Q-PACE: DYNAMIC PRECISION ALLOCATION FOR QUANTIZATION-AWARE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization-aware training (QAT) leverages lower-precision arithmetic to reduce the cost of LLM deployment, but aggressive quantization degrades final performance. In order to assign precision to different layers, common approaches rely on quantization error proxies. We provide a new approach, called Q-PACE, consisting of a second-order sensitivity model that predicts the loss increase as a sum of quantization noise MSE weighted by per-layer curvature coefficients. During training, we periodically re-compute these coefficients using perturbations across layers, and re-assign precision. Pretraining and supervised fine-tuning experiments on LLMs of up to 4B parameters show that Q-PACE consistently improves over existing recipes, and achieves comparable loss at substantially lower total memory budgets. We further find that quantization sensitivity is highly predictable by depth and layer type, and its stability during training allows for infrequent, cheap recalibration.

1 INTRODUCTION

Quantization-aware training (QAT) methods (Courbariaux et al., 2016; Jacob et al., 2018) use low-precision arithmetic to lower the cost of both *deploying* LLMs, as they result in lower-bitwidth accurate models, and *training* them, as the matrix multiplications during training can be mapped to lower precision. Recent recipes make end-to-end NVFP4 and MXFP4 pretraining viable by leveraging techniques such as block (micro) scaling, stochastic rounding, rotations, and unbiased gradient estimators (Castro et al., 2026; Chen et al., 2026; NVIDIA et al., 2026). However, there is still a clear loss gap between lower-precision approaches and BF16 and MXFP8 training, and uniform four-bit training can be lossy or even become unstable over long horizons (NVIDIA et al., 2026).

It is known that not every network layer needs the same precision, e.g. Malinovskii et al. (2025). Moreover, modern microscaling formats allow matrix products across various precisions, or even mixed-precision operands (Rouhani et al., 2023), so the bitwidth of each linear layer can be chosen separately, assuming a global budget constraint. The key question, which we consider here, is how to choose the layer-wise bitwidths that maximize accuracy while satisfying a precision budget. Current approaches either keep a fixed set of layers, typically the last few, in higher precision (NVIDIA et al., 2026), or rank layers by a local proxy, such as the product of gradient norm and quantization error (Pan et al., 2026). It is not clear whether these heuristics can be improved.

In this paper, we propose a new approach. Specifically, we show that the second-order model of the loss under independent, zero-mean quantization noise accurately predicts the expected loss increase by assigning "sensitivity" scores to each layer (Section 3). This is a new instance of the "linearity theorem" model (Malinovskii et al., 2025), previously used for one-shot, post-training quantization. We call our approach **Q-PACE**, **P**recision **A**llocation via **C**urvature **E**stimation for **Q**uantization-aware training, and apply it at training iterates. Estimated per-layer sensitivity coefficients depend only on the model weights and are stable across iterations, so Q-PACE re-estimates them periodically during training: once they are determined, we solve a knapsack problem to determine "optimal" per-layer bit-widths. This approach only requires forward passes for coefficient estimation, and has negligible memory overhead.

We evaluate Q-PACE on pretraining experiments for models between 350M and 4B parameters, and on supervised fine-tuning of Qwen3 models, against state-of-the-art baselines (NVIDIA et al., 2026; Pan et al., 2026), at budgets between 5.0 and 6.5 bits per weight. Our findings are as follows:

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.

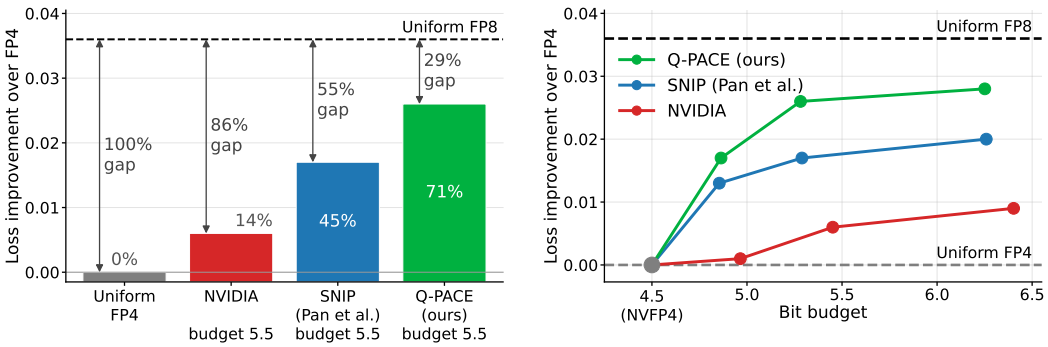

Figure 1: Final loss improvement for a 500M model trained with mixed precision across methods and budgets. We improve over *last layers in BF16* (NVIDIA et al., 2026) and *SNIP* (Pan et al., 2026).

- Q-PACE consistently improves over existing mixed-precision training strategies across model scales, precision budgets, and format options. In pretraining, it achieves the lowest final loss in every setting we evaluate, recovering up to 76% of the loss gap between uniform NVFP4 and MXFP8 training; the same trend holds for supervised fine-tuning.
- Q-PACE provides a substantially better quality–precision trade-off. At a fixed precision budget, Q-PACE produces better final loss than both SNIP and the NVIDIA heuristic, while at matched loss it requires fewer bits per model weight (Figure 1).
- A detailed analysis of the produced allocations shows that quantization sensitivity is highly structured across the network: it depends on both layer type and depth, and its profile evolves over the course of training. However, these changes occur much slower than parameter updates. As a result, only a small number of reallocations (on the order of 10 over a full training run) is sufficient, keeping the overhead of Q-PACE low in practice.

All in all, our results show that layer-wise precision can be assigned from a sensitivity model derived directly from the loss and the forward pass, rather than from more complex error proxies, and that doing so can recover a large fraction of the accuracy gap to uniform higher-precision training, at a small fraction of the training cost.

2 The Problem: QAT with Adaptive Mixed-Precision

Our goal is to choose the quantization levels of n linear weight tensors in a network for the purpose of quantization-aware training. We concentrate on linear layers, i.e. fully-connected layers or attention projections. For each of the full-precision weight tensors $W = (W_i)_{i=1}^n$, we want to select a corresponding *bitwidth*, b_i, from a set of candidate bitwidths, denoted as β.

We use the term *bitwidth* as it is common in the literature. Technically, the selection is among *formats* used for matrix multiplication operations. We distinguish between formats if they use a different value representation, even if they nominally have the same number of bits, such as FP8 E8M0 and FP8 E4M3. We denote the quantization of a tensor W_i by $Q(W_i, b_i)$. Any choice of quantization will influence the model's overall quality, i.e. the achieved *loss*, as well as its resource *cost*, such as storage and compute. Typically, there is a tradeoff between those: a larger bitwidth results in lower loss, but higher cost.

For a sequence $B = (b_1, \ldots, b_n) \in \beta^n$, called *bitwidth allocation*, we denote its loss by $L(W, B) := L(\{Q(W_i, b_i) \text{ for } b_i \in B\})$, and its resource cost by $C(W, B)$. We assume the cost to be additively linear, $C(W, B) := \sum_{i=1}^n c_i(b_i)$, where $c_i(b_i) = d(W_i)\kappa(b_i)$, d is a function of the size of the weight tensor W_i (but not of its actual values), and κ is a characteristic cost of a bitwidth.

With the goal to constraint memory at inference time, we take $d(W_i) =: d_i$ as the number of elements in a tensor, and $\kappa(b_i)$ as the number of bits required to store an element, which we write as b_i again, meaning that $C(W, B) = \sum_{i=1}^n d_i b_i$ measures the total model size in bits. Note, however, that the framework we describe here would also apply to other choices of $d(W_i)$ and $\kappa(b_i)$.

Ultimately, for a model with set of weights W, the problem of *optimal bitwidth allocation* is minimization of the loss under a given maximum model size constraint, C_0. Formally, we want to

find

$$B^* = \underset{B=(b_1,\dots,b_n)}{\operatorname{argmin}} L(W, B) \text{ subject to } \sum_i b_i d_i \leq C_0. \tag{1}$$

For post-training quantization, the bitwidth allocation only has to be chosen once. In this work, however, we focus on QAT, where the choice of quantization has a direct influence on the quality of learned weights, and one cannot assume that the best quantization strategy is constant over the course of training (Liu et al., 2025b).

Consequently, we are interested in finding a *sequence of allocations* $B^1, \dots, B^T$, where T is the number of training steps, that ultimately achieves the lowest final loss while adhering to the cost budget at each iteration. The new precision allocation at each optimization step then affects both the loss calculation in the forward pass and the errors in the gradient computation in the backward pass.

The final set of weights, W^T, which determines the ultimate loss, is affected by all previous allocations. Then the optimal sequence of allocations during training should be

$$(B^1, \dots, B^T)^* \coloneqq \underset{\text{all trajectories } (B^t)_t}{\operatorname{argmin}} L(W^T(B^1, \dots, B^{T-1}), B^T) \text{ subject to } \max_{t=1,\dots,T} C(B_t) \leq C_0. \tag{2}$$

Due to the complex structure with interactions between layers as well as time steps, a direct optimization of Eq. (2) is intractable. In the following section, we discuss approximations to $L(W, B)$ that make the optimization tractable.

3 METHOD

For simplicity of exposition, we first discuss a setup where only two bitwidth choices are available: high-precision, `high`, and low-precision, `low`, e.g. BF16 and FP4, respectively.

For a bitwidth allocation $B \in \{\texttt{low}, \texttt{high}\}^n$, we denote the quantization error (perturbation) of quantizing a layer into `low` format as $\delta W_i := Q(W_i, b_i) - W_i$. Assuming that `high` is the original data format, such that $Q(W_i, \texttt{high}) = W_i$, we have $\delta W_i = 0$ if $b_i = \texttt{high}$, but $\delta W_i \neq 0$ otherwise. With $\delta W := (\delta W_1, \dots, \delta W_n)$, a Taylor expansion of the loss function around the unquantized loss $L(W) = L_{\texttt{orig}}$ yields

$$L(Q(W, B)) - L(W) \approx \sum_i \Big\langle \frac{\partial L(W)}{\partial W_i}, \delta W_i \Big\rangle + \frac{1}{2} \sum_{i,j} \delta W_i^\top \frac{\partial^2 L(W)}{\partial^2 W_i W_j} \delta W_j. \tag{3}$$

Following the *Linearity Theorem* (Malinovskii et al., 2025), the quantization error δW_i can be modeled as an isotropic random variable with zero mean $\mathbb{E}[\delta W_i] = 0$ and finite variance $\mathbb{V}[\delta W_i] = \sigma_i^2 I$, $\sigma_i^2 > 0$. Moreover, the errors are expected to be uncorrelated between layers, $\mathbb{E}[\delta W_i \delta W_j] = 0$ for $1 \leq i < j \leq n$. Assuming this, we can take the expectation over δW_i in Eq. (3) to obtain:

$$\begin{aligned} \mathbb{E}\Big[L(W + \delta W) - L(W)\Big] &\approx \mathbb{E}\Big[\sum_i \langle \nabla_{W_i}, \delta W_i \rangle + \frac{1}{2} \sum_{i,j} \delta W_i^\top H_{ij} \delta W_j\Big] = \\ &= \frac{1}{2} \sum_i \operatorname{Tr}(H_{ii})\, \sigma_i^2 = \frac{1}{2} \sum_i \operatorname{Tr}(H_{ii}) \underbrace{\frac{1}{d_i} \mathbb{E}||\delta W_i||_F^2}_{=\mathrm{MSE}(Q(W_i), W_i)}, \end{aligned} \tag{4}$$

where d_i denotes the number of elements in the i-th weight matrix, and H_{ii} denotes the Hessian matrix of L with respect to these weights.

The derivation presented above can be done for any precision `low`; thus, we obtain, for any bitwidth allocation:

$$L(W, B) \approx L_{\texttt{orig}} + \sum_i \alpha_i \widehat{\mathrm{MSE}}(Q(W_i, B_i), W_i), \tag{5}$$

where $L_{\texttt{orig}} = L(W)$ is the loss over full-precision weights, $\widehat{\mathrm{MSE}}$ is an estimate of the MSE, and α_i is an estimate of $\frac{1}{2} \operatorname{Tr}(H_{ii})$ that can be interpreted as the aggregated curvature of a layer.

Algorithm 1 Q-PACE

Require: Weight tensors $W = (W_1, \ldots, W_n)$, calibration data $\mathcal{D}$, update interval τ

repeat
  Evaluate $L_{\texttt{orig}}$ on $\mathcal{D}$
  for layers $i = 1, \ldots, n$ **do**
    for m noise levels $\{\sigma_k^2\}$ **do**
      Add noise $\delta W_i \sim N(0, \sigma_k^2)$ to layer weight W_i
      Determine $\widehat{\mathrm{MSE}}(W_i, W_i + \delta W_i)$
      Evaluate $L(W + \delta W_i)$ on $\mathcal{D}$
    end for
    Estimate sensitivity α_i via least squares using Eq. (6)
  end for
  $B \leftarrow$ solution of knapsack problem Eq. (1)
  Train model with the new bitwidth allocation B for τ steps
until termination condition

Connection to Hutchinson trace estimation. For an isotropic random probe z satisfying $\mathbb{E}[zz^\top] = I$, we have $\mathbb{E}_z[L(W + \epsilon z) - L(W)] \approx \mathbb{E}_z\left[z^\top H_{ii} z\right] = \frac{\epsilon^2}{2} \operatorname{Tr}(H_{ii})$ (Hutchinson, 1990). Our forward-pass calibration can be viewed as a variant of Hutchinson trace estimation: the layer-wise curvature α_i is estimated from the increase in loss induced by isotropic random perturbations, normalized by their mean-squared magnitude.

MSE estimation. The estimate $\widehat{\mathrm{MSE}}$ can be obtained empirically by measuring the statistics of the actual tensor weights or the analytic approximation, $\widehat{\mathrm{MSE}}(Q(W_i), W_i) \propto 4^{-b_i}$, based the analysis in App. A. The experiments (App. A) showed a moderate advantage of using the analytical MSE estimates over measured ones, so we adopt the former throughout this study.

Curvature estimation. To estimate α_i, we use that $Q(W_i, B_i) = W_i$ and $\mathrm{MSE}(Q(W_i, B_i), W_i) = 0$ for the layers that are kept in full precision, so effectively Eq. (5) sums only over those layers that are quantized. Then, for layer i we model quantization error by adding Gaussian noise δW_i to weight tensor and can directly read off

$$L(W, B_i) - L_{\texttt{orig}} = \alpha_i \, \widehat{\mathrm{MSE}}(Q(W_i, B_i), W_i) = \alpha_i \, \widehat{\mathrm{MSE}}(W_i + \delta W_i, W_i). \tag{6}$$

The dataset on which the loss in Eq. 6 is evaluated, $\mathcal{D}$, we call the *calibration dataset*. It should be large enough for an accurate loss approximation while keeping it small enough to avoid excessive computational overhead.

