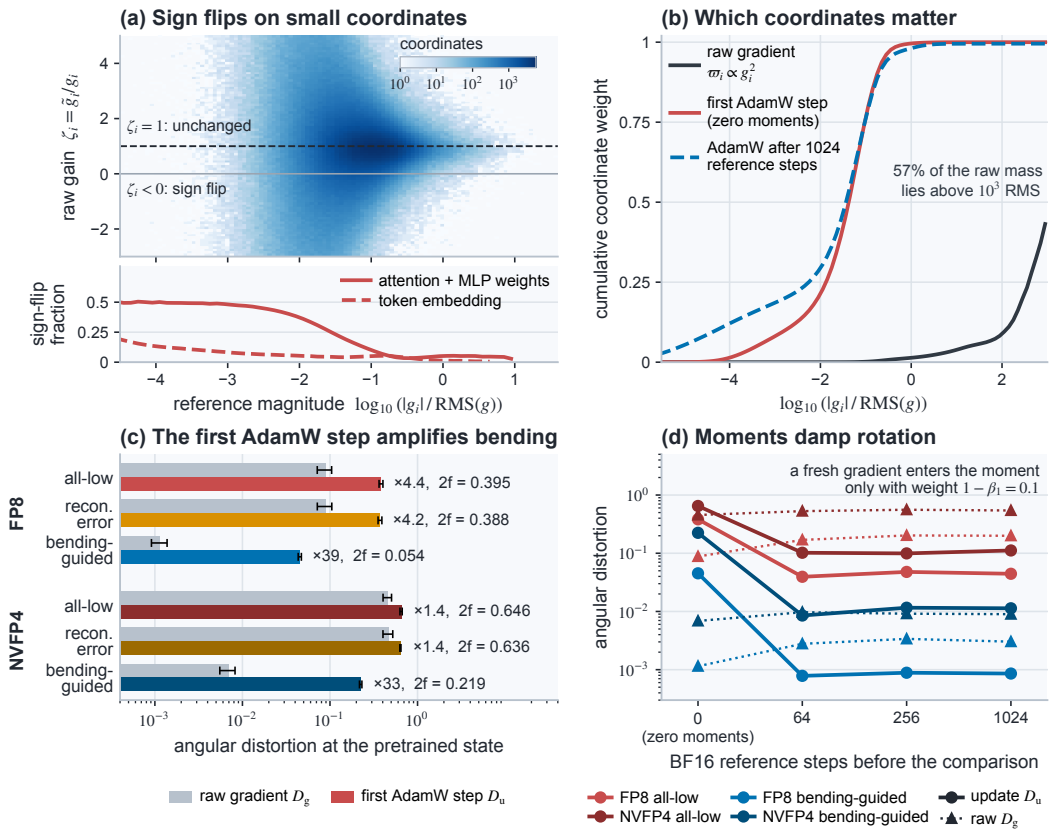


Figure 8: Coordinate-wise update gains on GPT-2 124M. AdamW updates exhibit more bending than raw gradients because low precision flips the signs of small coordinates. Reference and precision-policy gradients pass through the same common-state AdamW map under the stated protocol. (a) Raw gain $\zeta_i = \tilde{g}_i/g_i$ and sign-flip fraction under the all-low FP8 policy. (b) Cumulative reference weight ϖ_i under three update maps. (c) D_g and D_u at the pretrained state; $2f$ is twice the sign-flip fraction. (d) The same diagnostics after 64 to 1024 BF16 reference steps.

only raw D_g is only 0.00049 for MX and 0.00022 for NV, but the first common-state adaptive update has $D_\mathrm{u} = 0.077$ and 0.054, respectively. Thus, a small direct parameter-gradient-readout error can become much more prominent after the adaptive coordinate-wise update map. Theorem 1 gives the exact decomposition in optimizer-induced update gains and reference-update weights, while Corollary 3 isolates the fixed-divisor case in which raw gains are reweighted.

Figure 8 measures the gains and weights of Theorem 1 on GPT-2 AdamW states and gives this amplification a coordinate-level origin. The measurement uses GPT-2 124M on four 2×256-token WikiText-103 calibration batches. Reference and precision-policy gradients are clipped to norm 1 and passed through the same common-state AdamW map with the decoder continued-pretraining recipe: learning rate 5×10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-2}. The initial checkpoint uses the paper's D_u protocol at pretrained weights with zero moments; later states follow 64 to 1024 BF16 reference steps. The protected sets are the three-group bending-guided sets, the early and middle normalization groups with the LM head under FP8 and the early attention and final normalization groups with the LM head under NVFP4, and the three-group reconstruction-error sets, the embedding with the early and final normalization groups under FP8 and the embedding with the middle attention and late normalization groups under NVFP4.

At the pretrained state with zero moments, Figure 8(a) plots the raw gain $\zeta_i = \tilde{g}_i/g_i$ of attention and MLP weights under the all-low FP8 policy over a random sample of two million coordinates. The gain remains close to 1 above the RMS gradient magnitude, but the gain dispersion grows as $|g_i|$ shrinks, and the sign flips for about half of the coordinates below 10^{-3} RMS. The token embedding

Table 24: Complete Fprop, Dgrad, and Wgrad factorial for the 124M setting of Table 21.

	MX endpoint			NV endpoint		
Low passes	Raw D_{g}	Angle (°)	D_{u}	Raw D_{g}	Angle (°)	D_{u}
None (BF16)	0	0	0	0	0	0
Wgrad	0.00049	1.79	0.0771	0.00022	1.18	0.0541
Dgrad	0.00409	5.16	0.1477	0.00173	3.35	0.1059
Dgrad, Wgrad	0.00469	5.52	0.1724	0.00194	3.56	0.1237
Fprop	0.29120	44.70	0.6754	0.02266	12.16	0.3657
Fprop, Wgrad	0.28329	44.06	0.6781	0.02295	12.24	0.3713
Fprop, Dgrad	0.23287	39.71	0.6758	0.02314	12.31	0.3789
Fprop, Dgrad, Wgrad	0.22711	39.19	0.6785	0.02344	12.38	0.3841

flips less often because 99.5% of the token-embedding rows correspond to tokens absent from the batch.

The raw diagnostic hides these flips because 93% of the raw gradient energy lies in the token embedding. The raw diagnostic's weights $\varpi_i \propto g_i^2$ concentrate on the largest coordinates, as Figure 8(b) shows. By contrast, the bias-corrected first AdamW step is the sign-like map $g_i/(|g_i| + \epsilon)$ for any (β_1, β_2) and weights all 124 million coordinates nearly equally. Consequently, $\zeta_i \approx \pm 1$ and $D_{\mathrm{u}} \approx 2f$ for sign-flip fraction f.

For example, if quantization changes the parameter gradient from $g = (1, 0.01)$ to $(1, -0.01)$, the resulting bending is only $D_{\mathrm{g}} \approx 2 \times 10^{-4}$, whereas the sign-like updates $(1, 1)$ and $(1, -1)$ are orthogonal. Under all-low FP8, the first GPT-2 update has a norm ratio of 1.01 and bending of 0.38, close to twice the 20% gradient-coordinate sign-flip fraction (Figure 8).

Figure 8(c) reports $D_{\mathrm{u}} = 0.383$ and $D_{\mathrm{g}} = 0.088$ for all-low FP8 at $f = 0.198$, and $D_{\mathrm{u}} = 0.649$ and $D_{\mathrm{g}} = 0.454$ for all-low NVFP4 at $f = 0.323$. Even a bending-guided precision allocation with negligible raw-gradient bending flips 2.7% of signs under FP8 and 11% under NVFP4 at the first step. Evaluating Theorem 1 on the measured gains and weights reproduces every measured D_{u} with $\lambda_0 = 0$, while $2f$ agrees up to the ϵ smoothing. Because both gradients are clipped to unit norm and the first AdamW step is sign-like, the update norm ratio stays between 0.98 and 1.06 for all six policies at all four states of the figure, whereas the update bending ranges from 0.0008 to 0.65, so update direction accounts for most of the difference between policies.

These results motivate scoring individual GEMM passes with D_{u} (Appendix E.6) and using the zero-moment D_{u} as a conservative selection statistic. Once moments accumulate, the common-state comparison sees only the $(1 - \beta_1)$ increment of the fresh gradient. Figure 8(d) then shows update bending below raw-gradient bending: After 64 to 1024 BF16 reference steps, shared-state D_{u} lies between 0.04 and 0.05 for all-low FP8 and between 0.10 and 0.11 for all-low NVFP4, while raw D_{g} grows to 0.20 and 0.54. Along a policy's own trajectory, however, the optimizer moments average rotated gradients, as in Table 14. This damping characterizes the common-state comparison; a policy's own optimizer history can accumulate directional error.

