Beyond Fixed Precision: Measuring and Tracking Minimum Sufficient Precision in Learning

Anonymous authors
Paper under double-blind review

Abstract

Mixed-precision training typically assigns one of a few hardware-supported formats to each layer, operator, or graph region. Such assignment rules choose among available formats without measuring the precision that a given learning computation actually requires. We introduce Minimum Sufficient Precision (MSP), the lowest precision that preserves a training step's numerical or learning-level behavior for one operator execution. We measure MSP with controlled precision sweeps across four model families and a controlled stress workload, relate its variation to numerical and learning-side factors, and use it to provision executable precision levels during training. In 56% of 120 valid comparisons the numerical and learning-level criteria select different MSPs, so numerical accuracy alone does not guarantee that training behavior is preserved; the requirement also moves with training state, and refreshing more often does not consistently improve fidelity. Precision is therefore a local, criterion-specific, and time-varying computational requirement to be measured and tracked, rather than a fixed datatype assignment.

1 Introduction

Low-precision and mixed-precision arithmetic are now standard tools for making deep network training affordable: large models are trained end to end in FP8 (Peng et al., 2023), and the wider efficiency landscape is surveyed elsewhere (Zhu et al., 2023). Training cost is a first-order concern at current model scales (Zhao et al., 2023). In this paradigm precision is a choice among a small set of hardware-supported formats, such as FP8, FP16, BF16, TF32, or FP32, which now includes block-scaled narrow formats (Rouhani et al., 2023) whose FP8 and INT8 accuracy differs by workload (van Baalen et al., 2023). The choice is expressed as an assignment: one format per layer, operator, or graph region, held for a run (Lee et al., 2023) or adjusted by a schedule. Existing methods therefore decide which available precision to use.

Those formats describe what hardware provides, not necessarily what a learning computation requires, and the two need not coincide. For some operator executions a native format supplies far more precision than the computation can use, and the surplus costs throughput; for others the format is not enough, and the arithmetic changes what the training step computes. The same operator can also need a different level after the parameters or the inputs change. Assignment rules, among them per-layer quantization and cyclic precision schedules, choose among the available formats without measuring the requirement itself. Figure 1 contrasts the datatype view with the demand view taken in this paper.

We address the question of the minimum numerical precision sufficient to preserve a given learning computation, and answer it with a measurement. Minimum Sufficient Precision (MSP) is the lowest calibrated arithmetic precision that preserves a chosen fidelity criterion for a particular operator execution. MSP is therefore local to one execution, criterion-specific, and state-dependent, because the execution itself changes during training. The criteria can be separated: a level can keep the operator output numerically close to a high-precision reference while changing what the training step does with that output. A level measured at one training state can also become insufficient later, so the problem has three

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

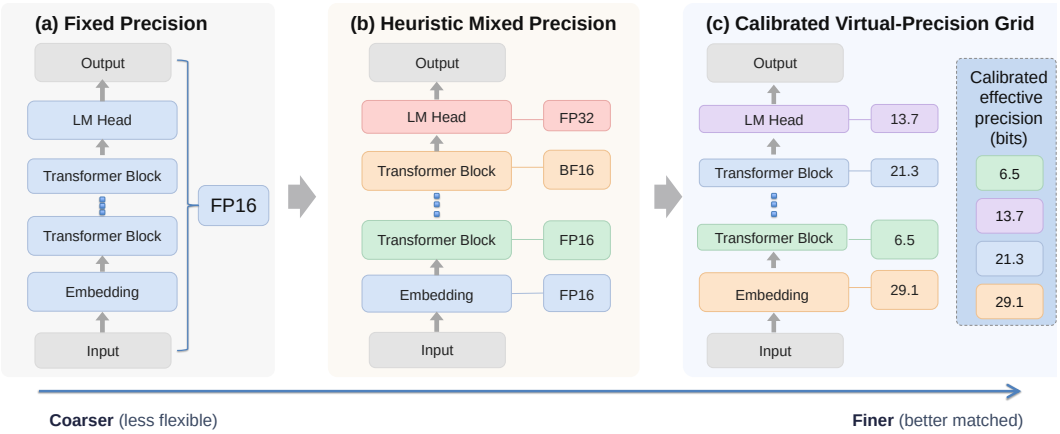

Figure 1: From fixed datatypes to a calibrated discrete precision grid. (a) A fixed datatype applies one hardware format to every operator. (b) Heuristic mixed precision assigns native formats per layer. (c) The grid used here provides a further set of executable precision operating points beyond the coarse native datatype choices.

parts: how much precision is needed, why the requirement varies, and how long a measured value stays valid.

Measured over our conditions, the requirement is heterogeneous rather than concentrated on a single level: it varies across model families, operators, inputs, and training states, and the numerical and learning-level criteria do not always select the same level. That makes precision an allocatable computational resource rather than a fixed property of a layer type, which raises the question of how to allocate it.

We therefore follow a simple loop: measure, explain, track, and provision. The first step measures the MSP of each operator execution under a controlled precision sweep; the second relates the measured variation to numerical factors and to the optimization response; the third reuses a stored value until it becomes stale and refreshes it when it does; and the fourth maps the requirement onto the closest sufficient level of a calibrated discrete grid that the backend can execute.

The paper makes three contributions:

- We define and measure operator/state-specific MSP with controlled precision sweeps and an offline MSP Oracle, and we separate numerical MSP from learning-level MSP. The two criteria disagree on the required level in 67 of the 120 valid comparisons.
- We relate MSP variation to numerical factors and to the optimization response, and we study when a measured value stops being valid, which governs how long it can be reused.
- We perform MSP-guided provisioning on a calibrated discrete precision grid and evaluate the quality–precision–runtime trade-off on the four evaluation model families; the controlled stress workload contributes to the demand, explanation, and parity analyses rather than to this evaluation.

2 Precision Demand: Definition and Measurement

We measure the precision required by one operator execution while everything else is held fixed: the checkpoint and training state, the input batch, the random state, and every non-target operation keep their reference values, and only the arithmetic precision of the target operator changes. This isolates the effect of arithmetic precision from trajectory divergence, since the data order and parameter state are identical at every level.

We refer to the computation-specific precision requirement as its precision demand. The first criterion asks whether the operator output stays numerically close to a high-precision reference:

$$D_{\text{num}}(o,t,P) = \frac{\|y_{o,t}^{P} - y_{o,t}^{\text{ref}}\|}{\max(\|y_{o,t}^{\text{ref}}\|, \tau)}.$$

Here $y_{o,t}^{P}$ is the target output at precision P, $y_{o,t}^{\text{ref}}$ the reference output, and the positive floor τ keeps the ratio defined when the reference norm is close to zero.