In principle, Eq. (6) yields $\alpha_1, \ldots, \alpha_n$ from $n + 1$ forward passes over $\mathcal{D}$, but we note that using several noise perturbations per layer with different variances improves our loss estimation. We used $m \in \{3, \ldots, 10\}$ noise levels to estimate α_i. Therefore, curvature estimation requires the total of $mn + 1$ forward passes. See details on hyperparameter choice in App. E.

Bitwidth assignment. Once we have determined sensitivities α_i for all $i = 1, \ldots, n$ and available bitwidths, we can solve Eq. (1) with Eq. (5) as a proxy for the intractable $L(W, B)$. As a discrete selection problem with linear objective and linear constraints, this constitutes a *multiple choice knapsack problem (MCKP)*. For our implementation we use pseudo-polynomial dynamic programming (Pisinger, 1995): the budget C_0 is discretized into M bins, and each layer cost $d_i b_i$ is rounded up to the whole number of bins. The discrete problem is then solved via a dynamic program that tracks an allocation that gives minimal loss degradation for each cost level. The time complexity of the algorithm is $O(nM|\beta|)$, and in practice, for the models used, we measure $\lesssim 1$ s of knapsack solving time, which is negligible compared to multiple forward passes needed for estimating α_i.

Q-PACE. Ideally, the estimates α_i and $\widehat{\mathrm{MSE}}$ should stay up-to-date at any time during training. However, given that they require multiple forward passes to be updated, it would be unreasonable to change them at every optimization step. In practice, we observed that sensitivities demonstrate a stable pattern during training and infrequent re-allocations suffice. Therefore, we propose to update the estimates only at regular intervals of τ steps. In our experiments, re-allocation periods lower than 10k training batches did not yield measurable improvement. Summarizing the discussion above, we

obtain the Q-PACE procedure which is our main contribution in this work, see Algorithm 1 for the pseudo-code.

Computational and memory overhead. Q-PACE calibration requires only forward passes. To perturb layer i we add δW_i to the weights and evaluate the loss on $\mathcal{D}$. The only extra memory is a single noise tensor the size of the largest layer, which is negligible compared to the training state. In contrast, gradient-based methods (Pan et al., 2026) require storing the activations and gradients needed for the backward pass, resulting in a substantially larger memory overhead.

One recalibration costs $mn + 1$ forward passes over $\mathcal{D}$: one unperturbed pass for L_{bf16} and one per (layer, noise level) pair. It is followed by a knapsack solve with negligible cost. For a batch size B the relative compute overhead is then $\frac{(mn+1)|\mathcal{D}|}{3\tau B}$, linear in both m and $|\mathcal{D}|$ and inversely proportional to τ. In our experiments the measured wall-clock time overhead was only 5% for 4B model corresponding to a conservative choice of $\tau = 5k$. Time overhead can likely be reduced substantially by increasing τ (Sec. 5). More details on hyperparameter choice can be found in App. E.

4 Experimental Results

4.1 Experimental Setup

We train a decoder-only Transformer, with each block following an OLMo-2-style (OLMo: et al., 2025) residual layout: multi-head self-attention and a two-layer MLP, with RMSNorm applied to each sub-layer output before the residual. Attention uses rotary position embeddings, and Qwen3-style QK-RMSNorm. We use model sizes of 350M, 500M, 800M, and 4B. We start our experiments with a 350M model on 18B tokens and evaluate all the strategies, then select the best-performing ones for larger up to 4B model runs.

The training data is ClimbMix dataset (Diao et al., 2025). We use a subset pre-tokenized with TokenMonster (vocab size 32 000). We used AdamW (Kingma & Ba, 2017; Loshchilov & Hutter, 2019) and Muon (Jordan et al., 2024) optimizers. The learning rate (LR) schedule used is WSD (Hu et al., 2024). The optimal LR is found via a grid search. Context length range is from 1024 for 350M model to 4096 for 4B model, more details are provided in App. C.

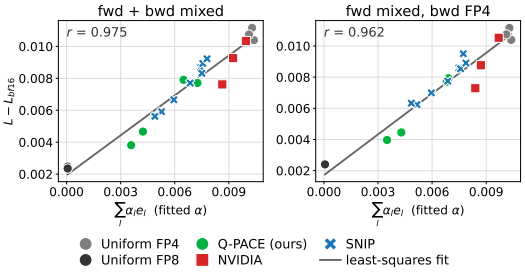

Figure 2: Test loss vs. the modeled sensitivity-weighted forward error for 500M models' final allocations.

Quantization-aware training. Linear layers in the blocks (Q, K, V, O, and MLP projections) are the only quantized modules; embeddings and the LM head remain in high precision. We follow the e2e QAT setup by (Panferov et al., 2026), which performs all of the weights-activations matmul and gradients-weight/activations matmuls in FP4. In our experiments, the candidate low precision formats for matmix multiplications are NVFP4, MXFP6, MXFP8. Unless stated otherwise, all matmuls for a layer, forward and backward, are performed in the same precision. The weight master copy and optimizer state are stored in full precision.

Baselines. We compare Q-PACE against uniform NVFP4, MXFP8, the *last-layers* strategy of NVIDIA et al. (2026), and SNIP (Pan et al., 2026). NVIDIA keeps a suffix of quantization-sensitive layers at `high` precision, with the cutoff chosen to satisfy the budget. SNIP solves a similar knapsack problem, but estimates layer costs using gradient-based forward sensitivity and backward-update divergence. We also consider a version of SNIP, named SNIP weight-gradient-only, as it performs better than the original SNIP recipe. See App. D for details.

4.2 Forward Error Is Sufficient for Loss Estimation

Before we report our method's performance, we provide a justification for our choice of modeling the quantization error purely as a function of the MSE, i.e. weight quantization error, Eq. (4). Other mixed-precision training methods model loss degradation using the error introduced during the backward pass (Pan et al., 2026). We test whether the signal from the backward pass is necessary for predicting the loss under QAT.

Table 1: Final test loss across models. Budget is the target average weight bitwidth. "Rec." (recovery) is the share of the FP4-to-FP8 gap recovered, $(L_{\text{FP4}} - L)/(L_{\text{FP4}} - L_{\text{FP8}})$. The column "gain" reports the ratio between the gain of Q-PACE over FP4 and the gain of the NVIDIA allocation over FP4, $(L_{\text{FP4}} - L_{\text{Q-PACE}})/(L_{\text{FP4}} - L_{\text{NVIDIA}})$. The std error measured over 3 random seeds for 350M model is ≤ 0.001. Best loss per row is shown in bold.

Model	Budget	FP4	FP8	NVIDIA	Rec.	SNIP	Rec.	Q-PACE	Rec.	gain
350M, 16B tok	5.0 $\{4, 6, 8, 16\}$	2.886	2.848	2.884	6%	2.873	34%	**2.870**	**42%**	7.5×
	5.5 $\{4, 6, 8, 16\}$			2.880	17%	2.870	43%	**2.861**	**68%**	3.9×
	6.5 $\{4, 8, 16\}$			2.877	26%	2.865	56%	**2.860**	**70%**	2.7×
	6.5 $\{4, 16\}$			2.877	26%	2.874	32%	**2.870**	**43%**	1.7×
500M, 74B tok	5.0 $\{4, 6, 8, 16\}$	2.693	2.657	2.692	2%	2.680	36%	**2.676**	**45%**	18.8×
	5.5 $\{4, 6, 8, 16\}$			2.687	14%	2.676	45%	**2.667**	**71%**	4.9×
	6.5 $\{4, 8, 16\}$			2.684	24%	2.673	54%	**2.665**	**76%**	3.1×
	6.5 $\{4, 16\}$			2.684	24%	2.681	31%	**2.676**	**47%**	1.9×
800M, 80B tok	5.0 $\{4, 16\}$	2.616	2.581	2.613	9%	2.613	9%	**2.609**	**20%**	2.2×
	5.5 $\{4, 16\}$			2.611	14%	2.612	11%	**2.607**	**26%**	1.9×
	6.0 $\{4, 16\}$			2.607	26%	2.605	31%	**2.602**	**40%**	1.5×
4B, 105B tok	6.5 $\{4, 16\}$	2.278	2.251	2.271	28%	2.270	31%	**2.267**	**44%**	1.6×

For this, we compare two training settings for a 500M model trained on 74B tokens, with various precision allocation methods. In the first setting, the layer-wise precision assignment is applied to both the forward and backward matrix multiplications. In the second, the precision assignment is applied only to the forward matmuls, while all the backward matmuls are computed in FP4 (FP4 bwd). The second setup introduces additional backward pass quantization error that is not captured by our forward error metric (Eq. 5).

Figure 2 shows the final test loss as a function of our loss estimate (Eq. 5), which uses forward error only. In both settings, the final loss remains linear in the forward error. We calculate the Pearson correlation between the loss estimate and final test loss, r. When forward and backward precisions are the same per layer, the correlations are $r = 0.98$ for our model. Keeping the backward pass uniformly in FP4 does not change this, and the corresponding Pearson correlation remains $r = 0.96$.

We conclude that, although additional perturbation is indeed introduced during the backward pass in QAT, backward pass metrics are *not required* to predict the impact on the loss. Across the precision allocation methods tested, forward error alone can be a good predictor of final loss. Supporting results in App. B.

4.3 Pretraining Experiments

We compare the mixed-precision training methods by their final test loss in Table 1. Across all model scales, precision budgets, and candidate precision sets, Q-PACE achieves lower final test loss than both the NVIDIA heuristic and SNIP (Pan et al., 2026). These gains are obtained under the same average precision budget. Q-PACE recovers substantially more of the FP4-to-FP8 loss gap than other methods, up to 76% of the gap. Q-PACE provides $1.5\times$–$19\times$ the loss improvement of the NVIDIA across the reported settings. The largest ratios occur at tight budgets, where the NVIDIA allocation improves only marginally over uniform FP4, and the advantage remains present as the model scale increases.

This shows that allocating precision according to curvature-aware layer sensitivity is consistently more effective compared to other baselines.

The benefit of sensitivity-aware allocation is particularly clear when more than 2 precision levels are available. At the same 6.5-bit budget, replacing the bitwidth set $\{4, 16\}$ with $\{4, 8, 16\}$ increases the recovered gap of Q-PACE from 43% to 70% for the 350M model and from 47% to 76% for the 500M model. Figure 4 further shows how our method improves with additional *bitwidth choices*. With the same budget of 6.5 bits/parameter, using $\{4, 8, 16\}$ as available formats allows to close 75% of gap to the FP8 level, whereas with only $\{4, 16\}$ choices it reaches 44%.

4.4 Supervised Fine-Tuning Experiments

We extend our experiments to supervised fine-tuning (SFT). We fine-tune Qwen3-4B-Base (Team, 2025) on Tulu-3 (Lambert et al., 2025) with budgets from 5.0 to 6.5 over three random seeds. Table 2 reports the held-out loss, the recovered share of the FP4-to-FP8 gap, and the mean accuracy over the

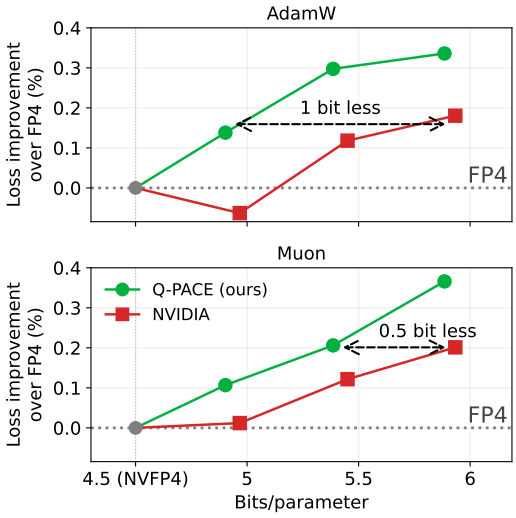

Figure 3: Loss improvement over uniform FP4 for 800M models with varying budgets. To achieve the same quality models trained with Q-PACE need 1 bit/parameter less than those trained with NVIDIA when using AdamW and 0.5 bit/parameter less when using Muon.

	Loss	Rec.	Acc.
uniform FP4	$0.7375_{\pm 0.0005}$	0%	$55.0_{\pm 0.3}$
uniform FP8	$0.6790_{\pm 0.0001}$	100%	$57.3_{\pm 0.2}$
budget 5 $\{4, 6, 8, 16\}$			
NVIDIA	$0.7289_{\pm 0.0006}$	15%	$55.0_{\pm 0.2}$
SNIP	$0.7135_{\pm 0.0005}$	41%	$55.9_{\pm 0.1}$
Q-PACE	$\mathbf{0.7098}_{\pm 0.0007}$	**47%**	$\mathbf{55.8}_{\pm 0.3}$
budget 5.5 $\{4, 6, 8, 16\}$			
NVIDIA	$0.7233_{\pm 0.0007}$	24%	$55.4_{\pm 0.2}$
SNIP	$0.7066_{\pm 0.0018}$	53%	$56.3_{\pm 0.3}$
Q-PACE	$\mathbf{0.7021}_{\pm 0.0004}$	**60%**	$\mathbf{56.4}_{\pm 0.2}$
budget 6.5 $\{4, 8, 16\}$			
NVIDIA	$0.7176_{\pm 0.0001}$	34%	$54.9_{\pm 0.2}$
SNIP	$0.6999_{\pm 0.0005}$	64%	$56.7_{\pm 0.1}$
Q-PACE	$\mathbf{0.6940}_{\pm 0.0003}$	**74%**	$\mathbf{56.3}_{\pm 0.4}$
budget 6.5 $\{4, 16\}$			
NVIDIA	$0.7176_{\pm 0.0001}$	34%	$54.9_{\pm 0.2}$
SNIP	$0.7112_{\pm 0.0003}$	45%	$56.0_{\pm 0.1}$
Q-PACE	$\mathbf{0.7094}_{\pm 0.0002}$	**48%**	$\mathbf{55.9}_{\pm 0.4}$

Table 2: SFT held-out loss, recovered share of the FP4-to-FP8 gap (Rec.), and mean downstream accuracy of Qwen3-4B-Base on Tulu-3 (mean and standard deviation over three seeds).

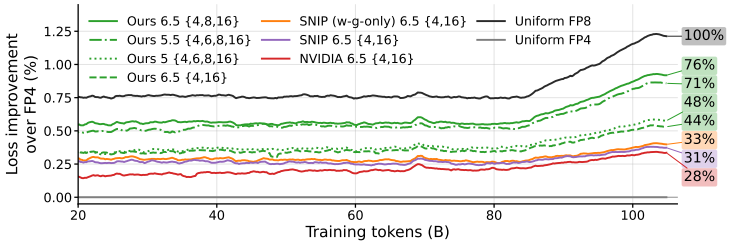

Figure 4: Pretraining experiment for a 4B model. Test-loss improvement over uniform FP4; labels give the recovered gap between FP4 and FP8 trainings, realized bits/parameter, and the precision options in parentheses.

downstream benchmarks. Q-PACE reaches the lowest loss at every budget, with a substantial margin over the baselines. Q-PACE outperforms the NVIDIA baseline in the downstream tasks average. However, it falls very close to SNIP and within the noise range. Detailed downstream benchmarks are provided in App. H.1.