The Wgrad-only rows use reference activations and reference output gradients and therefore implement the fixed-input kernel error. The rows that quantize Fprop and Dgrad let low-precision operand states change while Wgrad remains BF16, and the all-path row additionally turns on the low-precision Wgrad GEMM. Comparing these controlled rows exposes input-state and interaction effects without mislabeling Wgrad as an activation-side upstream error. Because both angular-distortion metrics are nonlinear, differences between the scalar metric values are descriptive; the vector-valued single-intervention analysis below is used for attribution. The observed Fprop dominance characterizes this pretrained state, tensor coverage, and explicit-dequantization wrapper. Actual-kernel full-pipeline FP4 training may shift the bottleneck because kernel layouts and Wgrad-specific scaling and rotation differ (Cim et al., 2026).

Operand-Level Resolution. Table 25 resolves each pass into the two operands of Eq. 1 by quantizing one operand at a time in every block GEMM, on the same four batches. The rows of Table 25 that quantize both operands of a pass or every operand repeat rows of Table 24 in an independent rerun on

Table 25: Operand-level factorial for the GPT-2 124M block GEMMs at the checkpoint-0 state of Table 21, as means over four paired batches. Each row quantizes the listed operands of Eq. 1 in every block GEMM and uses BF16 for the other operands; NRMSE is the pooled relative quantization error $(\sum \|Q(Z) - Z\|^2 / \sum \|Z\|^2)^{1/2}$ of the quantized operands.

		MX endpoint			NV endpoint		
Pass	Quantized operands	NRMSE	Raw D_{g}	D_{u}	NRMSE	Raw D_{g}	D_{u}
Fprop	X	0.146	0.16864	0.5821	0.092	0.01123	0.2646
	W	0.117	0.03420	0.3627	0.095	0.00931	0.2380
	X and W	0.134	0.29105	0.6753	0.093	0.02268	0.3658
Dgrad	G_Y	0.128	0.00197	0.1018	0.093	0.00062	0.0659
	W	0.118	0.00202	0.1006	0.094	0.00114	0.0797
	G_Y and W	0.118	0.00423	0.1474	0.094	0.00171	0.1055
Wgrad	X	0.125	0.00018	0.0494	0.099	0.00011	0.0377
	G_Y	0.141	0.00030	0.0564	0.077	0.00012	0.0362
	X and G_Y	0.125	0.00049	0.0771	0.099	0.00022	0.0542
Every pass that reads the operand	X	0.135	0.16735	0.5843	0.096	0.01140	0.2688
	W	0.118	0.03195	0.3720	0.095	0.01006	0.2523
	G_Y	0.134	0.00234	0.1212	0.086	0.00073	0.0787
	all operands	0.127	0.22648	0.6785	0.095	0.02368	0.3842

one NVIDIA GeForce RTX 5090 Laptop GPU, which agrees with Table 24 to within 0.0007 in raw D_{g} and 0.0005 in D_{u}. The six operands carry relative quantization errors between 0.117 and 0.146 under the MX core and between 0.077 and 0.099 under the NV core, yet the resulting raw D_{g} spans nearly three orders of magnitude under MX and two under NV, ordered by the route of the error in Sec. 2: Fprop operands rotate the gradient most, Dgrad operands less, and Wgrad operands least.

Within Fprop, the NV core produces similar gradient bending from quantizing the activation and the weight, $D_{\mathrm{g}} = 0.011$ and 0.009, whereas under the MX core, quantizing the activation produces five times as much bending as quantizing the weight, 0.169 against 0.034. Quantizing the gradient operand G_Y in both backward passes gives raw D_{g} of 0.0023 under MX and 0.0007 under NV, below every row that quantizes X or W in Fprop.

The AdamW update preserves the pass ordering but compresses the range of bending values, from two orders of magnitude to a factor of about seven under NV, because the sign-like first step counts every small coordinate that quantization flips (Figure 8). In every single-operand cell and batch, $\kappa^{\perp}$ exceeds $|\kappa^{\parallel}|$, by a factor of 3 to 60 in the batch means, so each operand error mostly rotates the gradient.

Summary. Fprop dominates the FP4 gradient error at this pretrained state, and resolving each pass into the two constituent operands preserves that order: The six operands carry similar relative errors, yet the resulting raw D_{g} spans two to three orders of magnitude, from the Fprop operands down to the Wgrad operands. The AdamW map nevertheless amplifies small direct readout errors: Wgrad-only rotates the raw gradient by under 2° yet gives D_{u} of 0.077 for MX and 0.054 for NV, because the first bias-corrected step weights all coordinates nearly equally and $D_{\mathrm{u}} \approx 2f$ for sign-flip fraction f. The selector over individual GEMM passes therefore scores the zero-moment D_{u}.

E.5 Incoming Load, Local Directional Error, and Propagation

For each of six depth–operation groups and three passes, we replay a single low-precision intervention on the BF16 graph. For an isolated Dgrad intervention, define the realized local residual

$$e_k^{\mathrm{D}} := d_k(v_k^{\mathrm{ref}}; \pi_k).$$

In the frozen or linearized specialization, $e_k^{\mathrm{D}} = E_k v_k^{\mathrm{ref}}$. Using e_k^{D} directly makes the LQA measurement for each pass exact for the realized, possibly nonlinear quantizer. Whenever all three

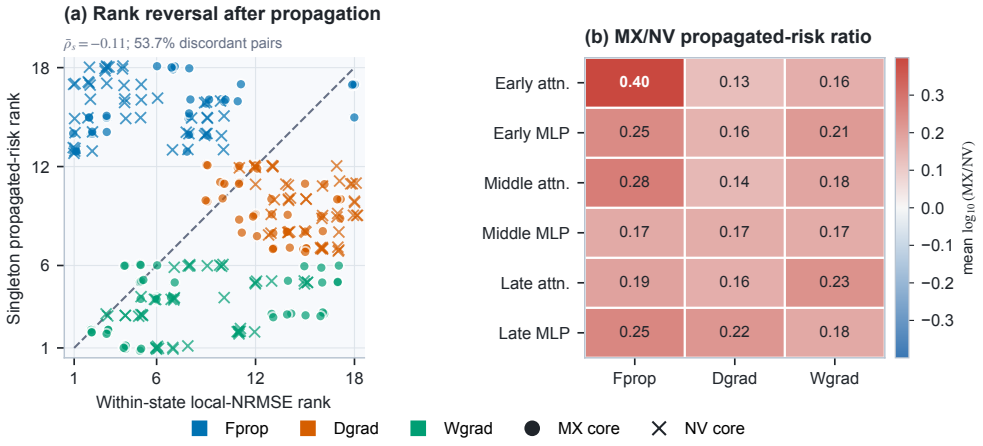


Figure 9: Local error does not determine propagated risk. (a) Within-state local-NRMSE rank against single-intervention propagated-risk rank, computed separately in each of 16 endpoint–batch states over 18 pass units. (b) Mean paired $\log_{10}$ ratio of MX to NV propagated risk over eight batches; positive cells favor NV.

denominators below are nonzero, we measure

$$L_k := \frac{\|v_k^{\mathrm{ref}}\|_2}{\|g_\theta\|_2}, \qquad Q_k := \frac{\|e_k^{\mathrm{D}}\|_2}{\|v_k^{\mathrm{ref}}\|_2}, \qquad A_k := \frac{\|\mathcal{T}_k e_k^{\mathrm{D}}\|_2}{\|e_k^{\mathrm{D}}\|_2}. \tag{16}$$

The common-state measurement makes L_k identical across MX and NV at a matched Dgrad site. Algebraically,

$$L_k Q_k A_k = \frac{\|\mathcal{T}_k e_k^{\mathrm{D}}\|_2}{\|g_\theta\|_2};$$

numerically, the measured factors reconstruct this normalized propagated contribution to maximum relative error 1.8×10^{-14}.

We reserve the exact LQA factorization for Dgrad: Fprop changes the forward state before the loss and backward graph are evaluated, while Wgrad is a direct parameter-gradient readout. For all three passes, the observed local GEMM-output NRMSE can still be compared with the exact single-intervention parameter-gradient shift as a ranking diagnostic.

Figure 9 shows that local reconstruction error is not a reliable all-pass ranking. Within each of the 16 endpoint–batch states, the mean Spearman correlation between local GEMM-output NRMSE rank and single-intervention propagated-risk rank is -0.11, and 53.7% of comparable pass pairs reverse order. The paired MX and NV propagated-risk ratios in Figure 9(b) separate most strongly at early-attention Fprop, and every pass cell shown favors NV in this state. The result supports evaluating the joint propagated effect: No single scalar factor consistently orders risk across these pass units.