The second criterion asks whether the perturbation changes what the training step does with that output. We call this learning-level fidelity. It compares a downstream quantity z, such as a gradient, an update, or a short-horizon loss trajectory, against the same quantity computed at reference precision,

$$D_{\text{learn}}(o,t,P) = d(z_{o,t}^{P}, z_{o,t}^{\text{ref}}),$$

with its own tolerance and learning horizon. The two criteria need not agree, so we report their MSP values separately under the two names introduced below.

Let $\mathcal{P}$ be the ordered set of calibrated precision levels that the backend can execute. For a fidelity measure D and a tolerance ϵ, the Minimum Sufficient Precision of an operator execution is the lowest level in $\mathcal{P}$ that satisfies the selected criterion:

$$P_{o,t}^{*}(D,\epsilon) = \min\{P_i \in \mathcal{P} : D(o,t,P_i) \le \epsilon\}.$$

Using D_{num} in this definition gives the numerical MSP P_{num}^{*}, and using D_{learn} gives the learning-level MSP P_{learn}^{*}. MSP is therefore the lowest tested precision level that satisfies the criterion. Because the available levels form a finite set, we refer to the resulting value as the grid-resolved MSP. If no level satisfies the criterion, MSP is unresolved above the available precision range rather than equal to the widest level.

The offline MSP Oracle evaluates the full precision set for one operator execution, keeps the complete fidelity curve, and returns the lowest satisfying level. It is an offline measurement tool. Each executable configuration c_i is calibrated against a high-precision reference (Amestoy et al., 2024) and mapped to a single number, $c_i \mapsto P_{\text{eff}}(c_i)$, which we call its calibrated effective precision. The current implementation exposes four such levels,

$$\mathcal{P} = \{6.5,\ 13.7,\ 21.3,\ 29.1\}$$

denoting calibrated arithmetic effective precision; we refer to the four levels in increasing order as $p_1, \ldots, p_4$. The demand is therefore fine-grained but discrete, and a finer supply can reduce over-provisioning relative to the native datatype ladder (Appendix B).

Figure 2 contrasts the two practices. On the left, the standard training loop fixes one datatype for every operator: the forward pass, the backward pass, and the update all run on formats such as FP32 or BF16, whatever each operation does to the numbers. On the right, the loop attaches a measured demand to each operator execution instead, and maintains it with four components: the local demand of the executing operator, the factors that explain why the demand takes that value, the validity of a stored demand as the training state moves, and the mapping onto the discrete level that the backend can execute. The inset sketches this on the precision axis: the numerical and the learning-level criterion each mark where the computation stops being preserved, and the gap between the two marks is the disagreement that the experiments quantify. The rest of the paper follows these four components in order: how the demand is measured, what explains it, when it stays valid, and how it is provisioned.

3 Understanding and Using MSP

3.1 Why MSP Varies

The same arithmetic error can have very different effects on learning. Two sources are at work: the computation generates more or less numerical error depending on its shape and operands, and the optimization process reacts differently to that perturbation. MSP

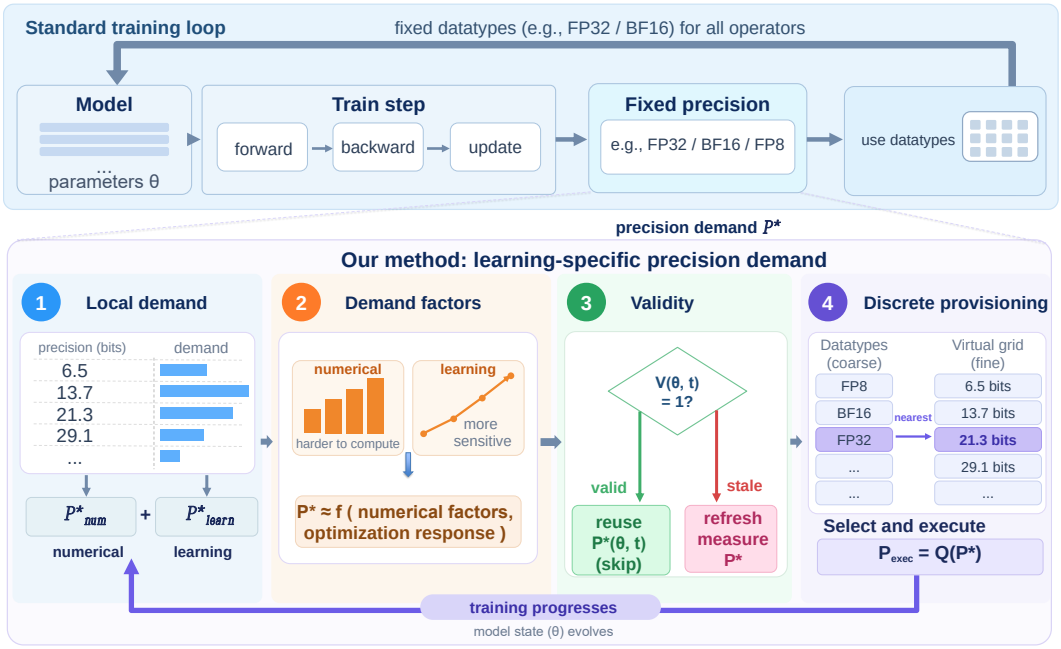

Figure 2: From fixed datatypes to learning-specific precision demand. The standard loop fixes one datatype for all operators, whereas the proposed loop measures the demand of each operator, relates it to numerical factors and the optimization response, tests whether a stored demand is still valid, and maps it to the nearest executable level of the calibrated grid.

therefore depends on both error generation and error sensitivity, and we describe each with features that are inexpensive to record alongside a measured label.

The first source is captured by numerical factors of the computation: reduction length, operand norms, dynamic range, scale ratio, and a cancellation proxy. For a matrix product $C = AB$ evaluated at precision P, floating-point analysis relates the perturbation of the result to these quantities (Fasi et al., 2023),

$$|\Delta C| \lesssim \gamma_K(P)\,|A|\,|B|,$$

where $|A|$ denotes the elementwise absolute value of the operands, K is the reduction dimension, and $\gamma_K(P)$ collects the precision-dependent rounding. Appendix D defines the cancellation proxy and the remaining arithmetic features used in the fits.