4.5 Performance-memory Trade-off

Next, we study the trade-off between model quality and precision budget. Figure 3 shows test loss improvement over the FP4 baseline 800M model across bitwidths. For both AdamW and Muon, Q-PACE forms an empirical Pareto frontier: at a fixed precision budget, it achieves a larger loss improvement than the NVIDIA, while at a matched loss it requires fewer bits per weight. For Adam, our method outperforms NVIDIA strategy while using 1 bit per parameter less, that is 16% less model size in memory. With Muon, the corresponding saving is approximately 0.5 bits per weight. Altogether, this indicates that the benefit of curvature-aware sensitivity is not restricted to a single precision budget or optimizer.

Since a model can be deployed under various weight and activation formats during inference, we additionally measure the loss–memory trade-off for different *inference formats*. Figure 5a shows the evaluation of 800M model checkpoints under several inference configurations, including $W_{\text{mix}}A_{\text{mix}}$, $W_{\text{mix}}A_8$, and $W4A8$. Across the evaluated configurations, models trained with Q-PACE again lie on the empirical Pareto frontier: for a given inference-memory budget they achieve lower loss, or equivalently reach a comparable loss with less memory. Thus, the advantage of sensitivity-guided precision allocation during training is retained under variable inference formats.

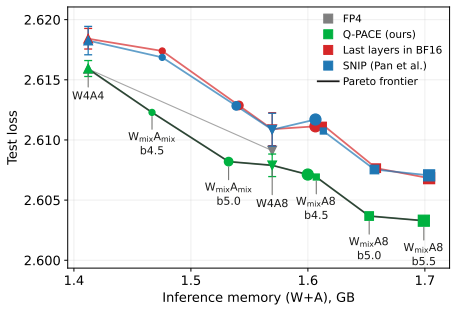

Figure 5a: Loss – inference memory tradeoff for 800M model across different inference formats: $W_{mix}A_{mix}$, $W_{mix}A_8$, $W4A8$.

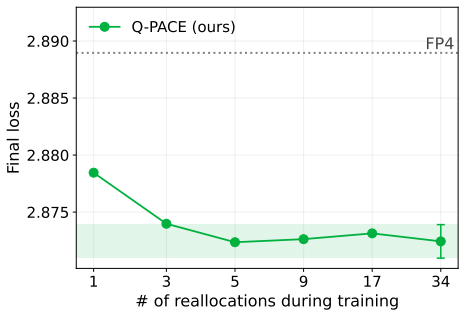

Figure 5b: Final loss as a function of number of reallocations during training for the 350M model.

5 Analysis

Sensitivity is structured by layer type and depth. Figure 6a shows the final precision allocation produced by Q-PACE and SNIP for the 4B model. The allocation is highly non-uniform: the assigned precision depends both on the layer type and its position in the model. The depth dependence is particularly pronounced in the final block, where Q-PACE promotes five of the six quantized linear layers to the full precision. This observation helps to explain why the NVIDIA heuristic is effective relative to uniform FP4 training: later layers are indeed highly sensitive to quantization.

However, sensitivity is not solely a function of depth. Layers of different types within the one block receive different precisions, while sensitive layers can also occur substantially earlier in the network. In particular, K and V projection layers receive a substantial fraction of the higher-precision budget under both Q-PACE and SNIP. At the same time, SNIP does not promote MLP projections at all, while Q-PACE assigns high precision to MLP projections toward the end of the model.

Sensitivity evolves during training. Figure 6b shows the fraction of each layer type assigned to BF16 throughout training. While the number of layers of each type, selected by SNIP, remains nearly constant over time, the allocation produced by Q-PACE evolves. The largest changes occur between the early and middle stages of training, after which the allocation becomes substantially more stable. K and V projections remain repeatedly promoted to BF16, while the share of promoted Q projections decreases over time, and the share of MLP grows.

Therefore, the sensitivity pattern has both a constant factor, determined by layer type and block number, as well as a time-dependent factor. The former explains why simple static heuristics can provide some benefit, whereas the latter motivates periodic reallocation during training. Additional analysis results can be found in App. G

How frequently is reallocation necessary? Next, we analyze the impact of the reallocation period τ. Figure 5b reports final loss while varying the number of reallocations over training. For a 350M model, reallocating more than five times over the full training run provides no additional benefits.

This result demonstrates that the gain of Q-PACE is not only due to finding a better *initial* mixed-precision allocation. Adapting the allocation dynamically during training provides an additional benefit. Secondly, it shows that sensitivity changes on a much slower timescale than optimization steps, allowing to keep the computational overhead of Q-PACE recalibration marginal.

6 Related Work

Post-training quantization (PTQ) (Dettmers et al., 2022; Xiao et al., 2023; Frantar et al., 2023) has become a standard approach for reducing the memory and inference cost of LLMs. Most PTQ methods apply the same bitwidth to every layer.

Non-uniform compression. A parallel line of work distributes compression non-uniformly across layers. AMC (He et al., 2018) and HAQ (Wang et al., 2019) used reinforcement learning with hardware feedback for this, while DNAS (Wu et al., 2018) relaxed the search into a differentiable one. HAWQ (Dong et al., 2019; 2020; Yao et al., 2021) instead scores layers by second-order information, and selects bitwidths by integer programming. SPDY (Frantar & Alistarh, 2022) and

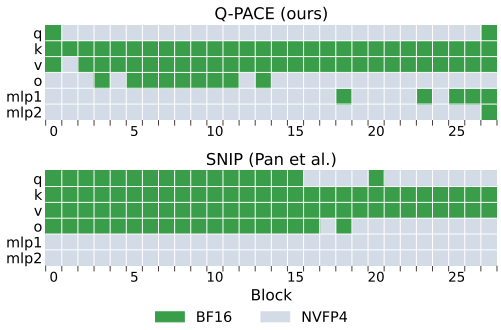

Figure 6a: Final format allocation for 4B model at 6.5 bit/param budget for Q-PACE and SNIP (Pan et al., 2026). Both method favor K and V projection layers, with Q-PACE also promoting the MLP in the last blocks.

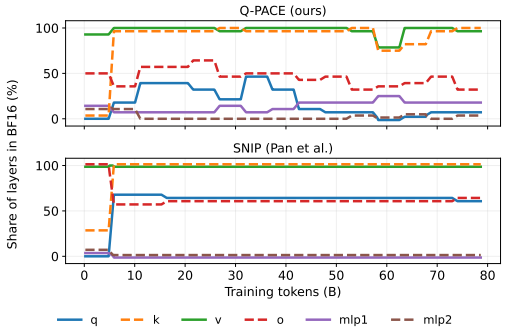

Figure 6b: Share of layers promoted to BF16 in 4B model at 6.5 bit/param, by layer type, over training duration.

OBC (Frantar et al., 2022) cast the allocation as a dynamic program over per-layer compression levels. EvoPress (Sieberling et al., 2025) showed that such additive ("error-monotone") models can fail, and replaced them with an evolutionary search. For LLMs, mixed-precision schemes range from keeping salient weights or groups in higher precision (Dettmers et al., 2024; Kim et al., 2024; Huang et al., 2025), to layer-wise allocation driven by per-layer sensitivity estimates and knapsack solvers (Malinovskii et al., 2025; Helcig et al., 2026), or even more complex manifold optimization (Helcig & Alistarh, 2026). Other work accounts for interactions between layers (Schaefer et al., 2023) or for perturbations (Hu et al., 2025). The closest work to ours is HIGGS (Malinovskii et al., 2025), whose linearity theorem underlies our sensitivity model. Q-PACE differs from it in three main ways: 1. HIGGS allocates bitwidths once; we allocate *during* training; 2. HIGGS compresses only the weights, whereas we compress weights, activations, and gradient multiplications. 3., We investigate hardware-supported microscaling formats (NVFP4, MXFP6, MXFP8, BF16) and use an analytic 4^{-b} distortion model, so no per-layer, per-format errors are needed.

Mixed precision during training. Non-uniform precision *during* training has received less attention. Low-precision training recipes typically keep a hand-picked set of components in higher precision, e.g. DeepSeek-V3 keeps the embeddings, output head, MoE gating, normalization, and attention in higher precision (DeepSeek-AI, 2024), and NVFP4 pretraining keeps the last blocks in BF16 (NVIDIA et al., 2026). Earlier work on convolutional networks learned per-layer bitwidths jointly with the weights during QAT (Wu et al., 2018; Uhlich et al., 2020; van Baalen et al., 2020). However, these methods target the precision of the final inference model rather than the cost of training itself. The work closest to ours is SNIP (Pan et al., 2026), which periodically re-assigns per-layer precision during LLM pretraining by solving a knapsack problem. Its scores combine a first-order Taylor estimate of the forward loss change with an optimizer-dependent estimate of the divergence in the weight updates. Q-PACE replaces these with a second-order, loss-based sensitivity model, estimated from forward passes alone. We show that this yields substantially better accuracy.

7 Conclusion

We propose a new method for adaptive mixed-precision LLM training, which probes linear layers and estimates their sensitivity. This sensitivity information is then used for periodic reassignment of precision across layers. With no memory overhead, our method consistently improves over existing baselines and allows for substantial memory savings at a matched performance. Across pretraining experiments from 350M to 4B parameters and supervised fine-tuning of Qwen3-4B, Q-PACE consistently improves over existing mixed-precision strategies at the same precision budget, recovering up to 76% of the NVFP4-to-MXFP8 loss gap at 6.5 bit/parameter.

Our study is limited to a restricted range of model scales and precision formats, and primarily considers settings where the same layer-wise precision is used in the forward and backward passes. Extending dynamic allocation to larger models, broader format choices, and independently optimized forward and backward precisions are promising directions for future work.

AI USE STATEMENT

We used ChatGPT and Claude to assist in organizing and editing the manuscript, verifying mathematical reasoning and our interpretation of the results, retrieving bibliography entries, and preparing the LaTeX files for submission. Additionally, we employed Claude Code to visualize our experimental results. We have thoroughly reviewed and verified the final manuscript, including all citations, experimental details, and AI-assisted edits, and we take full responsibility for its content.

REPRODUCIBILITY STATEMENT

We provide detailed information intended to support reproduction of our results. The model architectures, training configurations, datasets, optimizer settings, learning-rate schedules, batch sizes, context lengths, and recalibration frequencies are described in Appendix C. The numerical formats and implementation details for NVFP4, MXFP6, MXFP8, and BF16 training, including the hardware used, are specified in Appendix C. The Q-PACE procedure is given explicitly in Algorithm 1, including the sensitivity-estimation and precision-allocation steps, and we report the calibration and reallocation hyperparameters used in our experiments. Appendix E studies the sensitivity of the method to its main calibration hyperparameters, while the remaining appendices provide additional ablations, analyses, and downstream results. All reported experimental results are obtained from the configurations described in the paper.

REFERENCES

W. R. Bennett. Spectra of quantized signals. *The Bell System Technical Journal*, 27(3):446–472, 1948.

Roberto L. Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native FP4 training can be optimal for large language models, 2026. URL https://arxiv.org/abs/2505.14669.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Yuxiang Chen, Yifan Liu, Xiaoming Xu, Pengle Zhang, Michael Beyer, Martin Rapp, Jun Zhu, and Jianfei Chen. TetraJet-v2: Accurate NVFP4 training for large language models with oscillation suppression and outlier control, 2026. URL https://arxiv.org/abs/2510.27527.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations, 2016. URL https://arxiv.org/abs/1511.00363.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-CLIMB: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/24a8968affe71ffe4067d022b9d16566-Paper-Datasets_and_Benchmarks_Track.pdf.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare Quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d77c703536718b95308130ff2e5cf9ee-Paper.pdf.

Elias Frantar and Dan Alistarh. SPDY: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning*, 2022.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, 2022.

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*, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Herbert Gish and John Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, 1968.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018.

Michael Helcig and Dan Alistarh. Model compression with exact budget constraints via Riemannian manifolds, 2026. URL https://arxiv.org/abs/2605.00649.

Michael Helcig, Eldar Kurtic, and Dan Alistarh. Statistically-lossless quantization of large language models, 2026. URL https://arxiv.org/abs/2605.02404.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Yuezhou Hu, Weiyu Huang, Zichen Liang, Chang Chen, Jintao Zhang, Jun Zhu, and Jianfei Chen. Identifying sensitive weights via post-quantization integral, 2025. URL https://arxiv.org/abs/2503.01901.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. In *International Conference on Machine Learning*, 2025.

M.F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 18:1059–1076, 01 1990. doi: 10.1080/03610918908812806.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *International Conference on Machine Learning*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training, 2025a. URL https://arxiv.org/abs/2502.16982.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. Paretoq: Improving scaling laws in extremely low-bit llm quantization, 2025b. URL https://arxiv.org/abs/2502.02631.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. HIGGS: Pushing the limits of large language model quantization via the linearity theorem. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, April 2025. URL https://aclanthology.org/2025.naacl-long.543/.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Muya Chang, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, Bita Darvish Rouhani, Carlo del Mundo, Deena Donia, Burc Eryilmaz, Henry Estela, Abhinav Goel, Oleg Goncharov, Yugi Guvvala, Robert Hesse, Russell Hewett, Herbert Hum, Ujval Kapasi, Brucek Khailany, Mikail Khona, Nick Knight, Alex Kondratenko, Ronny Krashinsky, Ben Lanir, Simon Layton, Michael Lightstone, Daniel Lo, Paulius Micikevicius, Asit Mishra, Tim Moon, Deepak Narayanan, Chao Ni, Abhijit Paithankar, Satish Pasumarthi, Ankit Patel, Mostofa Patwary, Ashwin Poojary, Gargi Prasad, Sweta Priyadarshi, Yigong Qin, Xiaowei Ren, Oleg Rybakov, Charbel Sakr, Sanjeev Satheesh, Stas Sergienko, Pasha Shamis, Kirthi Shankar, Nishant Sharma, Mohammad Shoeybi, Michael Siu, Misha Smelyanskiy, Darko Stosic, Dusan Stosic, Bor-Yiing Su, Frank Sun, Nima Tajbakhsh, Shelby Thomas, Przemek Tredak, Evgeny Tsykunov, Gandhi Vaithilingam, Aditya Vavre, Rangharajan Venkatesan, Roger Waleffe, Qiyu Wan, Hexin Wang, Mengdi Wang, Lizzie Wei, Hao Wu, Evan Wu, Keith Wyss, Ning Xu, Jinze Xue, Charlene Yang, Yujia Zhai, Ruoxi Zhang, Jingyang Zhu, and Zhongbo Zhu. Pretraining large language models with NVFP4, 2026. URL https://arxiv.org/abs/2509.25149.

Team OLMo:, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

Team Olmo:, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Yunjie Pan, Yongyi Yang, Hanmei Yang, and Scott Mahlke. SNIP: An adaptive mixed precision framework for subbyte large language model training, 2026. URL https://arxiv.org/abs/2602.01410.

Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet II: Accurate LLM pre-training in NVFP4 by improved unbiased gradient estimation, 2026. URL https://arxiv.org/abs/2601.22813.

David Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, 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 Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning, 2023. URL https://arxiv.org/abs/2310.10537.

Clemens JS Schaefer, Navid Lambert-Shirzad, Xiaofan Zhang, Chiachen Chou, Tom Jablin, Jian Li, Elfie Guo, Caitlin Stanton, Siddharth Joshi, and Yu Emma Wang. Augmenting hessians with

inter-layer dependencies for mixed-precision post-training quantization, 2023. URL https://arxiv.org/abs/2306.04879.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. EvoPress: Accurate dynamic model compression via evolutionary search. In *International Conference on Machine Learning*, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision DNNs: All you need is a good parametrization. In *International Conference on Learning Representations*, 2020.

Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. In *Advances in Neural Information Processing Systems*, 2020.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning. https://github.com/huggingface/trl, 2020. URL https://github.com/huggingface/trl.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of ConvNets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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 *International Conference on Machine Learning*, 2021.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A MSE ESTIMATION: ANALYTIC VS MEASURED

The estimate $\widehat{\mathrm{MSE}}$ can be obtained empirically by measuring the statistics of the actual weight tensors' quantization error. In practice, however, we rely on the efficient analytic approximation, $\widehat{\mathrm{MSE}}(Q(W_i), W_i) \propto 4^{-b_i}$, based on the following analysis.

For any bitwidth b (except BF16, where no quantization is required), assume quantization with a signed uniform b-bit grid. Then, the quantization error of a value $x \in [-\xi, \xi)$ is

$$\delta x = q(x) - x \qquad \text{for} \qquad q(x) = s \ \mathrm{clip}\big(\mathrm{round}(x/s), -2^{b-1}, 2^{b-1} - 1\big), \tag{7}$$

with scale parameter $s = 2^{-(b-1)}\xi$, i.e. there are 2^b quantization intervals of size s. Assuming that values are distributed approximately uniformly within each quantization interval, the rounding error has variance $s^2/12$ (Bennett, 1948). Consequently, a natural estimate for the MSE is

$$\mathbb{E}\|\delta X\|^2 \propto 4^{-b}, \tag{8}$$

recovering the expressions of *high-resolution quantization* (Gish & Pierce, 1968). Note that the exact constants of proportionality are not important in our application, since they will be absorbed automatically in the estimate of α_i.

Employing a single error curve for NVFP4, MXFP6, MXFP8, and BF16 format constitutes a simplifying assumption, as these formats differ in element grid, block size, and scale representation. However, preliminary experiments using the measured relative weight error for each tier did not improve the allocation over Eq. (8), so we adopt the modeled curve throughout this study.

Model	Budget	Error	Final test loss	Improvement over FP4
350M AdamW	6.33	Analytic	$2.8598_{\pm 0.0014}$	$0.527_{\pm 0.049}\,\%$
		Measured	$2.8598_{\pm 0.0017}$	$0.527_{\pm 0.060}\,\%$
800M AdamW	5.39	Analytic	$\mathbf{2.6081}$	$\mathbf{0.298}\,\%$
		Measured	2.6095	$0.242\,\%$

Table 3: Q-PACE with the analytic vs the measured quantization error for MSE estimation in Eq. (5) model.

B ON LOSS LINEARITY VS. FORWARD-PASS ERROR

Table 4 extends the analysis presented in Figure 2 to all budget cells. Restricting the mixed tiers to the forward pass has minimal impact on the recovered share for any method, and Q-PACE achieves the highest performance in every cell. Figure 7 evaluates each model's final allocation using the sensitivity-weighted weight error $\sum_l \alpha_l e_l$ rather than the activation NMSE. The sensitivities α_l are refitted on the model's final weights to reflect the loss increase under weight noise. Across both experimental settings and model sizes, the loss increase resulting from quantization, $L - L_{\mathrm{bf16}}$, exhibits a linear relationship with this error. This holds whether e_l is modeled as 4^{-b_l} or measured directly on the quantized weights ($r \geq 0.96$). When using the modeled error, the fitted slope approaches one. In contrast, with the measured error, the slope is approximately one half.

C TRAINING SETUP

Pretraining. The pretraining process employs a 32,000-token TokenMonster vocabulary, untied embeddings (tied in the 800M model), and Rotary Position Embedding (RoPE) with $\theta = 10^4$. Following RoPE, queries and keys are normalized, and a learned temperature parameter is applied to each attention head. Each attention and multilayer perceptron (MLP) branch undergoes RMS normalization prior to residual addition, and a final RMSNorm is applied before the output head. Table 5 provides a summary of each model training configuration. The 350M, 500M, and 4B models are trained with Muon on the weight matrices of the transformer blocks and AdamW on all other parameters. The Muon optimizer utilizes a momentum of 0.95 and scales each orthogonalized $m \times n$ update by $0.2\sqrt{\max(m, n)}$. The AdamW optimizer employs $\beta = (0.9, 0.95)$ and $\epsilon = 10^{-6}$ (Jordan et al., 2024; Liu et al., 2025a). The 800M model is trained with AdamW alone, with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and a weight decay of 0.1 on every parameter, including the embeddings.

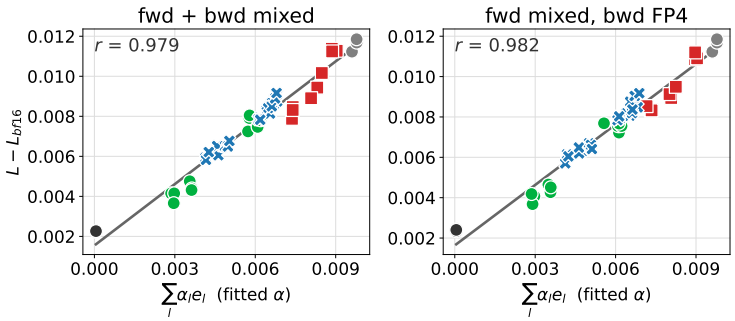

Figure 7a: 350M, analytical error $e_l = 4^{-b_l}$.

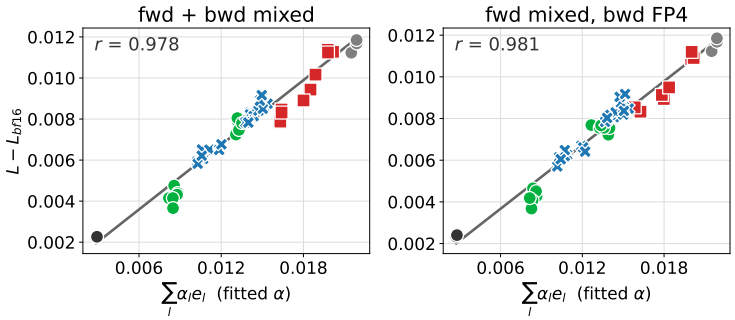

Figure 7b: 350M, measured error $e_l(b_l)$.

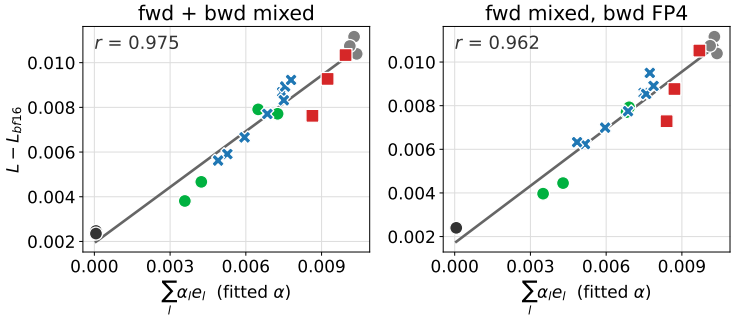

Figure 7c: 500M, analytical error $e_l = 4^{-b_l}$.

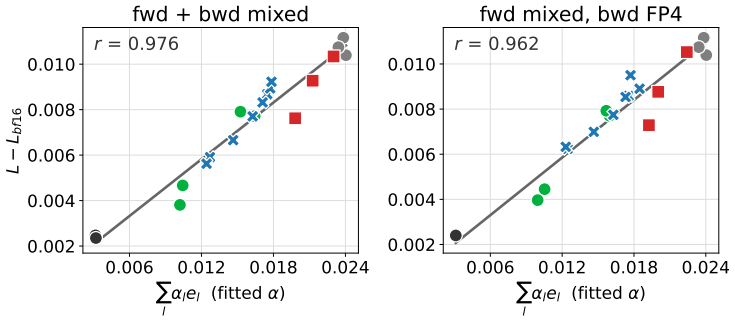

Figure 7d: 500M, measured error $e_l(b_l)$.

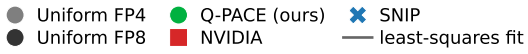

Figure 7: Loss increase from quantization, $L - L_{\mathrm{bf16}}$ (test loss minus that of the same weights in BF16), against the sensitivity-weighted weight error $\sum_l \alpha_l e_l$ of each model's final allocation, with α_l refitted on its final weights. Left: fwd + bwd mixed; right: fwd mixed, bwd FP4.

Table 4: Pretraining loss and recovered share of the NVFP4-to-MXFP8 gap (rec.) per budget cell when the tiers set both passes and when they set the forward only (FP4 backward everywhere; each setting against its own MXFP8 run).

		fwd mixed + bwd mixed		fwd mixed, bwd fp4	
Budget	Method	loss	rec.	loss	rec.
350M: uniform NVFP4 2.8864; uniform MXFP8 2.8480 (fwd mixed + bwd mixed), 2.8545 (fwd mixed, bwd fp4)					
5 $\{4, 6, 8, 16\}$	Q-PACE	**2.8703**	**42**%	**2.8725**	**44**%
	SNIP	2.8733	34%	2.8746	37%
	SNIP (w-g-only)	2.8734	34%	2.8744	38%
	NVIDIA	2.8843	6%	2.8844	6%
5.5 $\{4, 6, 8, 16\}$	Q-PACE	**2.8605**	**68**%	**2.8643**	**69**%
	SNIP	2.8698	43%	2.8721	45%
	SNIP (w-g-only)	2.8641	58%	2.8669	61%
	NVIDIA	2.8798	17%	2.8822	13%
6.5 $\{4, 8, 16\}$	Q-PACE	**2.8595**	**70**%	**2.8641**	**70**%
	SNIP	2.8647	56%	2.8675	59%
	SNIP (w-g-only)	2.8627	62%	2.8658	65%
	NVIDIA	2.8765	26%	2.8788	24%
6.5 $\{4, 16\}$	Q-PACE	**2.8697**	**43**%	**2.8720**	**45**%
	SNIP	2.8741	32%	2.8759	33%
	SNIP (w-g-only)	2.8719	38%	2.8735	41%
	NVIDIA	2.8765	26%	2.8788	24%
500M: uniform NVFP4 2.6926; uniform MXFP8 2.6565 (fwd mixed + bwd mixed), 2.6634 (fwd mixed, bwd fp4)					
5 $\{4, 6, 8, 16\}$	Q-PACE	**2.6764**	**45**%	**2.6793**	**46**%
	SNIP	2.6797	36%	2.6806	41%
	SNIP (w-g-only)	2.6802	34%	2.6807	41%
	NVIDIA	2.6917	2%	2.6915	4%
5.5 $\{4, 6, 8, 16\}$	Q-PACE	**2.6670**	**71**%	**2.6708**	**75**%
	SNIP	2.6763	45%	2.6781	50%
	SNIP (w-g-only)	2.6696	64%	2.6743	63%
	NVIDIA	2.6874	14%	2.6895	11%
6.5 $\{4, 8, 16\}$	Q-PACE	**2.6650**	**76**%	**2.6697**	**78**%
	SNIP	2.6733	54%	2.6755	59%
	SNIP (w-g-only)	2.6686	66%	2.6742	63%
	NVIDIA	2.6838	24%	2.6858	23%
6.5 $\{4, 16\}$	Q-PACE	**2.6756**	**47**%	**2.6803**	**42**%
	SNIP	2.6813	31%	2.6822	36%
	SNIP (w-g-only)	2.6787	39%	2.6806	41%
	NVIDIA	2.6838	24%	2.6858	23%

Table 5: Pretraining configurations.

	350M	500M	800M	4B
blocks / heads / GQA ratio (KV heads)	11 / 12 / 4 (3)	17 / 12 / 3 (4)	17 / 16 / 4 (4)	28 / 28 / 4 (7)
hidden / head dim	1536 / 128	1536 / 128	2048 / 128	3584 / 128
parameters (total / non-embedding)	370.8M / 272.5M	526.2M / 427.9M	814.3M / 748.8M	4.006B / 3.777B
switchable linears	66	102	102	168
context	1024	1024	1024	4096
global batch (seqs / tokens)	512 / 524k	512 / 524k	1024 / 1.05M	256 / 1.05M
steps / tokens	30k / 15.7B	140k / 73.4B	76.3k / 80.0B	100k / 104.9B
optimizer	Muon + AdamW	Muon + AdamW	AdamW	Muon + AdamW
master weights	fp32	fp32	fp32	fp32
peak LR / weight decay	6×10^{-3} / 0.01	6×10^{-3} / 0.01	7.5×10^{-4} / 0.1	2×10^{-3} / 0.02
embedding weight decay	0	0	0.1	0
schedule	WSD, cooldown 25k→30k	WSD 5%/20%	WSD 5%/20%	warmup 2000, cooldown 20000
recalibration period	4,000	10,000	2,000	10,000
seeds	3	1 (mixed) / 3 (uniform)	1	1

Supervised-Fine-tuning (SFT). Qwen3-4B-Base (Yang et al., 2025), which includes 252 switchable projections, is fine-tuned on 10% of the Tülu-3 dataset (Lambert et al., 2025) using TRL (von Werra et al., 2020). The Fused AdamW optimizer is configured with a learning rate of 5×10^{-6}, no weight decay, a linear-decay learning-rate scheduler with 3% warmup, and gradient clipping set to 1.0. Each epoch consists of 706 steps, with a global batch size of 128, a maximum sequence length of 4096, and training conducted in BF16 precision. The chat template includes an empty thinking block, and the loss function is computed using only assistant tokens. SFT recalibrates every 128 steps, resulting in five allocations over 706 steps. Reallocation is disabled during the final pretraining step, ensuring that each run concludes under an allocation it has previously trained on. Each calibration step samples a new batch from the reserved calibration split.

Numerics and hardware. MXFP8 employs E4M3 values with 32-element E8M0 scales, round-to-nearest (RTN) rounding, and a ceiling scale rule. MXFP6 utilizes E2M3 with RTN for the forward pass and stochastic rounding (SR) for the backward pass. NVFP4 adopts the Quartet II path, which incorporates randomized Hadamard transforms and its unbiased backward estimator (Panferov et al., 2026). In this implementation, MXFP6 values are represented exactly in E4M3 and processed through the MXFP8 GEMM path. Consequently, the 6-bit tier evaluates 6-bit numerics without assessing a native 6-bit execution path. All experiments are conducted on NVIDIA B200 GPUs, which natively support low-bit data formats.