Summary. For isolated Dgrad interventions, the LQA product reconstructs the propagated contribution exactly, to relative error 1.8×10^{-14}. Across pass units, however, local GEMM-output NRMSE ranks propagated risk poorly: The within-state Spearman correlation averages -0.11, 53.7% of comparable pairs reverse order, and every pass cell shown favors the NV core.

E.6 Allocating Precision to Individual GEMM Passes at Equal Work

We compare two action granularities under the deterministic NV core with one-dimensional scaling. The group-level action space contains 24 layer–operation groups, formed by crossing 12 layers with attention and MLP operations. Each group protects Fprop, Dgrad, and Wgrad together. The

Table 26: Equal-work intervention on individual GEMM passes over eight batches per state. Entries are means ± standard deviations. The upper block reports the selection state, and the lower block evaluates the same frozen masks at a new initialization and on previously unused validation blocks.

Selection state			
Selector	Protected actions	Raw D_{g}	D_{u}
All-low	none	0.04939 ± 0.00262	0.20633 ± 0.00492
Operation groups and Fisher	M0: F, D, W	0.04146 ± 0.00266	0.18634 ± 0.00463
Group fixed	A0, A11: F, D, W	0.04418 ± 0.00240	0.19504 ± 0.00404
Group Hessian	M3: F, D, W	0.04645 ± 0.00222	0.20002 ± 0.00553
Group random	M9: F, D, W	0.04775 ± 0.00268	0.20298 ± 0.00474
Individual GEMM passes, reconstruction error	A5-F, A6-F, A6-D, A7-F, A9-F, A11-F	0.04741 ± 0.00286	0.19950 ± 0.00516
Random GEMM passes	A0-W, M3-W, M10-F, A11-F	0.04870 ± 0.00270	0.20395 ± 0.00599
Allocation to individual GEMM passes	A0-F, M0-F, A1-F, M1-F	$\mathbf{0.03237 \pm 0.00197}$	$\mathbf{0.15953 \pm 0.00288}$
Frozen transfer state			
Selector	Protected actions	Raw D_{g}	D_{u}
All-low	none	0.04685 ± 0.00140	0.20956 ± 0.00346
Operation groups and Fisher	M0: F, D, W	0.04149 ± 0.00141	0.19350 ± 0.00381
Group fixed	A0, A11: F, D, W	0.04169 ± 0.00128	0.19663 ± 0.00421
Group Hessian	M3: F, D, W	0.04459 ± 0.00155	0.20373 ± 0.00216
Group random	M9: F, D, W	0.04552 ± 0.00152	0.20628 ± 0.00307
Individual GEMM passes, reconstruction error	A5-F, A6-F, A6-D, A7-F, A9-F, A11-F	0.04532 ± 0.00155	0.20399 ± 0.00300
Random GEMM passes	A0-W, M3-W, M10-F, A11-F	0.04616 ± 0.00168	0.20674 ± 0.00352
Allocation to individual GEMM passes	A0-F, M0-F, A1-F, M1-F	$\mathbf{0.03210 \pm 0.00095}$	$\mathbf{0.15943 \pm 0.00327}$

action space over individual GEMM passes refines these groups into 72 layer–operation–pass units. For a fixed token batch, the three linear passes have the same leading GEMM work, so a group action costs three times the protected weight count of the group. Every nonzero policy below spends exactly 14,155,776 pass-equivalent elements, or 5.56% of the 254,803,968 eligible group–pass total. The budget matches leading GEMM work; actual-kernel latency and memory remain hardware measurements.

Bending-guided precision allocation greedily minimizes the exact joint-mask D_{u} per protected element on eight calibration batches and recomputes the executed mask after every action. Reconstruction-error selection over individual GEMM passes and a frozen mask that selects individual GEMM passes at random share the 72-action set. Allocation to operation groups, activation-Fisher, finite-difference Hessian, fixed first-layer and last-layer attention, and random selection use the 24-group action space, the same calibration batches, and the same pass-equivalent budget; the independently selected allocation to operation groups and Fisher policies coincide on the layer-0 MLP group and are therefore reported as one executed mask. Allocation to individual GEMM passes selects Fprop in the attention and MLP operations of layers 0 and 1. In the action labels below, Ai and Mi denote the attention and MLP operations in layer i, respectively; F, D, and W denote Fprop, Dgrad, and Wgrad.

Table 26 first evaluates every frozen policy on eight disjoint held-out batches at one fixed initialization. The experiment then transfers every mask without retuning to a second initialization and to validation blocks 32 through 47, which are disjoint from both the original calibration and held-out blocks.

Table 26 shows that, against allocation to operation groups and Fisher, allocation to individual GEMM passes lowers held-out D_{u} by 0.02681, with a 95% paired-bootstrap interval of $[0.02420, 0.02953]$; against all-low, the reduction is 0.04680 $[0.04404, 0.04955]$. At the transferred state, the reductions are 0.03408 $[0.03324, 0.03498]$ and 0.05013 $[0.04907, 0.05106]$, respectively. The selector over

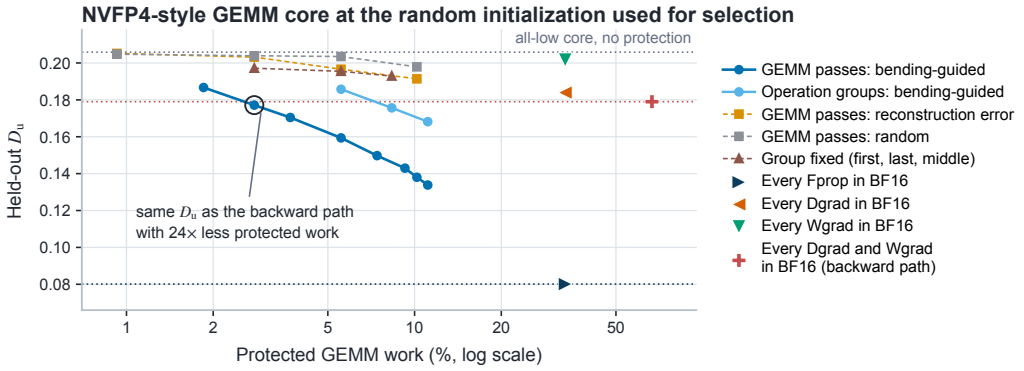


Figure 10: Held-out AdamW-update bending against protected GEMM work in the NV GEMM core of GPT-2 124M at the random initialization used for selection in the from-scratch study (Table 26). Solid lines trace the greedy paths of bending-guided precision allocation to individual GEMM passes and to operation groups, dashed lines connect the other selectors at budgets of 1.4%, 2.8%, 5.6%, and 11.1%, markers denote policies that execute entire GEMM pass types in BF16, and dotted lines mark the all-low, Fprop, and backward levels.

individual GEMM passes has lower D_u than every competitor on all eight batches in both states. Selecting individual GEMM passes allocates the protected work more selectively for this immediate objective and breaks the tie between allocation to operation groups and Fisher. The continued-pretraining study of Appendix E.8 tests whether the advantage of selecting individual GEMM passes survives optimization.

Summary. At equal leading-GEMM work, resolving the selector's actions to pass units helps: Allocation to individual GEMM passes lowers held-out D_u by 0.027 relative to allocation to operation groups and Fisher and by 0.047 relative to all-low, with every paired difference positive on all eight batches in both the selection and frozen-transfer states. The protected actions are the Fprop passes of the attention and MLP operations in layers 0 and 1.

E.7 Protected GEMM Work, Modeled Time, and Update Bending

The equal-work comparison of Appendix E.6 fixes one budget. Figure 10 traces the whole greedy path of bending-guided precision allocation to individual GEMM passes and to operation groups at the selection state of Table 26, the random initialization of the from-scratch study, up to 11.1% of the pass-equivalent work; Figure 3(d) shows the corresponding paths, in raw-gradient bending, at the pretrained weights of the continued-pretraining study of Appendix E.8.

At budgets of 1.4%, 2.8%, 5.6%, and 11.1%, the figure adds the reconstruction-error and random selectors over individual GEMM passes and the fixed group rule that protects the first-, last-, and middle-layer attention operations in that order; at 1.4%, no group action fits the budget. Uniform policies execute every Fprop, Dgrad, or Wgrad GEMM, or both backward GEMMs, in BF16. Every policy is scored on the eight held-out batches. The rerun on one RTX 5090 Laptop GPU reproduces the bending-guided selections of Table 26: At the 5.6% budget, allocation to individual GEMM passes returns the same four Fprop passes with held-out $D_u = 0.1594$, against 0.1858 for allocation to operation groups and 0.2059 for the all-low core.