The second source is the optimization response, that is, how the training process reacts to the perturbation. We probe it with two operational proxies that divide a learning-side deviation by the numerical deviation that produced it. Writing D_{grad} and D_{update} for D_{learn} evaluated on the gradient and on the optimizer update,

$$S_{\text{grad}} = D_{\text{grad}}/D_{\text{num}}, \qquad S_{\text{update}} = D_{\text{update}}/D_{\text{num}},$$

and we record gradient and update magnitudes and residual-path responses alongside them. Appendix D lists the remaining learning-side features.

Combining the two sources gives an explanatory relation for a measured MSP, $P^* \approx f(\text{numerical factors, optimization response})$, which predicts that the required precision rises with numerical error, with optimization sensitivity, or with a stricter tolerance. This is an explanatory relation rather than a universal predictor; the experiments report how much of the observed variation it accounts for.

3.2 MSP Validity

MSP is local rather than permanent: a value measured at one training state may stop satisfying the criterion later, so a deployment system must be able to tell whether a stored value is still usable. We write P^*_{o,t_0} for $P^*_{o,t_0}(D_{\text{learn}}, \epsilon)$, the learning-level MSP measured at t_0, and define validity as

$$V(o, t_0, t) = \mathbb{I}\left[D_{\text{learn}}(o, t, P^*_{o,t_0}) \leq \epsilon\right],$$

which is one when the stored level still satisfies the learning-level criterion at a later state t. A valid value can be reused; a stale value must be refreshed. The duration over which a measured MSP remains valid defines its reuse window, and Appendix E gives the formal definition.

3.3 Tracking

Tracking reuses a previously measured MSP while it remains valid and refreshes it once it becomes stale. Reuse is inexpensive but risks acting on an outdated label, whereas refreshing is accurate but requires a sweep. The experiments compare static reuse, periodic refresh, and state-aware policies.

3.4 Discrete Provisioning

Provisioning maps a measured or predicted MSP P^* onto the nearest level the backend can execute,

$$P_{\text{provision}} = \min\{P_i \in \mathcal{P} : P_i \geq P^*\},$$

that is, the closest sufficient executable level. On the current grid this rounds the demand up to the next calibrated level and pays that level's cost; the emulation backend is described in Appendix C and the configuration-level cost model in Appendix F. Table 1 lists the policy families that can consume a measured or predicted level.

4 Experiments

The evaluation follows the measure–explain–track–provision loop: it measures the precision demand of individual operator executions, asks what explains the measured variation, checks whether the two criteria agree, maps the requirement onto the calibrated grid, and then asks how long a measured value stays valid.

4.1 Experimental Setup

All measurements run on an NVIDIA A100 PCIe 40 GB GPU with compute capability SM80. Precision is swept with a real-valued GEMM backend that we designed, which exposes the four levels of the calibrated precision grid and spans approximately 6.5 to 29.1 in calibrated effective precision. We evaluate four evaluation model families and one controlled stress workload: public distilgpt2, public bert-tiny, MLP-Mixer-B/16, Poisson DeepONet, and a controlled ill-conditioned MLP, which serves as the controlled stress workload; the families span Transformer language models, an all-MLP vision backbone, and physics-informed operator learning, for which 4-bit integer training (Xi et al., 2023) and standardised benchmarks (Hao et al., 2024) are now available. A separate high-cancellation condition contributes a numerical demand only. Appendix G gives the controlled-workload protocol. The default tolerances are a numerical deviation of 10^{-4}, a gradient/update deviation of 10^{-2}, and a loss deviation of 10^{-3}, with the strict and loose settings of Appendix G. Runtime is measured per trace shape with eight warmup runs and 31 alternating CUDA-event repeats, and summarized as an accumulated managed-GEMM median; the timed path covers the GEMM kernels only and excludes packing, scale preparation, workspace setup, and launch overhead, so the runtime axis and the fidelity axis of the cost analysis come from different measurements.

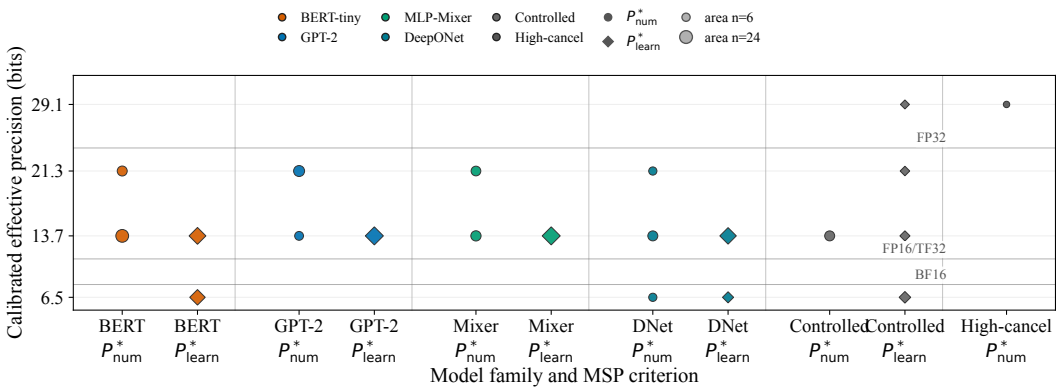

Figure 3: Precision demand across the evaluated model families and the controlled high-demand workload. Every measured requirement is placed on the calibrated grid, one column per model family and criterion, with marker area scaling with the number of conditions at that level and the native format widths marked for reference; the FP32 reference of 24 significand bits describes the demand–format gap and is not a measured assignment. Discrete markers are used because the calibrated grid has four levels and is not interpolated between them. Appendix H.2 gives the operator- and state-resolved map.

4.2 How Large Is the Precision Requirement?

We label every condition in the pooled set under both criteria and ask whether the resulting levels concentrate on one value or spread across the grid.

Figure 3 shows that MSP varies across model families, operators, inputs, and training states rather than collapsing to a single precision level. In the Transformer workloads the numerical criterion usually demands at least as much as the learning-level criterion, and the widest grid level is reached by measurement rather than only by a lower bound: the controlled stress workload contributes one measured learning-level requirement at 29.1 bits. The spread follows the error mechanism behind D_{num}: conditions differ in reduction length, operand norms, and cancellation, and the controlled stress workload sweeps those factors directly through its input regimes, so its demand reaches the widest level while the public workloads cluster at p_2 and p_3.

4.3 What Explains MSP?

We fit the explanatory relation on the pooled valid conditions, and separately on the controlled stress workload, which isolates numerical regimes rather than representing a domain benchmark.