D SNIP WEIGHT-GRAD-ONLY

SNIP (Pan et al., 2026) provides a first-order alternative to our method. Let G_l be the full-precision gradient of layer l on the calibration batch and let $\delta W_l^{(b)} = q_b(W_l) - W_l$ be its measured quantization error at tier b. We use

$$q_{l,b} = \frac{\|G_l\|_F \|\delta W_l^{(b)}\|_F}{\sqrt{d_l}}, \tag{9}$$

which bounds the magnitude of the first-order loss change up to the stated normalization. The same budgeted solver minimizes the selected scores. The original SNIP framework also includes activation- and optimizer-propagated backward terms. We observed that using only Eq. 9 outperforms the full pipeline. Therefore, we added it to some of the experiments for comparison and call it SNIP weight-grad-only.

E Q-PACE HYPERPARAMETER ABLATION

Here we evaluate how the performance of Q-PACE depends on the two main hyperparameters: the number of noise levels m used to estimate each layer sensitivity, and the size of the calibration dataset $|\mathcal{D}|$.

Figure 8 reports the final test-loss improvement for 350M model over uniform FP4 training for different combinations of m and $|\mathcal{D}|$. Using only a single perturbation and a single calibration sequence leads to the least accurate allocation. Increasing either the number of noise levels or the amount of calibration data improves the resulting allocation, but the gains quickly saturate.

This indicates that Q-PACE does not require a highly accurate curvature estimate to identify useful relative differences in layer sensitivity. A modest number of perturbations ($m \geq 4$) and calibration examples $|\mathcal{D}| \geq 64$ is already sufficient to obtain effective precision allocations. This robustness to calibration hyperparameters allows us to keep the computational overhead small during training.

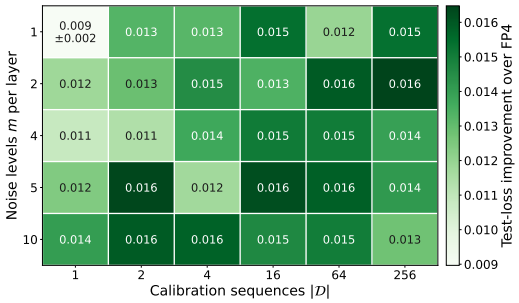

Figure 8: Test loss gain over uniform FP4 for 350M model over the number of noise levels m per layer and calibration sequences $|\mathcal{D}|$. Performance improves over the smallest calibration setting (1,1) and then remains relatively stable across a broad range of m and $|\mathcal{D}|$, indicating diminishing returns from additional calibration compute.

F ALTERNATIVE SENSITIVITY MODELS

Q-PACE is primary sensitivity model. We additionally compare against several alternative approximations.

Q-PACE inversed. Alternatively to Q-PACE model, one can consider a symmetric linear approximation relative to the loss of uniformly FP4 quantized model:

$$L(W, B) = L_{fp4} - \sum_i \alpha_i \frac{1}{d_i} ||\delta W_i||_F^2, \tag{10}$$

here $\Delta W_i = 0$ for quantized weights, and the sum effectively covers the set of **non-quantized** weights. $L_{fp4} = L(W, (b_i = fp4))$ is the loss evaluated on fully quantized model weights.

Table 6: Test loss results for 350M model trained on 18B tokens with budget 6.5 bit/parameter and choices 4,16. Average and std error over 3 random seeds.

Method	Final validation loss	Improvement over FP4
Q-PACE inversed	$2.8551_{\pm 0.0010}$	$0.554_{\pm 0.038}\,\%$
Q-PACE	$2.8554_{\pm 0.0011}$	$0.544_{\pm 0.037}\,\%$
Q-PACE: measured MSE(W)	$2.8569_{\pm 0.0021}$	$0.491_{\pm 0.061}\,\%$
Q-PACE: MSE(W_{grad})	$2.8570_{\pm 0.0005}$	$0.488_{\pm 0.031}\,\%$
Quadratic Q-PACE	$2.8575_{\pm 0.0018}$	$0.472_{\pm 0.050}\,\%$
Q-PACE: weights + activations MSE	$2.8581_{\pm 0.0010}$	$0.450_{\pm 0.048}\,\%$
Cosine sim.	$2.8593_{\pm 0.0039}$	$0.410_{\pm 0.138}\,\%$
First-order Taylor	$2.8628_{\pm 0.0015}$	$0.280_{\pm 0.044}\,\%$
NVIDIA	$2.8641_{\pm 0.0012}$	$0.240_{\pm 0.046}\,\%$
FP4	$2.8710_{\pm 0.0004}$	0.00

Generally, we want to assign each linear layer a scalar value to measure the "sensitivity" of a particular layer, that is, how much the loss would change if the layer were quantized while keeping the remaining layers fixed.

$$L(W, B) = \sum_i f(\alpha_i) + L_{bf16}. \tag{11}$$

First-order Taylor sensitivity. By keeping only the linear term, one can obtain the first-order approximation, which relates to the quality metric used by (Pan et al., 2026).

$$\alpha_i = \langle \nabla_{W_i}, \Delta W_i \rangle, \quad L(W, B) = \sum_i \alpha_i + L_{bf16}. \tag{12}$$

Cosine similarity.

$$\alpha_i = 1 - \cos(W_i, Q(W_i)) = 1 - \frac{\langle W_i, Q(W_i) \rangle}{||W_i||_F ||Q(W_i)||_F}. \tag{13}$$

Q-PACE-Quadratic model. Q-PACE formulation considers only independent effects of each layer. It can be extended to include pair-wise interactions between layers:

$$L(W, B) = \sum_i f(\alpha_i) + \sum_{i \neq j} g(\gamma_{ij}) + L_{bf16}. \tag{14}$$

$$L(W, B) = \sum_k \alpha_k \frac{||\Delta W_k||_F^2}{||W_k||_F^2} + \sum_{k \neq j} \gamma_{kj} \frac{||\Delta W_k||_F ||\Delta W_j||_F}{||W_k||_F ||W_j||_F} + L_{bf16}. \tag{15}$$

where $\{k\}$ is the set of quantized weights.

In Table 6, we report the test loss results for 350M-parameter model runs using the alternative strategies described above. All evaluated strategies improve over the FP4 baseline, although the magnitude of the improvement varies. The best results are obtained by the inversed Q-PACE variant and the default Q-PACE, which achieve equal improvements within error interval. Variants based on measured weight error, gradient MSE, and the quadratic Q-PACE perform similarly, yielding improvements between 0.472% and 0.491%. In contrast, cosine similarity, first-order Taylor, and NVIDIA provide smaller gains over FP4.

Overall, these results show that the performance of Q-PACE is robust across several alternative formulations, while the default Q-PACE formulation remains the strongest variant.

G Additional sensitivity analysis

We report additional sensitivity results in this section for the 800M model. These results complement the analysis in the main text by showing how the sensitivity profile varies across layer types, model depth, and training time.

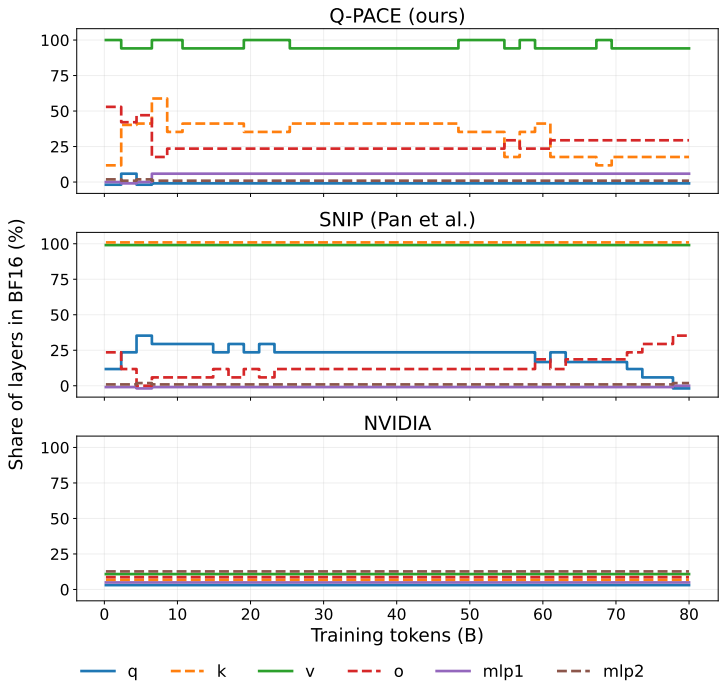

Figure 9: Share of different layer types promoted to BF16 over the training duration.

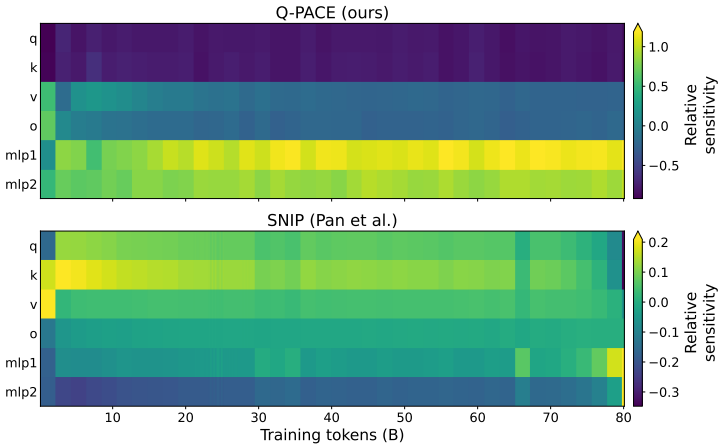

Figure 10: Relative sensitivity by layer type over training for the 800M model. Values are averaged across layers of the same type at each calibration point. Q-PACE assigns the largest relative sensitivity to the MLP projections, whereas SNIP places comparatively greater sensitivity on K projections.

Figure 9 shows the fraction of each layer type assigned to BF16 throughout training. The allocations produced by Q-PACE are highly non-uniform across layer types and evolve over the course of training. In particular, the V projections are consistently assigned high precision, while the fractions of K and O projections with high precision vary. In contrast, SNIP keeps nearly all K and V projections in BF16 for most of training, while the NVIDIA baseline induces a static allocation across layer types.

This illustrates that Q-PACE adapts its precision allocation over training time.

To better understand these allocations, Figure 10 shows the corresponding relative sensitivity scores aggregated by layer type. For Q-PACE, the MLP projections exhibit the largest relative sensitivities, whereas attention projections, in particular Q and K, are substantially less sensitive on average. SNIP produces a different ordering, assigning higher relative sensitivity to attention projections and lower sensitivity to the MLP projections. The sensitivity profiles are comparatively stable after the early stages of training.

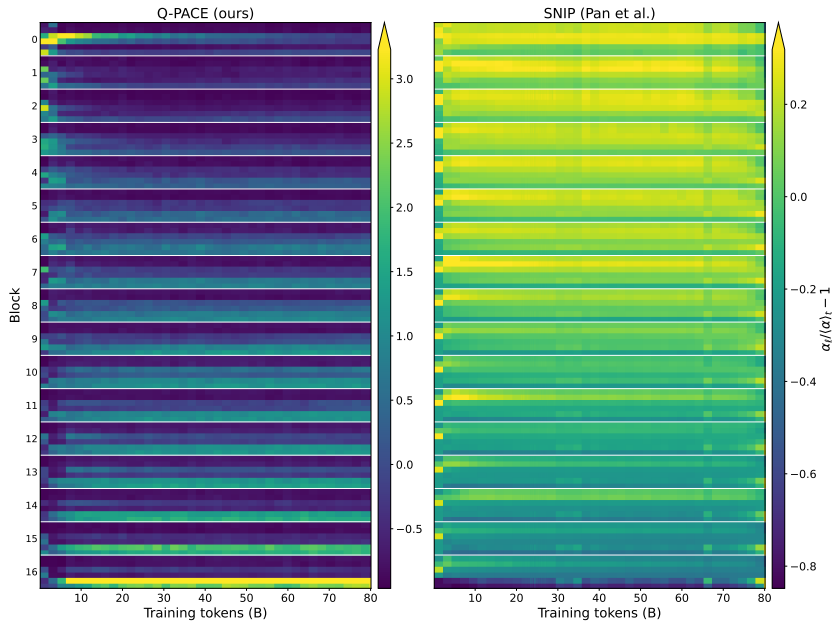

Figure 11: Relative sensitivity over training for the 800M model.

This difference helps explain why the two methods can produce substantially different mixed-precision allocations despite operating under the same global precision budget.

Sensitivity is also strongly structured by model depth. Figure 11 shows the sensitivity α_i normalized by the average sensitivity at that training point. For Q-PACE, later blocks are generally more sensitive, with the final block standing out throughout most of training. After an initial transition, this pattern becomes remarkably stable. SNIP exhibits a qualitatively different depth profile, with higher relative sensitivity concentrated toward earlier blocks and decreasing toward the end of the network. These results further show that quantization sensitivity cannot be described only by layer type or only by depth: Q-PACE captures a structured combination of both factors that remains largely stable over long portions of training.

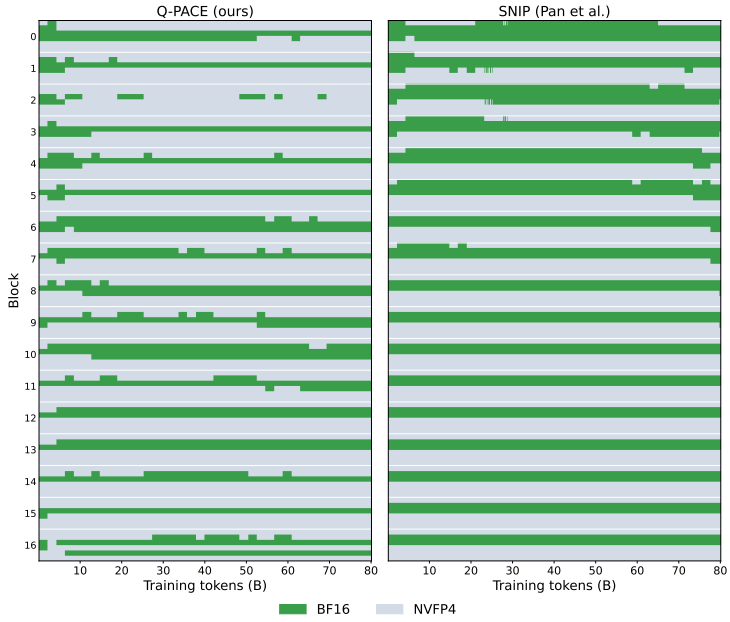

Figure 12: Bitwidth allocations throughout training for 800M model at budget 5.5 bits/parameter with bit choices 4,16.