Allocation to individual GEMM passes has the lowest held-out D_u at every budget, on all eight held-out batches against every other selector. The first two selections of individual GEMM passes, the Fprop passes of the layer-0 attention and MLP operations at 2.8% of the work, already match the BF16 backward path, which spends two thirds of the work, with held-out D_u of 0.1771 against 0.1790 and a lower value on six of eight batches; every later action falls below the backward path on all eight batches. Per pp of protected work, the allocation path over individual GEMM passes removes 3.1% to 5.0% of the all-low update bending, whereas the group-level path removes 1.6% to 1.8%, the BF16 Fprop path 1.8%, and the BF16 Dgrad, Wgrad, and backward paths 0.3%, 0.06%, and

0.2%. The BF16 Fprop path lowers D_{u} furthest, to 0.080, at one third of the work, and the allocation path over individual GEMM passes descends toward that level by protecting the Fprop passes of successive early layers. The full allocation path over individual GEMM passes takes 517 evaluations over eight calibration batches, about 2.1M tokens, or 1.6% of the 134.22M-token trajectory budget.

Measured and Modeled Time. Separate native-kernel timings on one RTX 5090 Laptop GPU show that the block GEMMs of a continued-pretraining step run 4.7 times faster in NVFP4 than in BF16, and the protected work sets how much of that speedup a mask achieves: The first selection of an individual GEMM pass achieves 97%, the six-action mask 82%, and the 16.7% masks 60% to 62%, whereas the BF16 backward path achieves 30% (Table 27); the target-level mask still runs the block GEMMs 2.2 times faster than BF16 while the weights learned under the target-level mask end within 0.004 of BF16 training. The modeled time $p + (1 - p)/r$ for a protected fraction p of the work reproduces the measured NVFP4 times to within 6% with the measured ratio $r = 4.75$, and measured FP8 kernels, as well as the FP8 ratios measured on an NVIDIA GH200 GPU (Table 19), give the same ordering. These timings exclude operand quantization and non-GEMM work; the training outcomes in Appendix E.8.1 use simulated FP4 arithmetic. Appendix E reports the format factorials, pass and operand decompositions, and from-scratch trajectories.

Table 27 converts protected GEMM work into GEMM time, measured and modeled. The measured columns time the 144 block GEMMs, three passes of 48 linear layers, of one continued-pretraining step of GPT-2 124M, 8 sequences of 256 tokens, on one NVIDIA GeForce RTX 5090 Laptop GPU with native kernels: BF16 GEMMs for the protected passes and NVFP4 scaled GEMMs, with E2M1 operands, 16-value blocks, and E4M3 block scales, or per-tensor FP8-E4M3 scaled GEMMs with fast accumulation for the other passes, all with BF16 output.

As in the model, only the GEMM kernels are timed; operand quantization and non-GEMM work are excluded. The native NVFP4 kernels run the Fprop, Dgrad, and Wgrad GEMMs of a step 5.3, 4.7, and 4.2 times faster than BF16 and the FP8 kernels 2.4, 2.2, and 2.0 times faster, and in a wider GPT-style block of width 4096 with 8,192 tokens, NVFP4 is 4.6 to 4.8 times and FP8 1.9 to 2.0 times faster than BF16.

The modeled columns use the time $p + (1 - p)/r$ relative to BF16 for a protected fraction p of the pass-equivalent work (Appendix C.1), where r is the throughput ratio of the low-precision kernel to the BF16 kernel, instantiated with the measured FP8-E4M3 ratios of Table 19 on the GH200, $r = 2.85$ at 4096^3 and $r = 2.54$ at $8192 \times 4096 \times 4096$, and with $r = 4$; the table lists the $r = 2.85$ and $r = 4$ values. With the measured all-low NVFP4 ratio of 4.75, the model reproduces every measured NVFP4 time of the table to within 6%: The protected share of the work largely sets the time, so a mask that reaches a given bending with less protected work achieves more of the speedup.

On the measured NVFP4 kernels, the first selection of an individual GEMM pass achieves 97% of the all-low speedup and the six-action mask 82%, whereas the BF16 backward path achieves 30%, and the target-level mask runs the block GEMMs 2.2 times faster than BF16.

At the random initialization and the 5.56% budget of Table 26, allocation to individual GEMM passes retains 91% to 92% of the modeled speedup at the GH200 FP8 ratios, whereas executing every Fprop GEMM in BF16 retains 62% to 66%, and executing every Dgrad and Wgrad GEMM in BF16, the path that a late switch of the backward pass to BF16 restores, retains 45% to 49%; the first two selections of individual GEMM passes match the D_{u} of the BF16 backward path there and retain 95% to 96% of the modeled speedup.

Summary. Allocation to individual GEMM passes achieves the lowest held-out update bending at every tested budget from 1.4% to 11.1% of GEMM work, and the first two selected Fprop passes, 2.8% of the work, match the bending of the BF16 backward path, which protects two thirds of the work. Native NVFP4 kernels run the block GEMMs 4.7 times faster than BF16, and the modeled time $p + (1 - p)/r$ reproduces the measured times to within 6%, so the protected share of the work sets how much of that speedup a mask retains.

Table 27: Block-GEMM time of one GPT-2 124M continued-pretraining step relative to BF16, measured with native kernels on one RTX 5090 Laptop GPU and modeled as $p + (1 - p)/r$ for a protected fraction p, both excluding operand quantization and non-GEMM time. Retained speedup is the ratio of the policy speedup factor to the all-low speedup factor, both relative to BF16; masks as in Table 28.

Setup		Measured time	Retained speedup (%)			
Policy	Protected work (%)	NVFP4	Measured NVFP4	Measured FP8	Modeled $r = 2.85$	Modeled $r = 4$
All-low core	0.0	0.211	100	100	100	100
Selected forward GEMM, first selection	0.9	0.218	97	99	98	97
Selected forward GEMMs	2.8	0.233	90	96	95	92
Selected forward GEMMs	5.6	0.256	82	93	91	86
Last block in BF16	8.3	0.276	76	91	87	80
Selected forward GEMMs	11.1	0.304	69	87	83	75
Selected forward GEMMs	16.7	0.353	60	81	76	67
Last two blocks in BF16	16.7	0.342	62	83	76	67
First and last blocks in BF16	16.7	0.342	62	83	76	67
Selected forward GEMMs	22.2	0.401	53	77	71	60
Selected forward GEMMs, target level	29.6	0.465	45	71	65	53
Fprop path in BF16	33.3	0.498	42	69	62	50
Backward path in BF16	66.7	0.713	30	57	45	33
BF16	100.0	1.000	21	45	35	25

E.8 Compute-Budget Allocation in Continued Pretraining

E.8.1 Allocation under a Compute Budget

Scoring forward and backward passes on one scale concentrates protection where bending originates: Before training, the Fprop pass of one attention layer removes more bending than the BF16 backward path with 72 times the GEMM work of that Fprop pass, and after continued pretraining, masks over individual GEMM passes close more of the loss gap than block protection of equal work and than the backward path at every tested budget.

Before Training. We repeat the continued-pretraining protocol of Appendix D.7.1 with the NV GEMM core, which quantizes both operands of every Fprop, Dgrad, and Wgrad GEMM in the four linear layers of each GPT-2 block (Sec. 2). At the pretrained weights, executing every Dgrad and Wgrad GEMM in BF16, two thirds of the GEMM work, removes 2% of the held-out bending, whereas executing every Fprop GEMM in BF16, one third of the work, removes 94% (Figure 3(d)), as the operand-level ordering of Appendix E.4 anticipates.

Local error points the other way. The Dgrad GEMM outputs carry the largest local error, an NRMSE of 0.14 on average against 0.09 for Fprop (Figure 11), so a reconstruction-error rule spends the available budget on Dgrad passes, whose protection leaves the bending essentially unchanged at every budget up to 11% of the work (Figure 3(d)).

Bending-guided precision allocation to individual GEMM passes scores every Fprop, Dgrad, and Wgrad pass of the attention and MLP GEMMs of every layer by the update bending D_{u} removed per unit of GEMM work and exploits this asymmetry: Per pp of work, protecting a single Fprop pass removes 2.1% of the update bending on average, and a single Dgrad or Wgrad pass 0.05%. The first selected action, the Fprop pass of the layer-2 attention GEMMs, 0.9% of the work, removes 31% of the held-out bending, more than the whole BF16 backward path with 72 times the work, and the path descends smoothly, protecting the Fprop passes of early and middle layers first.