As Figure 4 shows, numerical factors account for a substantial portion of the variation in MSP while learning-side factors provide complementary explanatory power: the numerical-only model reaches $R^2 = 0.57$, the learning-only model 0.09, and the combined model 0.69 on the pooled conditions. The combined model's gain concentrates where the two criteria disagree: there the downstream response, not the size of the numerical perturbation, decides the label, so the learning-side factors carry information that the numerical features cannot.

The relation is explanatory rather than predictive: Appendix D reports how far it transfers to MLP-Mixer-B/16.

4.4 Do the Two Criteria Agree?

We compare P^*_{num} with P^*_{learn} on every valid comparison and count the disagreements.

The left panel of Figure 5 compares the two criteria on every valid comparison. They disagree on the required level in 67 of 120 valid comparisons, and the per-workload rates of Appendix G show the disagreements spread across all model families.

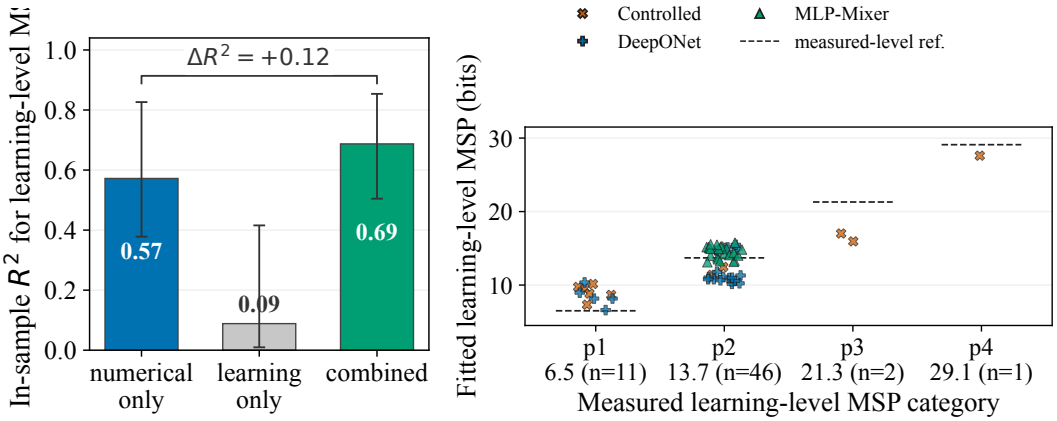

Figure 4: Numerical and learning-side factors associated with MSP. The left panel compares the explained variance of the numerical-only, learning-only, and combined models on the pooled conditions, with bootstrap 95% confidence intervals; the right panel plots the fitted learning-level MSP against the measured learning-level category, with the dashed reference at the measured level.

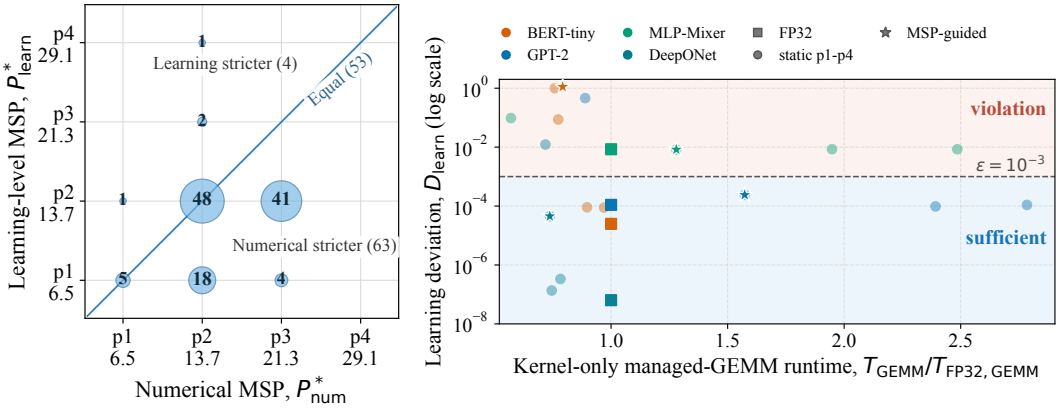

Figure 5: Numerical versus learning-level fidelity, and the cost of acting on the result. The left panel plots numerical MSP against learning-level MSP over every valid comparison: the diagonal marks equal requirements, the marker area scales with the number of conditions at that level pair, and off-diagonal conditions show that the two criteria can impose different precision requirements; they select the same level in 53 of the 120 comparisons, the numerical criterion is stricter in 63, and the learning-level criterion is stricter in 4. The right panel gives the quality–precision–runtime frontier: colour marks the workload and marker shape separates the FP32 baseline, the static levels p_1–p_4, and MSP-guided provisioning. Its horizontal axis is kernel-only managed-GEMM time normalized to the matching FP32 accumulation within each workload, and its vertical axis reports the ten-step learning deviation relative to the FP64 reference on a logarithmic scale, so the two axes come from different measurements; exact FP64-zero values are omitted because they are undefined on the log axis, and the dashed line marks the predeclared learning-loss threshold $\epsilon = 10^{-3}$, above which the region is labelled as a violation. Points are not connected, and operating points slower than FP32 are retained; Appendix G gives the timing protocol.

Numerical sufficiency does not always imply learning-level sufficiency.

4.5 What Does MSP-Guided Provisioning Cost in Quality?

We compare the fixed-precision baselines FP32 and FP64, static precision emulation, and MSP-guided provisioning on the calibrated grid, using the three-step protocol for short-

horizon behavior and a ten-step writeback for cumulative fidelity. The two Transformer workloads contribute three text windows each, and MLP-Mixer-B/16 and Poisson DeepONet one representative window each. The frontier covers the four evaluation model families; the controlled stress workload isolates numerical regimes rather than a training workload, so it enters the demand, explanation, and parity analyses rather than this one. Appendix H lists the operating points.