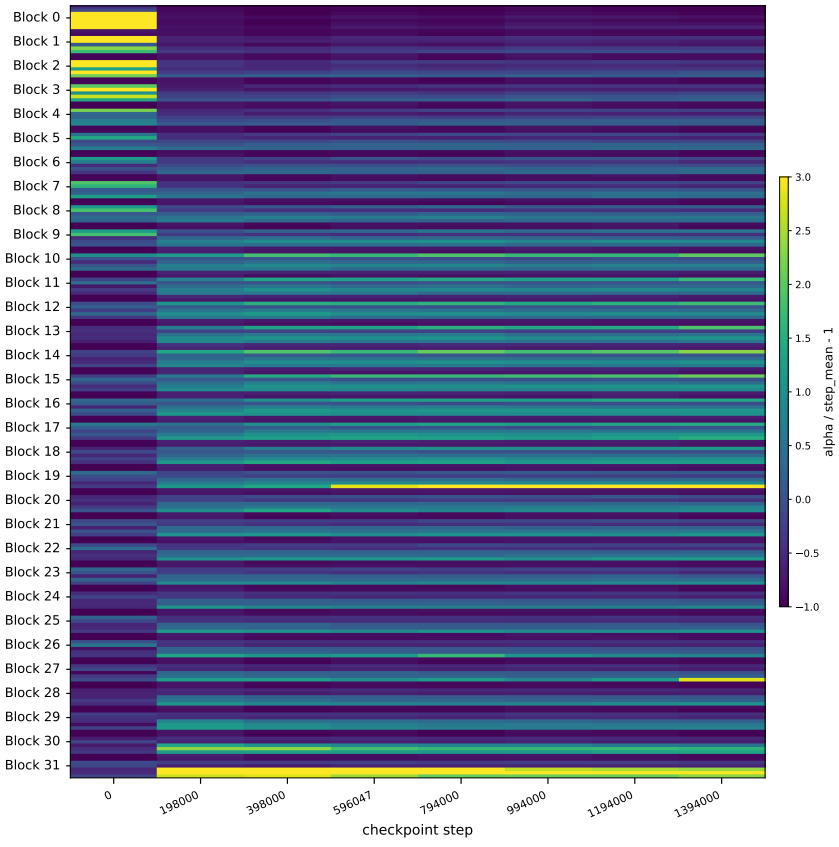

Figure 13: Layer-wise sensitivity heatmap for Olmo-3-7B (Olmo: et al., 2026) during training. Lighter color corresponds to higher sensitivity α.

H BUDGET TRADE-OFF, SENSITIVITY, AND ALLOCATIONS

This section presents the figures supporting the grid and 4B results. Figure 14 displays each allocation on the budget vs. loss plots. At both pretraining sizes and during SFT, Q-PACE achieves greater improvement over uniform NVFP4 than both SNIP variants and NVIDIA at a comparable bit count. The subsequent figures illustrate the distribution of precision. Sensitivity is normalized at each calibration, allowing for comparison over time but not across estimators (Figures 15 and 16). During pretraining, Q-PACE concentrates its score on the MLP up-projections, whereas SNIP distributes it evenly across projection types. In SFT, the score for Q-PACE shifts to the MLP down-projections. The allocations correspond to these scores and exhibit minimal change after the initial recalibration (Figures 17 to 20). Figures 21 to 24 provide a parallel analysis for the 4B runs across all 168 switchable layers.

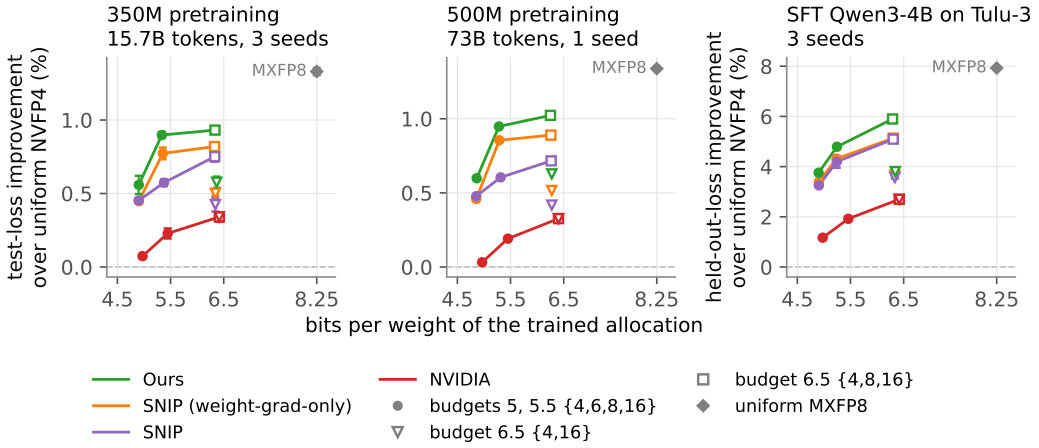

Figure 14: Loss improvement over uniform NVFP4 against the realized bits per weight of the trained allocation, for pretraining at 350M and 500M and for SFT.

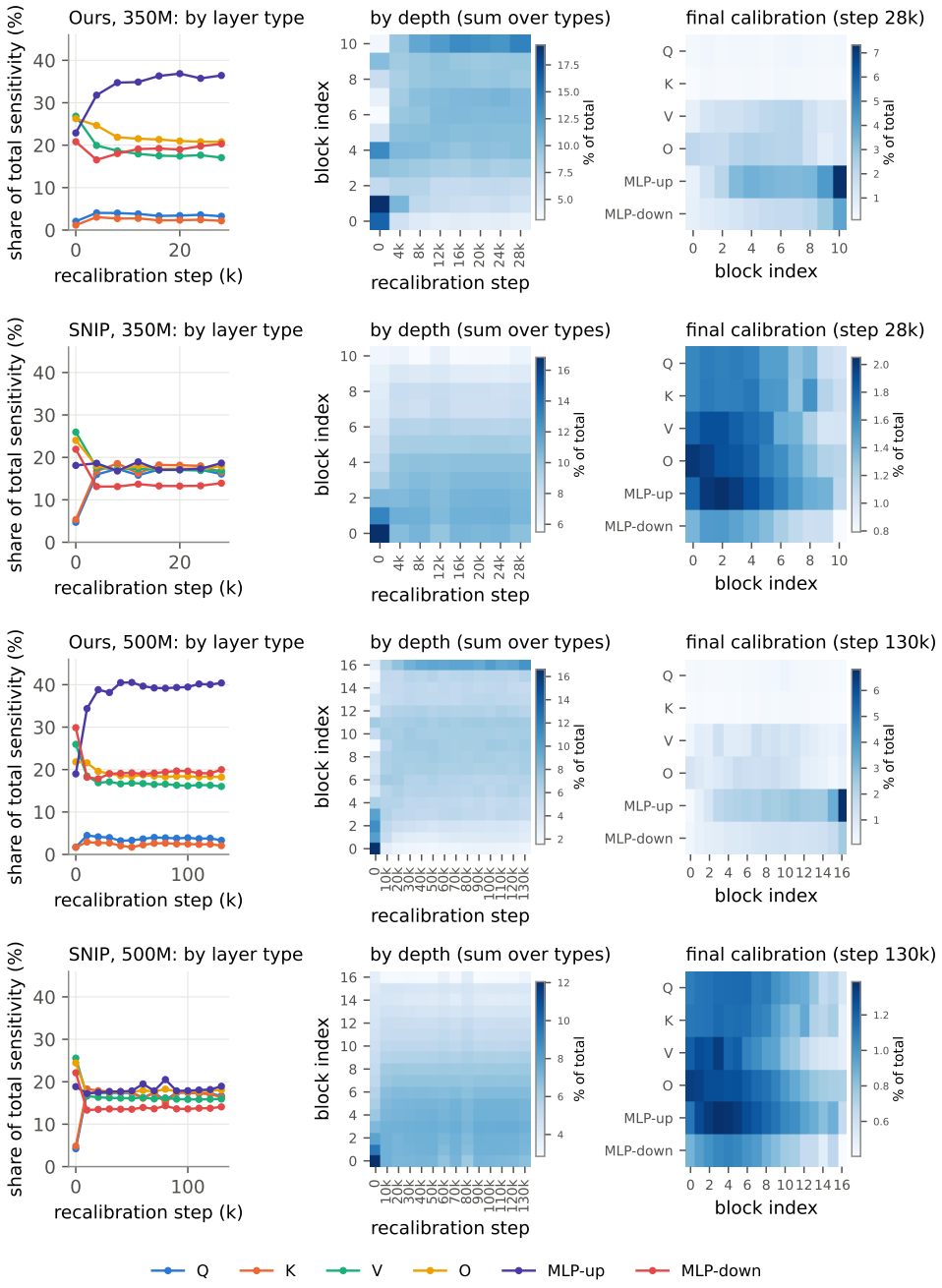

Figure 15: Normalized sensitivity of Q-PACE and SNIP at budget 6.5 on $\{4, 8, 16\}$ at 350M and 500M. Type shares over the calibrations, depth shares, and the final type-by-block map.

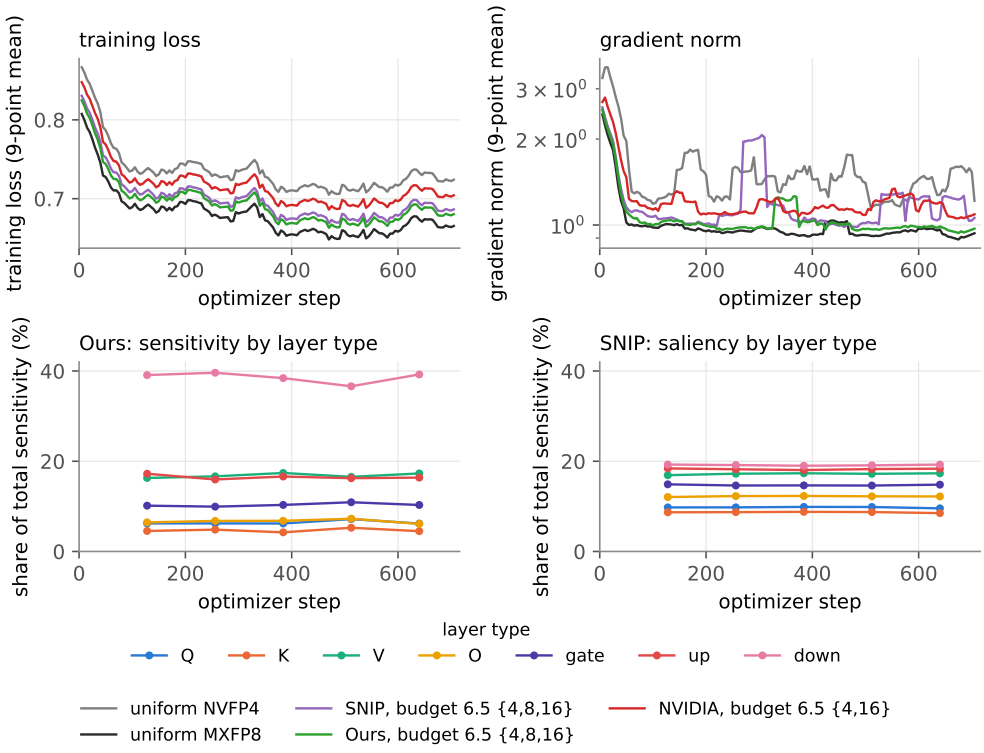

Figure 16: SFT training loss and gradient norm (mean of three seeds) and the normalized sensitivity by projection type of Q-PACE and SNIP at budget 6.5 on $\{4, 8, 16\}$.

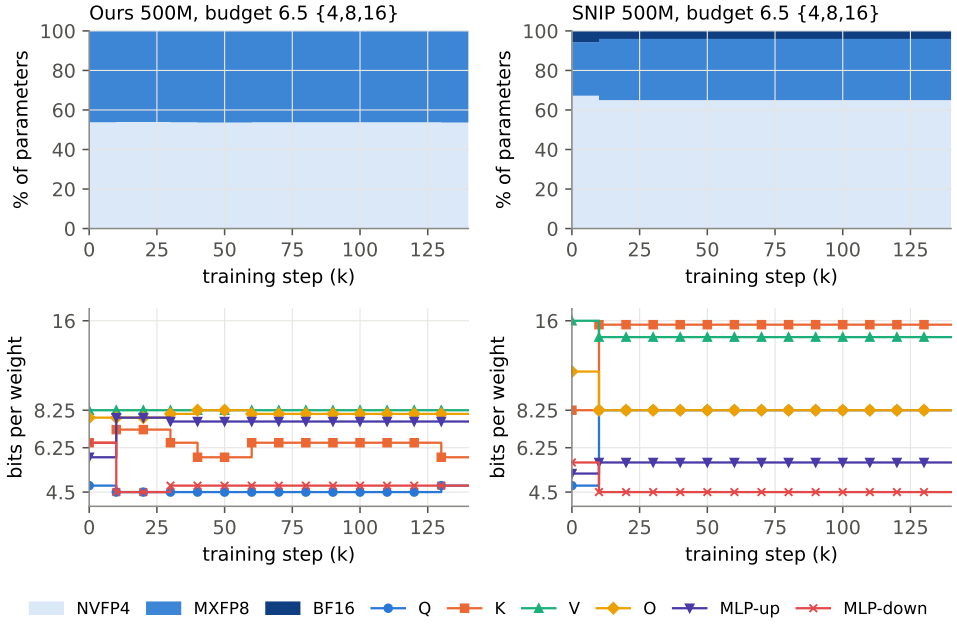

Figure 17: 500M allocations of Q-PACE and SNIP at budget 6.5 on $\{4, 8, 16\}$ over training. Parameter share per format and bits per weight by projection type.