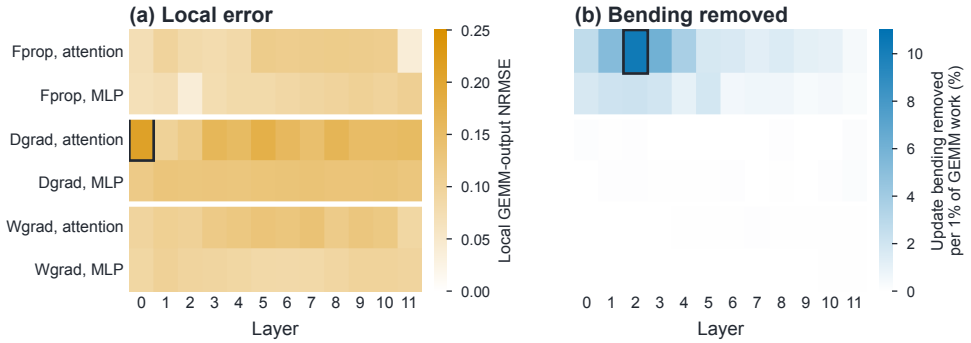


Figure 11: Local error and bending removed for every pass of the NV GEMM core of GPT-2 at the pretrained weights. (a) Local GEMM-output NRMSE, the score of reconstruction-error selection. (b) Calibration update bending removed by executing only that GEMM pass in BF16, per pp of protected GEMM work, the first-step score of bending-guided precision allocation to individual GEMM passes. Boxes mark the pass that each score ranks first.

Block protection in the spirit of the NVFP4 pretraining precision policy of NVIDIA (2025), which executes the first two and the last eight of the 62 blocks of a 12B-parameter hybrid model, 16% of the hybrid model's linear layers, in BF16, removes little: At 16.7% of the work, executing the last two blocks or the first and last blocks in BF16 removes 7% and 2% of the bending, against 70% for the mask over individual GEMM passes of the same work. The same asymmetry holds at the random initialization used for from-scratch selection, where the first two selections of individual GEMM passes, 2.8% of the work, match the update bending of the BF16 backward path (Appendix E.7).

After Training. Continued pretraining confirms what the probe measured (Figure 3(e) and Table 28 in Appendix E.8). The all-low core ends 0.077 above BF16 training in validation loss. Executing the whole backward path in BF16, two thirds of the GEMM work, closes 2.7% of this gap, and the reconstruction-error mask, which executes six attention Dgrad passes in BF16, closes 0.6%, whereas executing every Fprop GEMM in BF16, one third of the work, closes 99.3%.

Allocation to individual GEMM passes spends the budget on forward passes, and the loss-gap closure of the selected masks rises with the budget as the bending path predicts: 6.5% at 0.9% of the work, 23.6% at 5.6%, 38.3% at 11.1%, 51.7% at 16.7%, and 66.1% at 22.2%. At equal work, the masks over individual GEMM passes close more of the gap than block protection on every seed: 23.6% against 10.4% for allocation to operation groups at 5.6% of the work, and 51.7% at 16.7% against 19.8% and 17.8% for the NVIDIA-style masks that execute the last two blocks or the first and last blocks in BF16, a closure that the mask over individual GEMM passes at 11.1% of the work already exceeds.

Along the path, the calibration bending of the raw gradient falls below the floor of this core, $\tau = 0.0036$, at 29.6% of the work, where the path protects every Fprop pass except the MLP passes of the last two blocks; the frozen mask at this target-level budget closes 85.5% of the gap and learns weights within 0.004 of BF16 training without quantizers on average over seeds, and the complete Fprop path, which adds those two passes, closes 99.3%.

Across the 16 trained compute masks, the bending measured before training ranks the final gap with Spearman correlation 0.94 for the raw gradient and 0.92 for the AdamW update, against 0.50 for the summed local reconstruction error and 0.59 for the protected work alone; the whole-block masks close somewhat more of the gap than the measured bending of those masks predicts, yet less than masks over individual GEMM passes with less work.

E.8.2 Protocol and Complete Results

This subsection extends the budget-matched study of Appendix D.7 from boundary quantizers to GEMM-operand quantization. The NV core of Appendix E.2 quantizes both operands of every

Fprop, Dgrad, and Wgrad GEMM in the four linear layers of each GPT-2 block, with E2M1 values, 16-value blocks, E4M3 block scales, and an FP32 tensor scale, and the embeddings, normalization, attention scores, softmax, LM head, and optimizer stay on the BF16 reference path. The protocol is the decoder continued-pretraining protocol of Appendix C.3: pretrained GPT-2 124M, one pass over 8,192 WikiText-103 training sequences of 256 tokens with batch 8, AdamW with learning rate 5×10^{-5} and weight decay 0.01, a linear schedule with 3% warmup, and global clipping at 1.0, with seeds 17, 23, and 31 on one NVIDIA GeForce RTX 5090 Laptop GPU. Every policy of a seed sees the same data order and dropout masks, and the BF16 baseline uses the same custom-GEMM path with every operand in BF16. As in the boundary study, the calibration and held-out batches are drawn from the first 32 of the 512 validation blocks on which the final losses are evaluated.

Selection at the Pretrained Weights. The variants of bending-guided precision allocation that select individual GEMM passes or operation groups trace the corresponding bending–cost paths at the pretrained weights on eight 2×256-token calibration batches from the WikiText-103 validation split, as in Appendix E.6, but score the zero-moment AdamW update bending D_{u} under the continued-pretraining recipe, with unit clipping and weight decay 0.01; every mask is then evaluated on eight disjoint held-out batches.

The allocation path over individual GEMM passes protects the Fprop passes of the attention GEMMs in layers 2, 1, 3, 5, 6, and 4 first, 0.93% of the GEMM work each, and then the MLP Fprop passes of layers 1 and 2; tracing the path to 11.1% of the work takes 642 evaluations of 1.4 seconds each, and continuing the path until the selected mask's calibration D_{g} falls below the floor $\tau = 0.0036$ of this core, the bending of the BF16 reference path of the core relative to FP32 on the same batches, at 29.6% of the work, takes 678 more (Appendix C.1). All 22 actions on the path are Fprop passes.

Figure 11 maps the single-pass scores at the start of the path: Protecting one Fprop pass removes 2.1% of the calibration update bending per pp of GEMM work on average, and protecting one Dgrad or Wgrad pass 0.05%, whereas the local GEMM-output NRMSE, the score of reconstruction-error selection over individual GEMM passes, averages 0.141 for the Dgrad, 0.104 for the Wgrad, and 0.089 for the Fprop passes, so that selector protects Dgrad passes first. Table 28 lists the held-out update and raw bending of the trained masks next to the corresponding training outcomes, and Figure 3(d) shows every selector.

Training. Table 28 reports the masks that separate the selectors, each trained with three seeds. The masks over individual GEMM passes at 0.9%, 2.8%, 5.6%, and 7.4% of the work are the budgeted selections of the path at budgets of 1.4%, 2.8%, 5.6%, and 8.3%, and the masks at 11.1%, 16.7%, and 22.2% are prefixes of the same path; the reconstruction-error mask comes from knapsack selection over individual GEMM passes at 5.6%, and the NVIDIA-style masks execute the last block, the last two blocks, or the first and last blocks in BF16.

Every run stays finite, and the global gradient norm exceeds the clipping threshold at every optimizer step of every run. The all-low core ends 0.077 above BF16 training in validation loss and 0.034 above the BF16 loss without quantizers. Closure follows the calibration bending: The Fprop path closes 99.3%, the target-level mask 85.5%, the masks over individual GEMM passes 6.5%, 12.8%, 23.6%, 29.1%, 38.3%, 51.7%, and 66.1% in order of increasing protected work, the group-level mask 10.4%, the NVIDIA-style masks 9.1%, 19.8%, and 17.8%, the backward path 2.7%, and the reconstruction-error mask 0.6%. On every seed, the first selection of an individual GEMM pass closes more of the gap than the backward path and the reconstruction-error mask, the mask over individual GEMM passes at 5.6% of the work closes more than allocation to operation groups at the same work and the last block at 8.3%, and the masks over individual GEMM passes at 11.1% and 16.7% close more than both NVIDIA-style masks at 16.7%.