Short-horizon behavior is preserved on distilgpt2, where MSP-guided provisioning selects a mean level of $p = 2$ and reaches a loss deviation of 8.32×10^{-5} against the FP64 reference. Cumulative fidelity depends on the workload and the input window: over ten steps the same setting stays within 7.61×10^{-3} on distilgpt2 but reaches 1.116 on one BERT-tiny window. The arithmetic cost is likewise workload-dependent. Measured as kernel-only accumulated managed-GEMM time against the matching FP32 accumulation, MSP-guided provisioning costs 0.79 on BERT-tiny and 0.74 on Poisson DeepONet, about 21% and 26% less, while the same policy costs 1.57 on distilgpt2 and 1.28 on MLP-Mixer-B/16; the guided policy adds no kernel overhead of its own, since its ratio equals that of the static level it mostly selects. The saving is not a quality guarantee: the BERT-tiny point above reaches 1.116 while the Poisson DeepONet point stays at 4.5×10^{-5}. The violation has a specific mechanism: on that window the guided mix passes the single-execution criterion at the measured state, but its cheaper allocation, with most components at p_1 and a mean level of 1.5, accumulates ten-step drift that the uniform p_2 does not; under the same protocol the uniform p_2 stays at 0.087. The cost and fidelity results together delimit the effective scope of the method: the saving is realized where the measured demand sits at the lower levels and the criterion stays stable over the writeback horizon; where the demand sits higher, emulation at the provisioned level costs more than FP32, and where cumulative fidelity binds, provisioning must fall back to the wider static level, which stays at 8.8×10^{-5} on the same window. The end-to-end path also pays slicing, dispatch, and backward-copy work that this measurement excludes. The right panel of Figure 5 shows the resulting frontier against the fixed-precision baselines, and Table 1 places these operating points in a wider taxonomy.

MSP-guided provisioning can reduce precision while preserving short-horizon behavior, but both cumulative fidelity and the arithmetic cost advantage are workload-dependent rather than general.

4.6 When Does a Measured MSP Become Stale?

We follow stored values along four representative trajectories and compare static reuse with periodic and state-aware refresh policies.

Some workloads keep their requirement over extended periods, whereas others change with the training state. As Figure 7 in Appendix H shows, the DeepONet trunk stays stable across the window, one BERT-tiny trajectory is staged and another shifts at a single transition, and the MLP-Mixer channel moves between levels as the state evolves. The value of refreshing tracks how fast the requirement moves: in one BERT-tiny comparison, refreshing every five steps performs worse than both static reuse and refreshing every ten, because the requirement does not move between those states, so the extra sweeps add cost without improving fidelity.

5 Related Work

Mixed-precision training keeps a high-precision master copy of the parameters while executing selected operations in a narrower format (Micikevicius et al., 2018). A parallel line reduces the format width itself, down to one-bit architectures (Wang et al., 2023) and 1.58-bit variants (Ma et al., 2024), and post-training quantization compresses a trained model further, by second-order updates, low-rank finetuning, or the activation and rotation rescaling of (Frantar et al., 2023; Dettmers et al., 2023; Xiao et al., 2023; Lin et al., 2024; Ashkboos et al., 2024). Later work assigns precision per layer, per operator, or per graph rewrite, using the cost of casts (He et al., 2022), convergence behavior (Dai et al., 2025), or a cyclic schedule (Fu et al., 2021) as the assignment signal. The closest case switches

precision between epochs on information-theoretic conditions and aims at the lowest precision each layer tolerates (Kummer et al., 2021); we measure that lowest precision directly rather than schedule it. These methods select or assign the available formats; our focus is the requirement itself: we measure MSP with controlled sweeps, track when a measured value becomes stale, and use it to select an executable level.

Numerical analysis derives error bounds and minimum precision requirements from the arithmetic structure of an operation, whether through analytical precision requirements for a network (Sakr & Shanbhag, 2018) or through the multiplier and accumulator widths that low-precision arithmetic can tolerate (Courbariaux et al., 2014). Precision-aware scaling laws extend this to model quality and cost as a function of the format width (Kumar et al., 2024). These results bound the error of an arithmetic operation. We add a learning-level criterion that can select a different level from local output accuracy, and we study how a criterion-specific requirement varies across operators and training states.

Ozaki schemes reconstruct a high-precision matrix multiplication from low-precision operations by splitting operands into components and accumulating their products (Ozaki et al., 2012), and fused tensor-core implementations make that reconstruction practical on GPUs (Mukunoki et al., 2020), on integer tensor cores (Ootomo et al., 2024), and through kernels built for precision emulation (Lu et al., 2026). Recent work goes further and argues that on AI-optimised accelerators an FP8 tensor-core operation composed through Ozaki II can carry FP64-grade accuracy for matrix-dominated kernels, demoting native FP64 from a hardware requirement to a derived accuracy guarantee (Matsuoka, 2026). We use this mechanism as an execution substrate, and our question is different from all of the above: prior work asks which available precision to use or how to realize a target precision, whereas we study the requirement itself, why it varies, and how long it remains valid.

6 Discussion and Conclusion

MSP is local, criterion-specific, and grid-resolved. It belongs to one operator execution, and the two criteria can select different levels for the same execution. It is also reported on the calibrated discrete grid the backend can execute, so a measured value describes the closest sufficient executable level, and a finer grid reduces over-provisioning.

The Oracle runs offline: a complete sweep produces the labels and a reference, and deployment reuses a stored value while it remains valid, refreshes it when it becomes stale, or predicts it from recorded features.

The study covers an A100 SM80 backend, four calibrated precision levels, four evaluation model families plus one controlled stress workload, and short writeback trajectories. Operators differ in precision behavior, and the emulation backend is one of several ways to realize a calibrated grid. A full sweep costs a multiple of a single execution, which sits in the offline budget. Appendix H records the failure modes we observed, among them unresolved MSP and non-monotonic fidelity curves.

Training systems choose precision from a fixed set of datatypes, but the precision a learning computation requires is local, criterion-specific, and dynamic. We defined Minimum Sufficient Precision, measured it across four evaluation model families and one controlled stress workload, related its variation to numerical factors and to the optimization response, and tracked how long a measured value remains valid before provisioning it on a calibrated discrete grid. The two criteria disagreed in 67 of the 120 valid comparisons, and the requirement moved with training state, so the value of refreshing tracked how fast it moved. Measure the precision the computation requires, then provision the closest sufficient level the backend can realize.

Ethics Statement

This work studies the numerical precision that learning computations require. It uses public model checkpoints and standard text and image inputs, involves no human participants and no personal data, and deploys no model. Measurements run on a single A100 GPU over

short horizons, so the compute used is small relative to training the models involved. Our claims concern numerical accuracy and arithmetic cost within the measured conditions; we make no claim about fairness, safety, or downstream social impact, and we release no new model weights.

Reproducibility Statement

The measurement protocol, the controlled-workload conditions, the tolerance settings, and the accounting of conditions are given in Appendix G; the explanatory relation and its evaluation are given in Appendix D. Every number reported in the experiments is computed from the archived per-condition records without further post-processing, and the kernel-level timings follow the protocol stated there. The anonymous supplementary archive contains the GEMM backend, the measurement harness, the policy implementations, the per-condition CSVs behind every main-text figure, the plotting scripts, and the environment and configuration files, and its layout is recorded in the archive index.