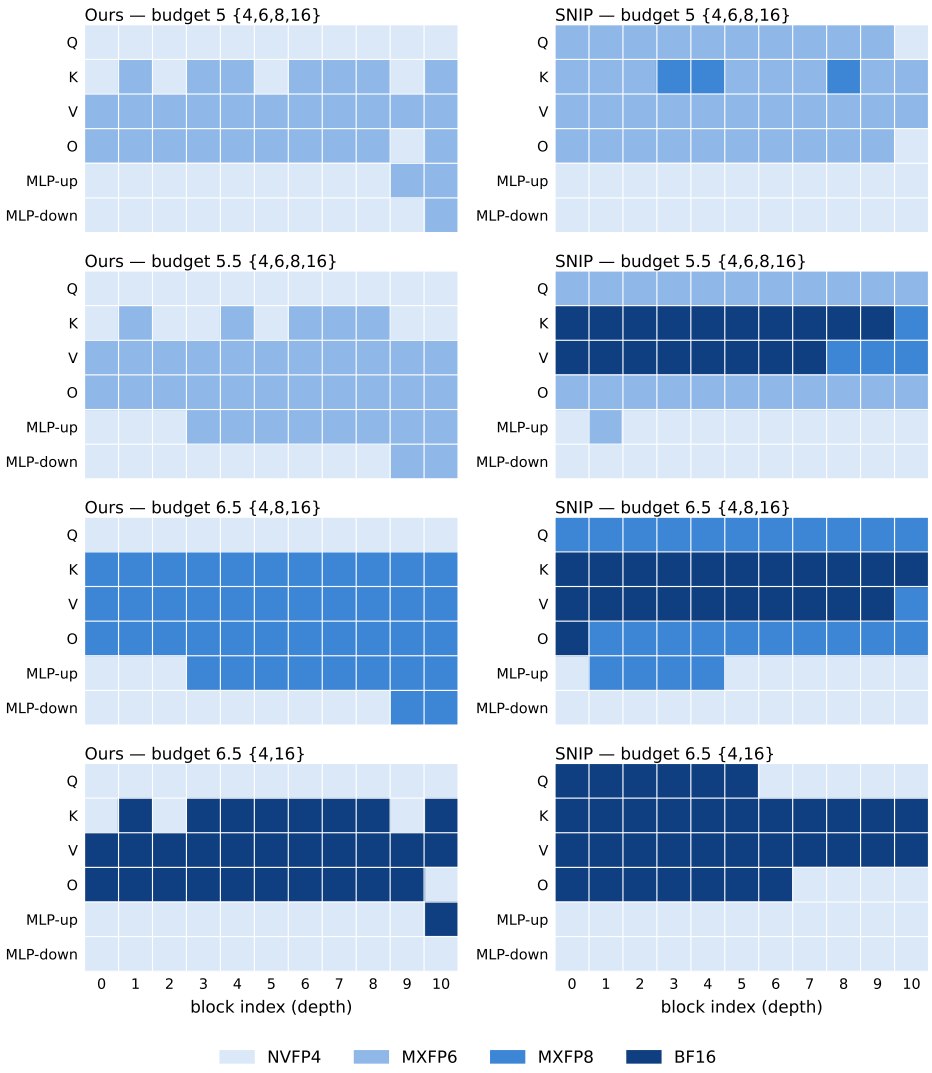

Figure 18: Final 350M allocations of Q-PACE and SNIP for every budget and menu.

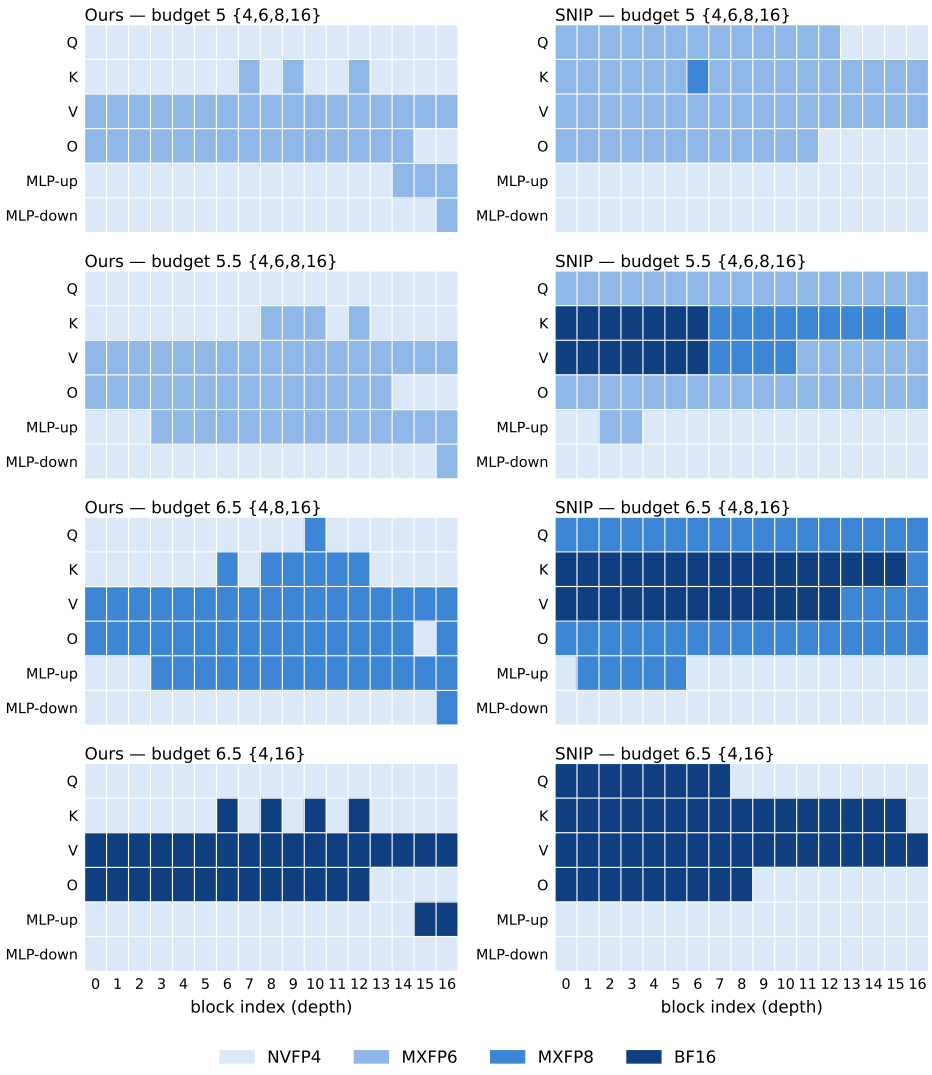

Figure 19: Final 500M allocations of Q-PACE and SNIP for every budget and menu.

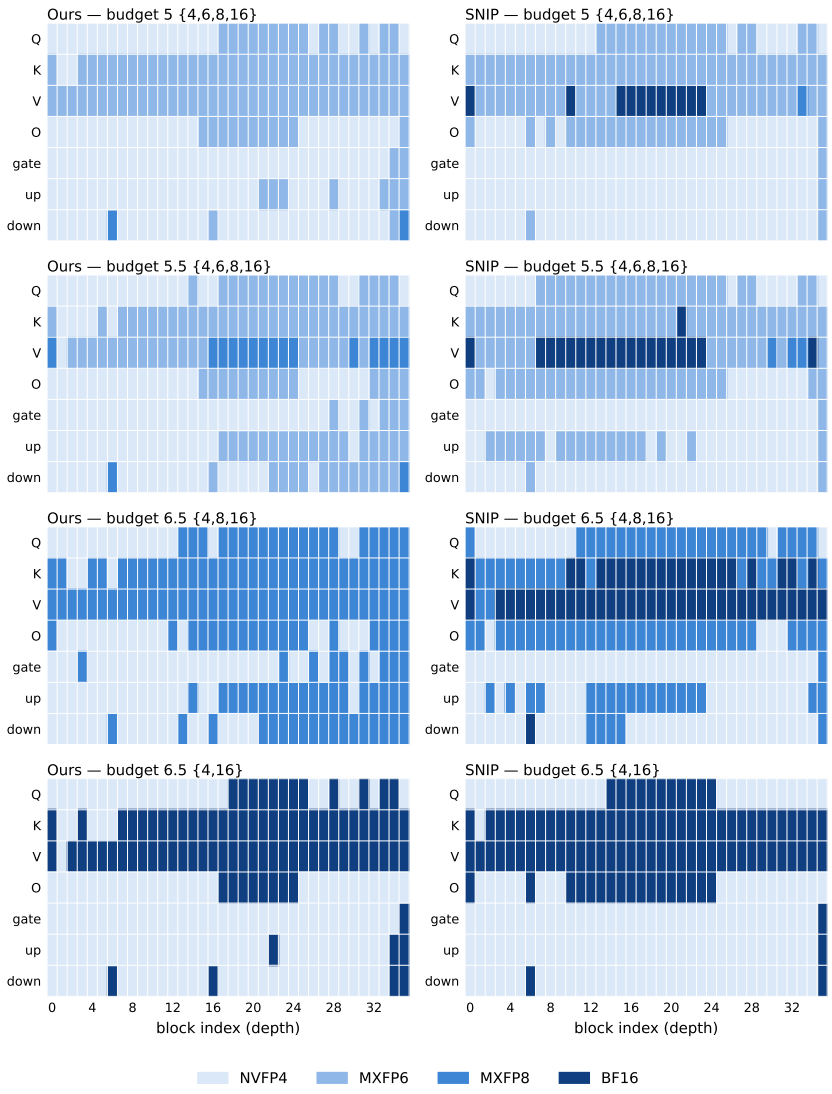

Figure 20: Final SFT allocations of Q-PACE and SNIP for every budget and menu. Qwen3 has three MLP projections per block.

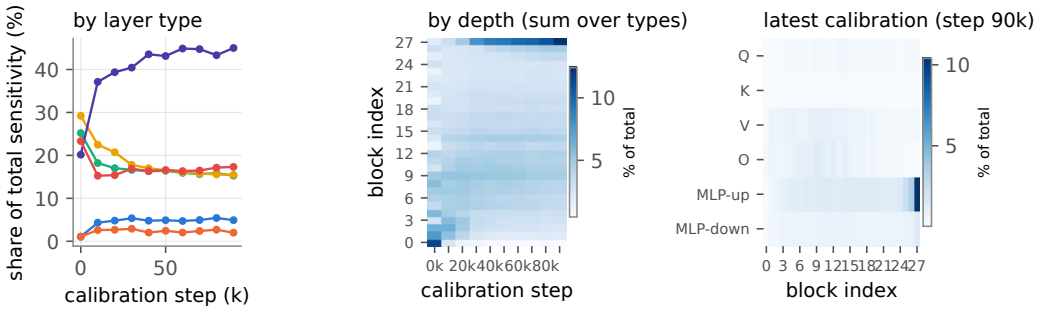

Figure 21a: Q-PACE

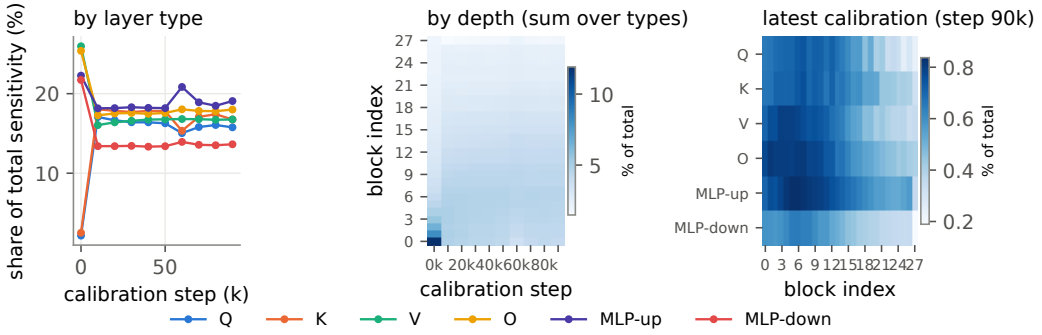

Figure 21b: SNIP

Figure 21: 4B normalized sensitivity at budget 6.5 on $\{4, 16\}$. Type shares over the calibrations, depth shares, and the latest type-by-block map.

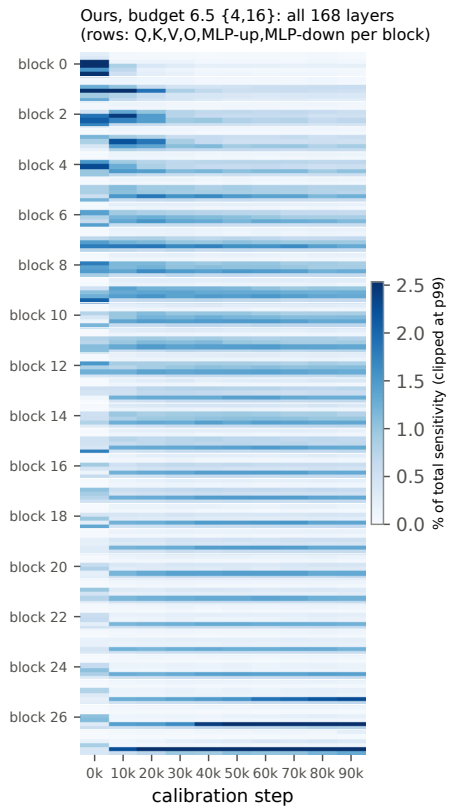

Figure 22a: Q-PACE

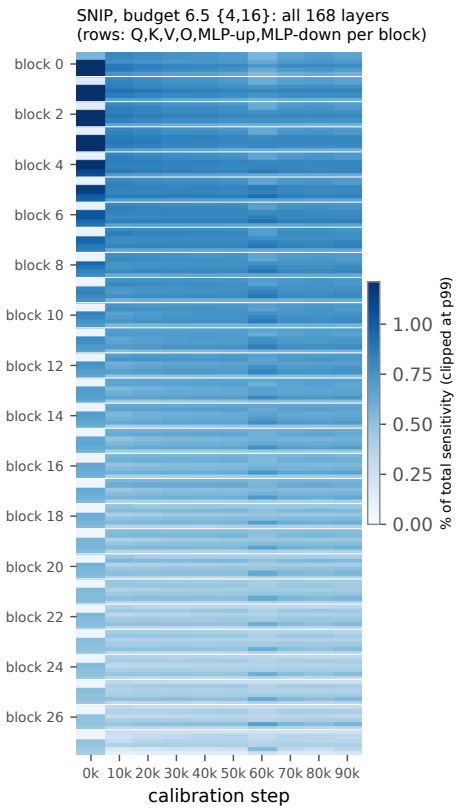

Figure 22b: SNIP

Figure 22: 4B sensitivity of all 168 layers across the calibrations at budget 6.5 on $\{4, 16\}$. Colors are clipped at the 99th percentile, on separate scales for the two estimators.