The whole-block masks close more of the gap than the measured bending of those masks suggests: Over all 16 trained masks, calibration D_{g} ranks the final gap with Spearman correlation 0.94, and over the 13 masks without whole-block protection with 0.995; the summed local reconstruction error reaches 0.50 and the protected work 0.59. Without quantizers, the Fprop path learns weights within 0.001 of BF16 training and the target-level mask within 0.004, on average over seeds.

Summary. In continued pretraining with the NV GEMM core, executing every Fprop GEMM in BF16 closes 99.3% of the loss gap with one third of the GEMM work, whereas the BF16 backward

Table 28: Compute-budget allocation in GPT-2 continued pretraining with the NV GEMM core, three seeds per mask. Work is the protected share of the block GEMM work, and D_{u} and D_{g} are the held-out update and raw bending at the pretrained weights, before training. Losses are means over seeds, and gap closed is mean $\pm$ standard deviation.

Setup		Cost	Before training		Outcome		
Policy	Protected passes	Work (%)	D_{u}	D_{g}	Val. loss	Unquantized loss	Gap closed (%)
BF16	every pass	100.0	0	0	3.281	3.281	–
All-low core	none	0.0	0.403	0.0270	3.358	3.316	0.0
Selected forward GEMMs	attention Fprop, layer 2	0.9	0.367	0.0186	3.353	3.314	6.5 $\pm$3.0
Selected forward GEMMs	attention Fprop, layers 1 to 3	2.8	0.334	0.0149	3.348	3.310	12.8 $\pm$0.6
Selected forward GEMMs	attention Fprop, layers 1 to 6	5.6	0.306	0.0129	3.340	3.304	23.6 $\pm$2.2
Group-level	attention, layers 1 and 2	5.6	0.352	0.0176	3.350	3.312	10.4 $\pm$2.1
Individual GEMM passes, reconstruction error	attention Dgrad, layers 0, 3, 5, 6, 8, 11	5.6	0.401	0.0271	3.358	3.315	0.6 $\pm$2.1
Selected forward GEMMs	attention Fprop, layers 1 to 6; MLP Fprop, layer 1	7.4	0.293	0.0118	3.336	3.303	29.1 $\pm$0.7
Last block in BF16	attention and MLP, layer 11	8.3	0.397	0.0259	3.351	3.313	9.1 $\pm$1.9
Selected forward GEMMs	attention Fprop, layers 0 to 6 and 8; MLP Fprop, layers 1 to 2	11.1	0.266	0.0104	3.329	3.299	38.3 $\pm$1.7
Selected forward GEMMs	attention Fprop, layers 0 to 6 and 8; MLP Fprop, layers 0 to 4	16.7	0.223	0.0081	3.318	3.295	51.7 $\pm$2.5
Last two blocks in BF16	attention and MLP, layers 10 to 11	16.7	0.388	0.0252	3.343	3.310	19.8 $\pm$1.8
First and last blocks in BF16	attention and MLP, layers 0 and 11	16.7	0.377	0.0263	3.345	3.309	17.8 $\pm$1.1
Selected forward GEMMs	attention Fprop, layers 0 to 9; MLP Fprop, layers 0 to 6	22.2	0.185	0.0055	3.307	3.291	66.1 $\pm$1.1
Selected forward GEMMs, target level	Fprop except MLP, layers 10 and 11	29.6	0.147	0.0033	3.292	3.285	85.5 $\pm$0.8
Fprop path in BF16	every Fprop GEMM	33.3	0.128	0.0015	3.282	3.282	99.3 $\pm$1.5
Backward path in BF16	every Dgrad and Wgrad GEMM	66.7	0.385	0.0264	3.356	3.316	2.7 $\pm$3.2

path closes 2.7% with two thirds and the reconstruction-error mask, which protects Dgrad passes, 0.6%. Masks over individual GEMM passes protect only Fprop passes and close more of the gap than equal-work group and block protection on every seed: 23.6% against 10.4% for operation groups at 5.6% of the work and 51.7% against at most 19.8% for NVIDIA-style block masks at 16.7%. The target-level mask, at 29.6% of the work, closes 85.5% and learns weights within 0.004 of BF16 without quantizers, and calibration bending ranks the 16 trained masks with Spearman correlation 0.94, against 0.50 for the summed local reconstruction error.

E.9 Full-Budget From-Scratch Trajectories

At each diagnostic checkpoint, we compare BF16 and FP4 gradients on the candidate's current weights, then apply both to the same candidate AdamW moments and parameters. The common-state comparison yields raw, clipped, and common-state update geometry without conflating the instantaneous numerical intervention with a separately trained BF16 state. The hypothetical update is never applied. Candidate policies are sharded across processes after a protocol-identical BF16 baseline has

Table 29: Two-seed GPT-2 Medium BF16 learning rate selection on WikiText-103 validation at the full trajectory budget in Table 20.

Learning rate	Validation loss
0.7×10^{-3}	3.32020 ± 0.00765
0.8×10^{-3}	$\mathbf{3.31400 \pm 0.00471}$
0.9×10^{-3}	3.32426 ± 0.00218

Table 30: Full-path fixed-state scale confirmation across the GPT-2 family; 124M, 355M, and 1.558B correspond to GPT-2, GPT-2 Medium, and GPT-2 XL, respectively. Paired-batch counts follow Table 20.

Initialization	Parameters	Core	Raw D_{g}	Angle (°)	R_{norm}	D_{u}	Policy loss
Pretrained	124M	MX	0.2271	39.19	0.9779	0.6785	4.5455
		NV	**0.0234**	**12.38**	1.0095	**0.3841**	3.9372
	355M	MX	0.1514	31.48	0.9466	0.6812	3.8891
		NV	**0.0257**	**12.39**	1.0396	**0.3825**	3.5198
	1.558B	MX	0.2386	40.19	0.9275	0.4675	3.5710
		NV	**0.1046**	**26.30**	1.0047	**0.3194**	3.5602
Random	1.558B	MX	0.1493	31.70	0.9633	0.4103	11.2192
		NV	**0.1077**	**26.84**	0.9743	**0.3430**	11.1791

completed; cross-process realized-state update vectors are unavailable, so the analysis uses the observed common-state comparisons. The final BF16 replay column evaluates each candidate's learned weights with BF16 GEMM operands and separates evaluation-time quantization from accumulated trajectory damage.

The 124M WikiText-103 trajectories of the BF16 baseline and of the policy with selected forward GEMMs in BF16 of Appendix E.6 are reported in Table 31, beside the matched actual-kernel FP8 baseline. Every reported run remains finite with zero detected loss spikes, and BF16 replay improves the endpoint of the policy with selected forward GEMMs in BF16 by only 0.0054, implying that most of the gap between that policy and BF16 is stored in the learned state.

Summary. Every reported 124M trajectory remains finite with no detected loss spike, and BF16 replay recovers only 0.0054 of the 0.034 loss gap between the policy with selected forward GEMMs in BF16 and BF16 training, so most of that gap is stored in the learned weights rather than added by quantization at evaluation.

E.10 Scale-Up to 355M and 1.56B Parameters

Table 29 tunes the BF16 learning rate specifically for GPT-2 Medium. Across two seeds, the three candidate rates yield six runs, each of which receives 8,192 updates and 134.22M input tokens. The selected 8.0×10^{-4} is an interior minimum bracketed by worse neighbors; every run has zero non-finite steps and zero detected loss spikes. Table 31 reports the resulting 355M trajectories.

Table 30 extends the fixed-state geometry. The 355M factorial uses four complete paired tuples, and the 1.56B endpoints use 16 pretrained and eight randomly initialized paired batches. NV reduces both D_{g} and D_{u} in every model and state cell. At 1.56B pretrained initialization, the paired MX-minus-NV differences are 0.13403 in raw D_{g} with a 95% paired-bootstrap interval of $[0.11675, 0.15472]$, $13.88°$ in angle $[12.61°, 15.38°]$, and 0.14814 in D_{u} $[0.13880, 0.15842]$. Immediate loss does not separate these pretrained endpoints: The immediate-loss interval $[-0.03625, 0.06083]$ crosses zero.

At random initialization, the corresponding geometry differences remain positive: 0.04154 $[0.03788, 0.04462]$, $4.86°$ $[4.46°, 5.19°]$, and 0.06729 $[0.06475, 0.06970]$. Thus, the scale result is not an artifact of public pretrained weights. At 1.56B, the evidence is diagnostic-batch geometry from 24 paired batches; trajectory evidence is reported at 124M and 355M.

Table 31: Matched from-scratch WikiText-103 trajectories under the three-seed budget of Table 20; actual-kernel rows use E4M3 `_scaled_mm`, and the NV core is simulated. BF16 replay evaluates each trained checkpoint in BF16, Hits counts runs whose final loss is at most 1% above the paired BF16 loss, and Clip is the percentage of steps with active gradient clipping.