AI Use Statement

The authors used an AI assistant for two tasks: proposing presentation forms for the figures, and polishing the language of the manuscript. The figures themselves were drawn by the authors. All content, experiments, and analysis are the authors' own work, and every suggested change was reviewed and approved by the authors, who remain responsible for the claims, results, and citations in this paper.

References

Patrick Amestoy, Alfredo Buttari, Nicholas J. Higham, Jean-Yves L'Excellent, Théo Mary, and Bastien Vieublé. Five-precision GMRES-based iterative refinement. SIAM Journal on Matrix Analysis and Applications, 2024.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In Advances in Neural Information Processing Systems (NeurIPS), 2024.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications, 2014.

Wenhao Dai, Ziyi Jia, and Yuesi Bai. Convergence-aware operator-wise mixed-precision training. CCF Transactions on High Performance Computing, 7:43–57, 2025.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In Advances in Neural Information Processing Systems (NeurIPS), 2023.

Massimiliano Fasi, Nicholas J. Higham, Florent Lopez, Théo Mary, and Mantas Mikaitis. Matrix multiplication in multiword arithmetic: Error analysis and application to GPU tensor cores. SIAM Journal on Scientific Computing, 45(1), 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In International Conference on Learning Representations (ICLR), 2023.

Yonggan Fu, Han Guo, Xin Yang, Yining Ding, Yingyan Lin, Meng Li, and Vikas Chandra. CPT: Efficient deep neural network training via cyclic precision. In International Conference on Learning Representations (ICLR), 2021.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, and Jun Zhu. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. In Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2024.

Xin He, Jianhua Sun, Hao Chen, and Dong Li. Campo: Cost-aware performance optimization for mixed-precision neural network training. In 2022 USENIX Annual Technical Conference (USENIX ATC 22). USENIX Association, 2022.

Tanishq Kumar, Zachary Ankner, Benjamin F. Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision, 2024.

Lorenz Kummer, Kevin Sidak, Tabea Reichmann, and Wilfried Gansterer. Adaptive precision training (AdaPT): A dynamic quantized training approach for dnns, 2021.

Wonyeol Lee, Rahul Sharma, and Alex Aiken. Training with mixed-precision floating-point assignments. Transactions on Machine Learning Research, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In Proceedings of Machine Learning and Systems (MLSys), 2024.

Denghui Lu, Alexander Maeder, Mathieu Luisier, and Alexandros Nikolaos Ziogas. EmuGEMM: Fused tensor core kernels for precision emulation in matrix multiplication, 2026.

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

Satoshi Matsuoka. FP8 is all you need (part 1): Debunking hardware FP64 as the HPC holy grail, 2026.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Haonan Wu. Mixed precision training. In International Conference on Learning Representations (ICLR), 2018.

Daichi Mukunoki, Katsuhisa Ozaki, and Takeshi Ogita. DGEMM using tensor cores, and its accurate and reproducible versions. In High Performance Computing (ISC High Performance 2020), volume 12151 of Lecture Notes in Computer Science, pp. 153–171. Springer, 2020.

Hiroyuki Ootomo, Katsuhisa Ozaki, and Rio Yokota. DGEMM on integer matrix multiplication unit. The International Journal of High Performance Computing Applications, 2024.

Katsuhisa Ozaki, Takeshi Ogita, Shin'ichi Oishi, and Siegfried M. Rump. Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications. Numerical Algorithms, 59(1):95–118, 2012.

Houwen Peng et al. FP8-LM: Training FP8 large language models, 2023.

Bita Darvish Rouhani et al. Microscaling data formats for deep learning, 2023.

Charbel Sakr and Naresh R. Shanbhag. Minimum precision requirements for deep learning with biomedical datasets. In 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2018.

Mart van Baalen, Andrey Kuzmin, Suparna S. Nair, et al. FP8 versus INT8 for efficient deep learning inference, 2023.

Hongyu Wang et al. BitNet: Scaling 1-bit transformers for large language models, 2023.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. In Advances in Neural Information Processing Systems (NeurIPS), 2023.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In Proceedings of the 40th International Conference on Machine Learning (ICML), 2023.

Wayne Xin Zhao et al. A survey of large language models, 2023.

Xiaosu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on efficient training of transformers. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI), 2023.

A Measurement Protocol Details

A.1 Captured state

The replay record contains the target operator identity, layer, shape, reduction dimensions, operands or replayable handles, graph location, the selected downstream reference quantity, and the model, optimizer, random, and data-loader state required by the fidelity horizon. Records are keyed by model, operator, seed, and input window, so that one condition can be replayed without re-running the surrounding training step. The measurement harness, its checkpoint representation, and the storage layout of the archived records are released with the supplementary material.

A.2 Intervention

Only the arithmetic precision of the target operator changes; every other operand, parameter, and surrounding operation keeps its reference value:

$$y^{P}_{o,t} = F_o(x_{o,t}; P), \qquad y^{\mathrm{ref}}_{o,t} = F_o(x_{o,t}; P_{\mathrm{ref}}).$$

A.3 Reference and repetition

The reference configuration is accepted once increasing reference precision no longer changes the selected fidelity quantities beyond the tolerance recorded in the experiment archive. Repeated identical replays estimate measurement noise. The protocol fixes the norm, the normalization dimensions, the floor τ, the learning-level fidelity distance and its horizon, the repetition count, and the confidence interval in the archived configuration files.

A.4 Non-monotonicity

The Oracle evaluates all calibrated precision levels. Each record retains the full curve and a monotonicity flag. If no level satisfies the criterion, MSP is reported as unresolved above the available precision range.

A.5 Threshold sensitivity

Each condition is labelled under strict, default, and loose tolerance settings around the default numerical deviation 10^{-4}, gradient/update deviation 10^{-2}, and loss deviation 10^{-3}. The per-condition label changes are reported with the archived summaries.

B Demand and Supply Accounting

Writing P_{dtype} for the narrowest native datatype at least as wide as the demand, and P_{virtual} for the smallest calibrated level at least as wide as it, the over-provision of a datatype choice and of a calibrated level are

$$G_{\mathrm{dtype}} = P_{\mathrm{dtype}} - P^*, \qquad G_{\mathrm{virtual}} = P_{\mathrm{virtual}} - P^*,$$

and the gap that remains after provisioning is

$$G_{\mathrm{supply}} = P_{\mathrm{provision}} - P^* \ \geq 0.$$

With native datatypes the residual gap is set by the spacing of the datatype ladder; with the calibrated grid it is set by the spacing of the four levels, so a finer supply can reduce over-provisioning without changing the learning behavior the demand was measured against.

C Precision Emulation Implementation

A configuration specifies (n_A, n_B), the component extraction and scaling, the retained product set $\Omega(c)$, the physical GEMM datatype, the accumulation datatype, and the reconstruction order. Operands are sliced into significand components, the retained component products are executed with INT8 tensor-core GEMMs on the A100 (IMMA.16816.S8.S8), accumulated exactly in INT32, and reconstructed along the diagonal. The reconstructed result is compared against an FP64 reference, and the component counts and reconstruction order are the parameters that produce the calibrated levels.

The reported levels are calibrated arithmetic effective precision: a level of 13.7 means that the reconstructed product matches a software-rounded reference to the accuracy of that level over the probe suite. The value is an arithmetic property, distinct from any storage format such as INT13.7 or a quantization grid.

For calibration probe b,

$$e(c, b) = \frac{\|\widetilde{C}(c, b) - C^{\mathrm{ref}}(b)\|}{\max(\|C^{\mathrm{ref}}(b)\|, \tau)}.$$

The same probes are evaluated under candidate software-rounded precisions. Configuration c receives the largest calibrated effective precision for which the calibration error over the probe suite stays within the tolerance of that level; the probe set, the aggregation quantile, and the candidate precision grid are listed in the archived calibration configuration. Held-out probes verify the ordering of the levels and identify non-monotonic or shape-specific configurations.

The cost record separates decomposition, component GEMMs, reconstruction, temporary storage, and end-to-end latency. Configurations dominated in both calibrated precision and cost are removed.

D Demand-Model Details

Arithmetic features include shapes, the reduction dimension, operand norms, scale ratios, dynamic ranges, and the cancellation proxy

$$R(A, B) = \frac{\| \, |A| \, |B| \, \|}{\max(\|AB\|, \tau)},$$

where A and B are the operands of the product and $|A|$ is the elementwise absolute value. The proxy is large when the product of the operand magnitudes is much larger than the magnitude of the result. Learning-side features include the operational proxies S_{grad} and S_{update} defined in the main text, gradient and update magnitudes, normalization responses, and residual-path responses. Features are computed per operator execution condition and are used in the reported fits without additional normalization.

Explanation is evaluated by explained variance on the pooled conditions, by the Spearman correlation between individual features and P^*_{learn}, and by the ordinal accuracy obtained when a model fitted on the controlled stress workload is applied to MLP-Mixer-B/16. The last measure is 0.333, so the combined relation is reported as an explanatory relation rather than as a cross-model predictor.

E MSP Validity and Lifetime

Validity $V(o, t_0, t)$ as defined in Section 3 is one while the level measured at t_0 still satisfies the learning-level criterion at t. The reuse window of that level is the longest interval over

Table 1: Precision-selection policies. Oracle MSP selection is an offline reference; the remaining adaptive policies are deployable policy families.

Policy	Decision unit	Selection signal	Role
Fixed precision	All target executions	Native datatype	Hardware baseline
Static emulation	All target executions	One calibrated level	Static baseline
Per-operator	Operator/layer/phase group	MSP quantile	Deployable policy
Oracle MSP	Operator execution	Measured MSP	Offline reference
Predicted MSP	Operator execution	Predicted MSP + margin	Deployable policy

which validity holds,

$$L_{\mathrm{MSP}}(o, t_0) = \max\{t - t_0 : \ V(o, t_0, t') = 1 \text{ for every } t_0 \le t' \le t\},$$

and a stored value is stale at the first state where $V = 0$. We describe the observed trajectories as stable, staged, transition-aligned, dynamic, or right-censored according to the shape of this window; the corresponding records are listed in the archive index.

F Precision-Selection Policy Details

Tracking chooses between reusing a stored level and refreshing it,

$$r_t \in \{0, 1\}, \qquad P_t = \begin{cases} P_{t_{\mathrm{last}}}, & r_t = 0, \\ \mathrm{EstimateMSP}(o, x_t, \theta_t), & r_t = 1, \end{cases}$$

where $r_t = 0$ reuses the last measured level and $r_t = 1$ recomputes it. Per-operator selection groups executions by operator role and layer and chooses an upper MSP quantile plus a margin, while predicted-MSP selection chooses its margin on held-out records to meet a target under-provision rate. Both eventually resolve to a configuration through

$$c^*(P_{\mathrm{target}}) = \arg \min_{c: P_{\mathrm{eff}}(c) \ge P_{\mathrm{target}}} T(c),$$

where $T(c)$ is the measured cost of executing configuration c; this selects the cheapest configuration whose calibrated effective precision is at least the target. The deployment record includes policy lookup, feature collection, fallback frequency, and all precision-emulation overheads. This paper evaluates the fixed, static, and measured-MSP entries of Table 1; the measured-MSP entry provisions the measured value directly and appears in the experiments as MSP-guided provisioning, while the per-operator and predicted-MSP entries describe the remaining design points of the same framework.

G Experimental Protocol

G.1 Hardware

All measurements use an NVIDIA A100 PCIe 40 GB GPU with compute capability sm80. The backend executes INT8 tensor-core GEMMs through IMMA.16816.S8.S8, with a boundary tile at $M = 70, N = 73$ covered by the correctness tests.

G.2 Software

The measurement harness and the emulation kernels are built for CUDA 12.9 and PyTorch 2.6.0+cu126; the kernel structure is described in Appendix C.

G.3 Workloads

distilgpt2 and bert-tiny use their public checkpoints with real text windows; MLP-Mixer-B/16 uses a public ImageNet-1k checkpoint with fixed image batches; the controlled ill-conditioned MLP varies the input regime over $[-1, 1]$, $[-3, 3]$, $[-6, 6]$, and $[-7, 7]$, of which

Table 2: Condition accounting. The five workloads above the rule enter the pooled statistics of Section 4, where 67 of the 120 valid comparisons show a numerical/learning-level mismatch. The row below is reported separately.

Workload	Conditions	In pooled statistics
distilgpt2	$4 \times 2 \times 3$	yes
bert-tiny	$6 \times 2 \times 3$	yes
MLP-Mixer-B/16	24	yes
Poisson DeepONet	24	yes
Controlled ill-conditioned MLP	12	yes
Pooled valid comparisons	120	—
High-cancellation condition	1	no; numerical only

the first, second, and fourth enter the main figures as the six primary conditions of seeds 4321 and 4322 on fc1 and fc2; Poisson DeepONet starts from its fitted state. The operator, seed, and window identifiers of every condition are listed in the archive index, and Table 2 gives the condition accounting.