(a) Prospective step-0 geometry

Policy	Execution	Raw D_{g}	D_{u}
GPT-2 124M			
BF16 baseline	BF16	0.0000	0.0000
Actual-kernel FP8-E4M3	actual	0.0040	0.0512
NV core, selected forward GEMMs	simulated	0.0312	0.1517
GPT-2 Medium 355M			
BF16 baseline	BF16	0.0000	0.0000
Actual-kernel FP8-E4M3	actual	0.0047	0.0620
NV core, selected forward GEMMs	simulated	0.0342	0.1728

(b) Final trajectory outcome and stability

Policy	Test loss	BF16 replay	Gap to BF16	Hits	Clip (%)
GPT-2 124M					
BF16 baseline	3.2313 $\pm$0.0086	3.2313 $\pm$0.0086	0	3/3	0.73
Actual-kernel FP8-E4M3	3.2336 $\pm$0.0110	3.2326 $\pm$0.0105	0.0023 $\pm$0.0038	3/3	0.74
NV core, selected forward GEMMs	3.2655 $\pm$0.0032	3.2601 $\pm$0.0038	0.0342 $\pm$0.0055	1/3	0.71
GPT-2 Medium 355M					
BF16 baseline	3.1313 $\pm$0.0011	3.1313 $\pm$0.0011	0	3/3	1.22
Actual-kernel FP8-E4M3	3.1351 $\pm$0.0019	3.1344 $\pm$0.0017	0.0038 $\pm$0.0010	3/3	1.24
NV core, selected forward GEMMs	3.1616 $\pm$0.0021	3.1577 $\pm$0.0032	0.0303 $\pm$0.0032	2/3	1.19

The replicated 355M factorial strengthens and qualifies the 124M attribution. E4M3 scaling is again the dominant raw-gradient effect: The paired main effect of E4M3 scaling is -0.12386 with interval $[-0.15999, -0.09186]$. The raw block-size main effect is -0.00186 $[-0.01204, 0.00554]$; the interaction is -0.00250 $[-0.02252, 0.02965]$. Neither separates from zero with four tuples.

After AdamW, the scale effect is -0.25835 $[-0.26871, -0.24800]$. The block and interaction effects are -0.04027 $[-0.04771, -0.03212]$ and -0.03802 $[-0.04282, -0.02985]$, respectively. Block size is therefore optimizer- and state-dependent; the replicated endpoint comparison is more stable than a universal block-only rule.

Summary. The NV core reduces both D_{g} and D_{u} relative to MX in every model and state cell up to 1.56B parameters, at pretrained and random initialization alike, although immediate loss does not separate the pretrained 1.56B endpoints. In the replicated 355M factorial, E4M3 scaling remains the dominant raw-gradient effect, whereas the block-size effect appears only after AdamW, so block size is optimizer- and state-dependent.

E.11 Matched Actual-Kernel FP8 Accuracy Baseline

The actual-kernel FP8 control, which quantizes only the operands of the block GEMMs, shows that low precision need not reproduce the severe behavior of all-low boundary quantization. Table 31 applies actual-kernel E4M3 scaled GEMMs to every GPT-2 block GEMM in from-scratch training trajectories. The final test loss is 3.2336 at 124M and 3.1351 at 355M, compared with 3.2313 and

3.1313 for BF16. All three runs at each scale meet the paired BF16-plus-1% target. The step-0 geometry is consistent with these outcomes: D_{g} is 0.0040 and 0.0047, compared with 0.0312 and 0.0342 for the simulated FP4 core with selected forward GEMMs in BF16. An auxiliary check isolates the kernel path: In the reported microbenchmarks, scaled GEMMs attain the same relative output error as simulated-precision FP8 matmul while running 26 to 32 times faster (Table 19). Together, these results support the FP8 path as a well-conditioned low-precision baseline; the remainder of this subsection gives the paired-bootstrap intervals.

The actual-kernel FP8 baseline consists of six runs that apply E4M3 scaled GEMMs to the same GPT-2 block Fprop, Dgrad, and Wgrad scope at 124M and 355M parameters. The six runs use the independently selected BF16 learning rates without FP8-specific retuning, the same per-seed initialization and data order, and the complete 8,192-step, 134.22M-token budget. Table 31 isolates the numerical path while retaining the tuned BF16 baseline. At both scales, the simulated-FP4 comparison executes selected forward GEMMs in BF16; for 355M, the 124M mask over individual GEMM passes is mapped by sending source layer i to target layers $2i$ and $2i+1$, without target-scale selector tuning. These protected paths define a simulated-FP4 accuracy baseline. Equal-format and actual-kernel throughput comparisons require an actual FP4 kernel path.

For the trajectories in Table 31, actual-kernel FP8 at 124M finishes at 3.23358 ± 0.01097, only 0.00229 above paired BF16; the corresponding 95% paired-bootstrap interval $[-0.00124, 0.00629]$ includes zero, one seed is numerically better than BF16, and all three meet the paired 1% target. Actual-kernel FP8 lowers the loss relative to the simulated-FP4 policy with selected forward GEMMs in BF16 by 0.03193 $[0.02685, 0.04142]$ on all three paired seeds. At 355M, actual-kernel FP8 reaches 3.13513 ± 0.00192. The 0.00383 BF16 gap at 355M has a three-seed interval of $[0.00285, 0.00485]$, which excludes zero; nevertheless, all seeds remain within 1%, and FP8 lowers the loss relative to the simulated-FP4 policy with selected forward GEMMs in BF16 by 0.02649 $[0.02321, 0.03049]$ on every seed. Thus, the tuned BF16 runs remain the high-precision baselines, while actual-kernel FP8 supplies a substantially stronger low-precision baseline than the simulated FP4 policies.

The prospective geometry in Table 31 agrees with the endpoint ordering. Actual-kernel FP8 gives step-0 D_{g} and D_{u} of 0.00398 and 0.05115 at 124M and of 0.00475 and 0.06197 at 355M, compared with 0.03115 and 0.15166 at 124M and 0.03421 and 0.17278 at 355M for the simulated-FP4 policy with selected forward GEMMs in BF16. BF16 replay removes only 0.00095 and 0.00073 from the two FP8 endpoint losses; the remaining replay gaps to the independently trained BF16 states are 0.00133 and 0.00311. All six trajectories complete with zero non-finite steps and zero detected loss spikes. The step-0 BF16 replays of all six trajectories differ from the stored references by at most 4.27×10^{-14}. These results validate the training controls and provide a hardware-executed low-precision anchor. Measuring optimized FP8 and FP4 speedups remains a systems task.

Summary. Actual-kernel E4M3 scaled GEMMs in every GPT-2 block finish within 0.0023 of the paired BF16 loss at 124M and 0.0038 at 355M and meet the BF16-plus-1% target on all six runs, whereas the simulated FP4 policy with selected forward GEMMs in BF16 ends 0.034 and 0.030 above BF16, meets the target on three of the six runs, and has higher loss than actual-kernel FP8 on every paired seed. The actual-kernel FP8 runs start with step-0 D_{g} and D_{u} several times smaller than the FP4-core values, so the same diagnostics that flag FP4 identify the FP8 path as well conditioned before training.

E.12 Statistical Protocol, Artifact Audit, and Next Tests

All confirmatory comparisons are paired by batch or seed. Strictly positive factorial outcomes are also analyzed on a logarithmic scale. We use 10,000 paired-bootstrap resamples. With four fixed-state tuples and three trajectory seeds, these intervals quantify finite-sample stability; trajectory results also report seed counts. Site-wise p-values are omitted because these sample sizes do not support a reliable multiple-testing analysis. A mixed-effects model is likewise poorly identified, so the appendix reports transparent paired effects. Reported analyses are restricted to protocol-qualified run identifiers, and the artifact audit records excluded records, source identities, and protocol checks.

The implementation is split by responsibility across format arithmetic, custom-autograd GEMMs, actual-kernel FP8 dispatch, pass actions, evaluation, selection, trajectory training, mask transfer,

cached-corpus loading, plotting, statistical analysis, merging, and artifact checking. Concurrent result collection uses cross-process locking, schema union, de-duplication, and atomic replacement.

The audit verifies expected protocol-cell counts, seed and policy coverage, finite outcomes, exact token and split metadata, pass-equivalent budgets, reference-run linkage, calibration and held-out ordering, and source and tokenizer identities. Step-0 BF16 replay differs from the stored reference value by at most 4.27×10^{-14}. Collected records are immutable and keyed by run-level source hashes.