G.4 Mismatch by workload

The pooled count is spread across all five workloads. Of the 120 comparisons, the numerical and learning-level criteria disagree on 24/36 conditions for bert-tiny, 16/24 for distilgpt2, 11/24 for MLP-Mixer-B/16, 7/24 for Poisson DeepONet, and 9/12 for the controlled ill-conditioned MLP. Conditions within one workload share a model, operator, and input family, so these per-workload rates are a more conservative summary than the pooled total, which counts correlated conditions together.

G.5 Metrics and statistics

Labels are produced with the numerical deviation 10^{-4}, gradient/update deviation 10^{-2}, and loss deviation 10^{-3} thresholds, with the strict and loose sensitivity settings described in Appendix A. Runtime is measured with CUDA events per trace shape, with eight warmups and 31 alternating repeats, and reported as an accumulated managed-GEMM median that covers the GEMM kernels only and excludes packing, scale preparation, workspace setup, and launch overhead.

H Additional Results

H.1 Per-workload demand

In the Transformer workloads, distilgpt2 has a numerical MSP of $p = 2$–3 and a learning-level MSP of $p = 2$, while bert-tiny spans $p = 2$–3 numerically and $p = 1$–2 at the learning level. MLP-Mixer-B/16 has a numerical MSP of $p = 2$ or $p = 3$ against a learning-level MSP of $p = 2$, and Poisson DeepONet spans $p = 1$–3 numerically against $p = 1$–2 at the learning level.

H.2 Operator- and state-resolved demand

Figure 6 resolves the pooled view of Figure 3 into operators and training states, grouped by model family; columns are ordinal condition indices within one family rather than shared samples across families, and unsampled cells are left blank.

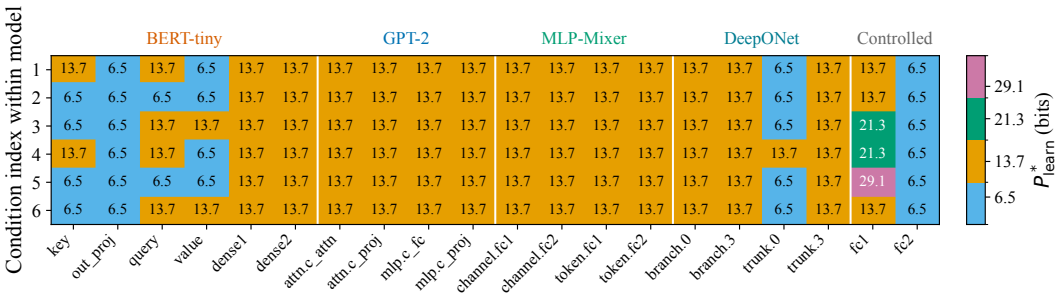

Figure 6: Measured learning-level MSP over operators and training states, grouped by model family. The map is the operator-resolved counterpart of Figure 3.

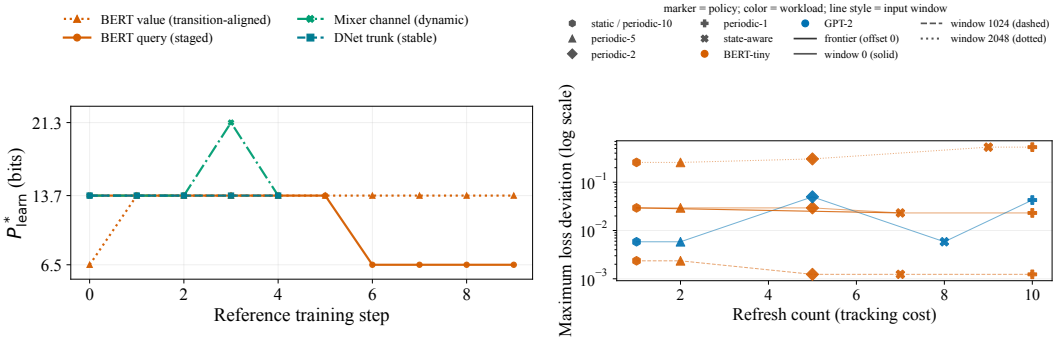

Figure 7: MSP dynamics and the refresh–fidelity trade-off. The left panel overlays four directly labelled trajectories of the stored learning-level MSP against the reference training step; the right panel reports the maximum loss deviation against the tracking cost, with marker for the policy and colour for the model, line style for the input window, and an offset-0 frontier line. Static reuse and a ten-step periodic refresh coincide over the ten-step horizon and share a legend entry.

H.3 Refresh policies

Figure 7 gives the two views behind the tracking results of Section 4.6: the stored learning-level MSP along four representative trajectories, and the refresh cost against the resulting maximum loss deviation.

H.4 Full provisioning numbers

Every deviation in this paragraph is measured against the FP64 reference. The three-step protocol uses a mean level of $p = 2$ on distilgpt2 with a loss deviation of approximately 8.32×10^{-5}, and a mean level of $p = 1.5$ on bert-tiny with a loss deviation of approximately 1.84×10^{-2}. Over ten steps, the distilgpt2 deviations across three windows are 2.39×10^{-4}, 1.59×10^{-3}, and 7.61×10^{-3}, while bert-tiny gives 1.116, 0.0570, and 0.512. On the same three bert-tiny windows the FP32 baseline itself deviates from FP64 by 2.49×10^{-5}, 6.90×10^{-6}, and 4.51×10^{-5}, two to three orders of magnitude below the loss threshold $\epsilon = 10^{-3}$; the long-horizon deviations above are therefore attributable to the precision of the guided setting rather than to trajectory noise.

H.5 Failure modes

The archive retains the calibration, mismatch, refresh, timing, and trajectory records behind the main-text figures, including four failure modes: MSP that is unresolved above the widest available level, stored levels that go stale as the state moves, non-monotonic fidelity curves that the Oracle evaluates in full, and policies whose emulation overhead offsets the precision they save.

The archive contains the per-condition demand maps, the per-workload MSP distributions, the calibration curves $P_{\text{eff}}(c)$, the mismatch lists, the strict/default/loose threshold summaries, the refresh-policy rows with the update and parameter deviations against the FP32 reference, the kernel-only timing summary, and the non-monotonic and stale-measurement cases. The main-text numbers of Section 4 are computed from these records without further post-processing.