Reproduction Check. The audited allocation used Python 3.10.20, PyTorch 2.5.1 with CUDA 12.4, NumPy 2.2.6, pandas 2.3.3, Transformers 4.53.1, and Datasets 3.2.0. A 128-test unit suite covers format arithmetic, pass isolation, selector budgets, optimizer-update geometry, frozen-mask transfer, and result-integrity checks. Reproduction first collects trajectories with explicit actions, expected costs, selector labels, and transfer annotations, then regenerates the paired statistical summaries and validates the complete experimental grids.

The audit rejects incomplete grids, source or data identity mismatches, protocol leakage, non-finite intermediate metrics, unequal pass budgets, lossy merges, or stale summaries. Ordinary checks are read-only; refreshing audit metadata or merging result collections requires an explicit action so that provenance remains consistent with the canonical records.

The experimental design points directly to the following next tests.

The GH200 has no hardware FP4 endpoint. These experiments quantify FP4 numerical mechanisms, whereas an FP4-capable platform is required to measure actual-kernel accuracy, speed, memory reduction, communication cost, and time-to-target. The actual-kernel E4M3 control uses current per-tensor scaling through a private PyTorch `_scaled_mm` endpoint and serves as a coverage- and budget-matched accuracy and geometry anchor; a fused public implementation is the appropriate setting for optimized throughput and memory measurements.

The 124M and 355M trajectories use three random initializations and 134.22M input tokens per policy and seed. This fixed budget isolates the precision intervention, so compute-optimal convergence and per-policy low-precision hyperparameter tuning are separate experiments; other corpora, web-scale pretraining, and downstream benchmark suites are natural extensions. The reported 355M trajectory with selected forward GEMMs in BF16 transfers a single 124M-derived mask by a frozen depth rule, with selector recalibration reserved for a scale-specific study. Selector rankings in from-scratch training and BF16 parity of the simulated FP4 cores therefore remain open empirical questions. The 1.56B evidence comprises 16 pretrained and eight random-init paired fixed-state batches, so a multi-seed 1.56B trajectory is the next scale-up experiment. The model-scale checks currently cover only the dense GPT-2 family and one tokenizer; architecture transfer, mixture-of-experts routing, and production-scale FP4 training remain open model axes.

The measured local-residual LQA identity applies directly only to isolated Dgrad; the identity uses the realized residual $d_k(v_k^{\mathrm{ref}}; \pi_k)$ and does not assume a linear quantizer, so Fprop state propagation and Wgrad direct parameter-gradient readout must be analyzed separately. Single-intervention propagation is likewise a local mechanism probe, and the selectors measure the joint effect of all selected protections by executing each complete mask. Each candidate shard has optimizer moments specific to that shard, so common-state BF16 replay is available at each candidate state; cross-process realized-update vectors are unavailable, and the analysis uses only observed vectors.

F Discussion

F.1 Evidence and Practical Tradeoffs

The experiments support the propagated-error view through a set of connected patterns. Forward accuracy and the two coordinates of gradient geometry can diverge: The ImageNet precision sweep contains cases where top-1, descent projection, and angular distortion order policies differently. The block-scaled rows of the ImageNet precision sweep recover accuracy without a monotonic improvement in D_{g}, showing that a scaling choice can change a local error and the parallel and orthogonal components of the propagated error in different proportions. The source-of-error ablation identifies the relevant operator: The dominant distortion in pretrained ResNet-18 comes from forward-

induced E_k and parameter-Jacobian shifts, whereas fixed-forward backward quantization contributes much less. Figure 6 attributes 76% of that forward-induced shift to flipped ReLU masks and 3% to quantized Wgrad inputs, and Figure 2 shows that identical 2.6% local errors are propagated into shifts that differ by 17× across ResNet-18 sites and 32× across GPT-2 sites.

Bending-guided precision allocation uses the distinction between gradient norm and direction to decide where to allocate high precision efficiently within a given budget. In GPT-2 continued pretraining, the calibration path predicts both the minimum viable memory budget and the budget at which bending approaches the BF16 level. Calibration bending ranks final losses with a Spearman correlation of 0.95, compared with 0.35 for local reconstruction error. The selected masks learn weights with BF16-level loss while using less high-precision activation-cache memory than a production-style precision policy. For the NV GEMM core, protecting forward passes closes 99.3% of the loss gap, compared with 2.7% for backward protection. At 16.7% of the GEMM work, masks over individual GEMM passes close 52% of the gap, compared with at most 20% for NVIDIA-style block protection, while still running the block GEMMs 2.8 times faster than BF16. Bending-guided precision allocation also improves low-precision fine-tuning of four vision backbones.

Precision decisions are also nonadditive. On the calibration batch, applying the classifier and residual masks together produces more angular distortion than the sum of the individual scalar distortions of the two masks, while other pairs are sub-additive in D_{g}. These scalar interactions alone do not identify the signs of the orthogonal inner products Γ_{ij}. The direct measurement in Figures 5(b) and 5(c) finds nearly all pairs positively aligned, with the classifier and LM-head contributions nearly orthogonal to the rest. The activation-cache footprints reported with the protection tables add the deployment constraint: A group-count budget can hide large differences in protected footprint, so Appendix D.7 compares every selector at matched footprint.

The fine-tuning results carry these gradient-geometry signals beyond the calibration setting. Table 17 shows that a protected set chosen using ImageNet angular geometry transfers to CIFAR-100 fine-tuning and recovers about 80% of the all-low FP8 top-1 drop in the ResNet-18 transfer evaluation.

The modern-backbone runs make the tradeoff more concrete. Cache-cost bending-guided precision allocation removes bending per unit protected footprint, and, as Table 18 shows, recovers most of the all-low FP8 gaps of ConvNeXt-T, Swin-T, and ViT-B/16, whose FP32 baselines reach 85.66% to 88.72%, while using high precision for 11% to 25% of the eligible boundary footprint. A practical implication is a two-stage rule: Use site criticality to identify precision-sensitive paths, then select the deployed mask on the joint accuracy–geometry–cost frontier. The budgeted allocation of Sec. 5 makes both stages explicit: Bending removed per unit cost is scored before training, the greedy path traces the bending–cost frontier together with the minimum viable budget and the budget at which bending reaches the BF16 floor, and the deployed mask is selected within the available precision allocation budget.

The decoder-LM study and the accompanying diagnostic checks provide three kinds of support for the claims in this paper. First, the bending diagnostic transfers from vision to a WikiText-103 decoder-only LM training setting, with LAMBADA serving as an auxiliary downstream proxy. Second, under NVFP4-style boundary quantization, bending-guided precision allocation recovers almost all of the loss gap, whereas reconstruction-error selection, which leaves the LM-head boundary in low precision, closes about a fifth of that gap, and a high-precision backward path does not close the loss gap reliably. Third, budget, trajectory, and decomposition checks show where the signal is concentrated. Together, these results show that bending-based scoring provides a direct and grouping-flexible signal for locating where low precision rotates the optimizer gradient.

F.2 Systems Coverage and Generalization

The evidence spans two coordinated settings with distinct tensor targets. Diagnostic boundary quantizers quantize module-output activations and incoming adjoints to expose propagation mechanisms, while actual-kernel FP8 paths quantize the operands of every Fprop, Dgrad, and Wgrad GEMM. The block-128 experiments test scaling granularity. Delayed scaling, fused nonlinear kernels, FP8 accumulators, distributed reduction, and optimizer-state precision are additional systems choices to which the diagnostic framework can be applied. The auxiliary kernel check shows that the actual-kernel FP8 path provides the expected GEMM behavior before the path is embedded in full GPT-2 pretraining trajectories.

The matched actual-kernel FP8 trajectories in Appendix E.11 apply E4M3 scaled GEMMs to every block GEMM of GPT-2 at 124M and 355M parameters and stay within the paired BF16-plus-1% target across all three runs at each scale. The trajectories' step-0 D_{g} and D_{u} values are several times smaller than the corresponding values under the simulated FP4 policies, so the same diagnostics that flag the FP4 cores also identify the FP8 path as well conditioned.

The mechanism taxonomy defines corresponding tests for embeddings, convolutions, normalization, softmax, optimizer state, communication, delayed production scaling, end-to-end ImageNet-1K training, routed models, recurrent models, and LLM training. The ResNet-18 and cross-backbone fine-tuning experiments give training-outcome evidence on CIFAR-100.

G LIMITATIONS

Our experiments cover several vision and language datasets and model architectures, but the current evaluation does not encompass every dataset or model family; broader evaluations can further test the generality of the findings.