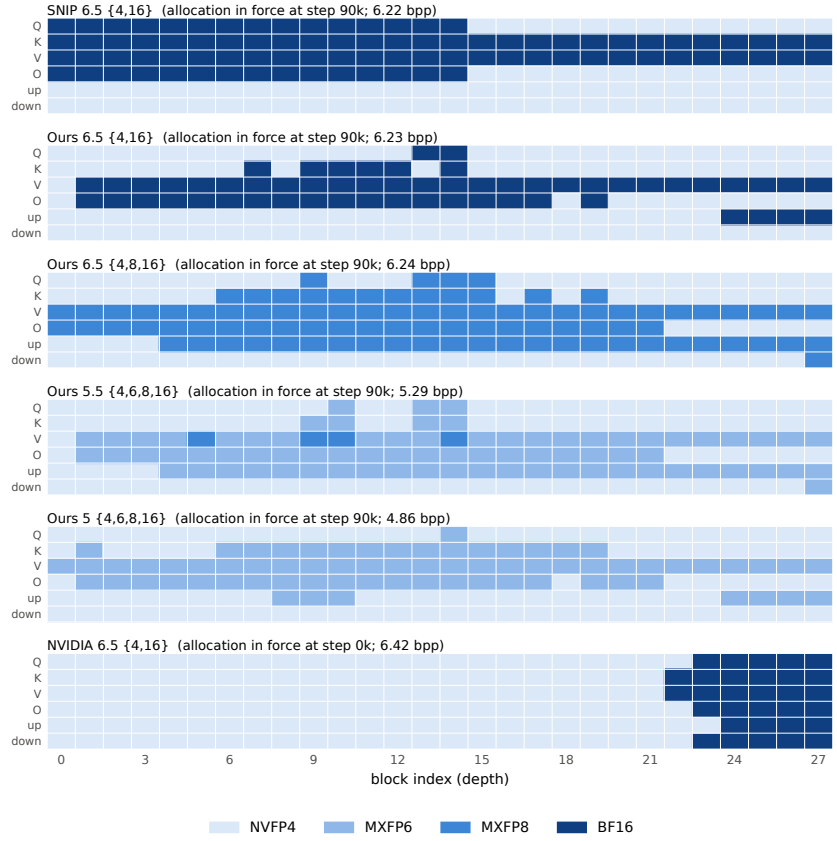

Figure 23a: Final allocations

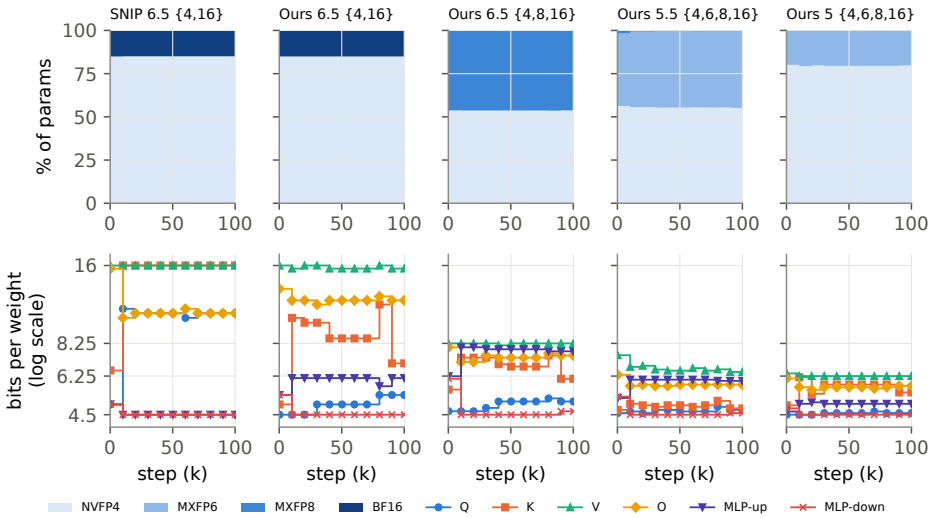

Figure 23b: Allocation dynamics

Figure 23: 4B final allocations and their dynamics. Parameter share per format and bits per weight by projection type.

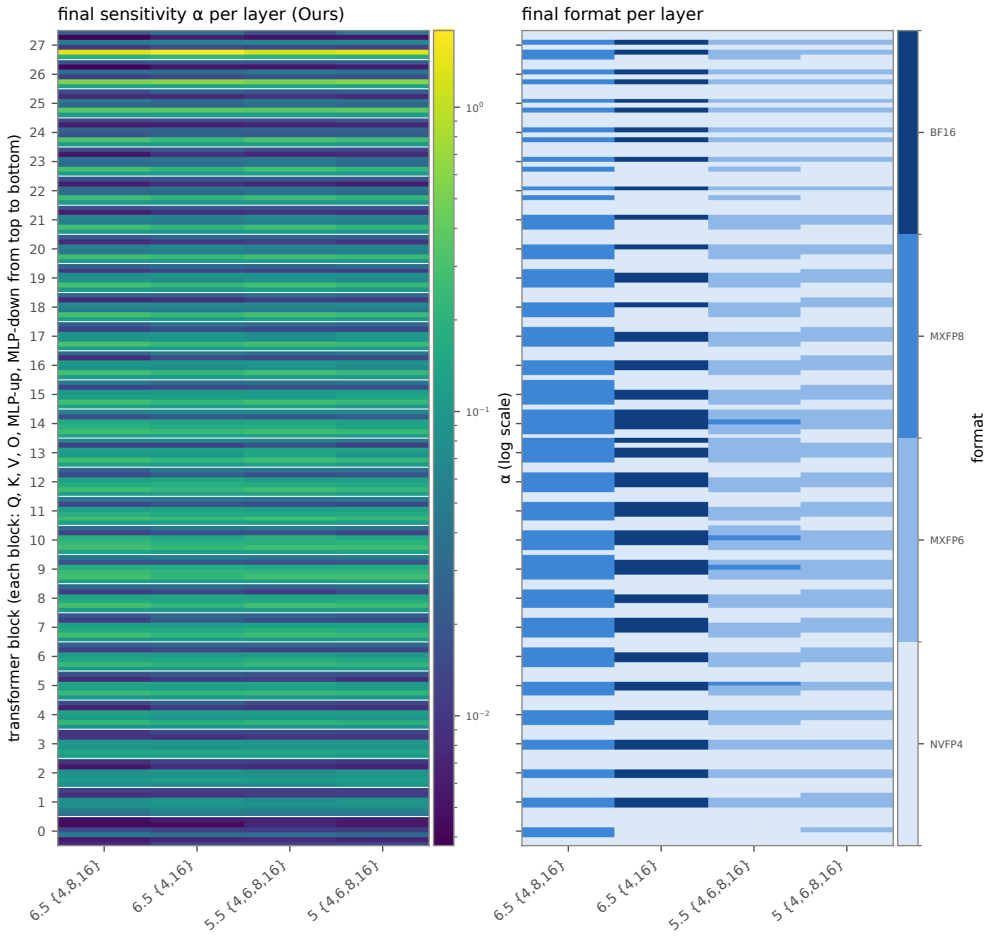

Figure 24: Final sensitivity (logarithmic color scale) and format of all 168 switchable layers in the four 4B Q-PACE runs.

H.1 DOWNSTREAM TASK SCORES

Accuracies are reported as percentages at the trained allocation and are averaged over three random seeds. The downstream mean presented in Table 2 represents the average performance across 24 distinct tasks, with Natural Questions counted once. All tasks are evaluated in a zero-shot setting, except for GSM8K, which is assessed using an eight-shot chain-of-thought approach. Evaluation is conducted using the LM Evaluation Harness (Gao et al., 2023).

- Generation and instruction following (Table 7): GSM8K (Cobbe et al., 2021), exact match; IFEval (Zhou et al., 2023), strict prompt-level accuracy; HumanEval (Chen et al., 2021), pass@1; Natural Questions (Kwiatkowski et al., 2019), exact match; TruthfulQA (Lin et al., 2022), MC2 accuracy.
- Knowledge and likelihood (Table 8): ARC-Challenge, ARC-Easy, HellaSwag, PIQA, WinoGrande, LAMBADA, and MMLU, accuracy.
- Probe tasks (Table 9): BoolQ, COPA, RACE, ASDiv, and CommonsenseQA, accuracy; LogiQA, SAT-Math, SciQ, OpenBookQA, AQuA-RAT, LSAT-AR, and LSAT-LR, length-normalized accuracy; Natural Questions, exact match.

Table 7: SFT generative and instruction benchmarks (accuracy %, mean ± seed standard deviation over three seeds).

run	GSM8K	IFEval	HumanEval	NQ	TQA
uniform references					
uniform NVFP4	$80.6_{\pm 1.4}$	$30.6_{\pm 0.8}$	$69.1_{\pm 2.9}$	$8.7_{\pm 0.4}$	$51.2_{\pm 0.7}$
uniform MXFP8	$84.2_{\pm 0.3}$	$34.3_{\pm 1.1}$	$75.8_{\pm 0.9}$	$11.2_{\pm 0.7}$	$52.6_{\pm 0.1}$
budget 5, menu $\{4, 6, 8, 16\}$					
NVIDIA	$79.6_{\pm 0.8}$	$28.7_{\pm 1.2}$	$70.9_{\pm 0.7}$	$8.8_{\pm 0.2}$	$\mathbf{50.7}_{\pm 0.8}$
Q-PACE	$80.2_{\pm 1.0}$	$\mathbf{34.4}_{\pm 1.1}$	$71.3_{\pm 2.7}$	$8.8_{\pm 0.5}$	$49.9_{\pm 0.3}$
SNIP	$80.9_{\pm 1.0}$	$33.3_{\pm 1.2}$	$72.2_{\pm 1.8}$	$\mathbf{9.4}_{\pm 0.3}$	$49.7_{\pm 0.6}$
budget 5.5, menu $\{4, 6, 8, 16\}$					
NVIDIA	$78.2_{\pm 2.2}$	$30.5_{\pm 0.7}$	$71.1_{\pm 1.5}$	$9.2_{\pm 0.4}$	$\mathbf{50.7}_{\pm 0.4}$
Q-PACE	$82.9_{\pm 1.0}$	$33.8_{\pm 0.6}$	$\mathbf{74.2}_{\pm 1.9}$	$9.6_{\pm 1.0}$	$50.6_{\pm 0.4}$
SNIP	$\mathbf{83.2}_{\pm 0.3}$	$33.6_{\pm 1.6}$	$71.7_{\pm 3.1}$	$\mathbf{9.7}_{\pm 0.6}$	$50.1_{\pm 0.3}$
budget 6.5, menu $\{4, 8, 16\}$					
NVIDIA	$79.2_{\pm 1.4}$	$28.4_{\pm 1.0}$	$69.7_{\pm 2.1}$	$9.6_{\pm 0.5}$	$50.2_{\pm 0.5}$
Q-PACE	$78.2_{\pm 3.4}$	$32.3_{\pm 0.7}$	$\mathbf{74.2}_{\pm 2.0}$	$\mathbf{9.8}_{\pm 0.4}$	$50.5_{\pm 0.9}$
SNIP	$82.8_{\pm 1.1}$	$\mathbf{34.1}_{\pm 1.5}$	$74.0_{\pm 1.4}$	$9.3_{\pm 0.1}$	$\mathbf{51.0}_{\pm 0.3}$
budget 6.5, menu $\{4, 16\}$					
NVIDIA	$79.2_{\pm 1.4}$	$28.4_{\pm 1.0}$	$69.7_{\pm 2.1}$	$\mathbf{9.6}_{\pm 0.5}$	$50.2_{\pm 0.5}$
Q-PACE	$\mathbf{80.1}_{\pm 2.2}$	$\mathbf{34.6}_{\pm 0.8}$	$71.3_{\pm 1.1}$	$9.1_{\pm 0.6}$	$50.0_{\pm 0.7}$
SNIP	$79.2_{\pm 1.6}$	$33.8_{\pm 0.9}$	$71.7_{\pm 2.5}$	$8.9_{\pm 0.1}$	$\mathbf{50.3}_{\pm 0.1}$

Table 8: SFT knowledge/likelihood tasks (accuracy %, mean ± seed standard deviation over three seeds).

run	ARC-c	ARC-e	HellaSw	PIQA	Wino	LAMBADA	MMLU
uniform references							
uniform NVFP4	$44.9_{\pm 0.6}$	$74.6_{\pm 0.3}$	$53.8_{\pm 0.1}$	$76.1_{\pm 0.4}$	$67.6_{\pm 0.8}$	$66.3_{\pm 0.2}$	$68.7_{\pm 0.1}$
uniform MXFP8	$47.0_{\pm 0.2}$	$78.4_{\pm 0.1}$	$54.6_{\pm 0.0}$	$78.0_{\pm 0.1}$	$70.7_{\pm 0.6}$	$68.6_{\pm 0.1}$	$71.1_{\pm 0.1}$
budget 5, menu $\{4, 6, 8, 16\}$							
NVIDIA	$45.1_{\pm 1.1}$	$75.2_{\pm 0.6}$	$53.6_{\pm 0.1}$	$75.9_{\pm 0.1}$	$68.3_{\pm 1.2}$	$\mathbf{67.3}_{\pm 0.3}$	$68.9_{\pm 0.1}$
Q-PACE	$46.1_{\pm 0.5}$	$76.7_{\pm 0.3}$	$54.0_{\pm 0.1}$	$76.3_{\pm 0.3}$	$\mathbf{69.2}_{\pm 0.9}$	$66.7_{\pm 0.1}$	$69.7_{\pm 0.0}$
SNIP	$46.2_{\pm 0.4}$	$\mathbf{77.3}_{\pm 0.2}$	$\mathbf{54.1}_{\pm 0.1}$	$\mathbf{76.8}_{\pm 0.1}$	$68.3_{\pm 1.1}$	$66.3_{\pm 0.5}$	$69.9_{\pm 0.1}$
budget 5.5, menu $\{4, 6, 8, 16\}$							
NVIDIA	$44.9_{\pm 1.1}$	$74.9_{\pm 0.8}$	$53.6_{\pm 0.1}$	$76.3_{\pm 0.2}$	$68.6_{\pm 0.9}$	$67.5_{\pm 0.2}$	$68.8_{\pm 0.2}$
Q-PACE	$47.0_{\pm 0.3}$	$77.2_{\pm 1.0}$	$54.1_{\pm 0.1}$	$76.9_{\pm 0.9}$	$\mathbf{69.5}_{\pm 0.1}$	$\mathbf{67.8}_{\pm 0.1}$	$69.8_{\pm 0.0}$
SNIP	$47.0_{\pm 1.3}$	$76.9_{\pm 0.3}$	$\mathbf{54.2}_{\pm 0.1}$	$\mathbf{77.0}_{\pm 0.2}$	$69.2_{\pm 0.5}$	$66.7_{\pm 0.7}$	$\mathbf{70.2}_{\pm 0.1}$
budget 6.5, menu $\{4, 8, 16\}$							
NVIDIA	$45.3_{\pm 0.8}$	$75.5_{\pm 0.7}$	$53.8_{\pm 0.2}$	$76.4_{\pm 0.3}$	$67.6_{\pm 0.6}$	$67.3_{\pm 0.4}$	$68.9_{\pm 0.3}$
Q-PACE	$\mathbf{47.6}_{\pm 0.9}$	$77.7_{\pm 1.3}$	$54.1_{\pm 0.0}$	$77.0_{\pm 0.2}$	$\mathbf{69.0}_{\pm 0.3}$	$\mathbf{67.8}_{\pm 0.2}$	$70.1_{\pm 0.0}$
SNIP	$47.0_{\pm 0.6}$	$77.5_{\pm 0.7}$	$\mathbf{54.1}_{\pm 0.1}$	$\mathbf{77.2}_{\pm 0.4}$	$68.5_{\pm 0.3}$	$67.3_{\pm 0.2}$	$\mathbf{70.4}_{\pm 0.1}$
budget 6.5, menu $\{4, 16\}$							
NVIDIA	$45.3_{\pm 0.8}$	$75.5_{\pm 0.7}$	$53.8_{\pm 0.2}$	$76.4_{\pm 0.3}$	$67.6_{\pm 0.6}$	$67.3_{\pm 0.4}$	$68.9_{\pm 0.3}$
Q-PACE	$46.6_{\pm 0.9}$	$76.6_{\pm 0.5}$	$54.0_{\pm 0.2}$	$76.5_{\pm 0.5}$	$68.3_{\pm 0.5}$	$66.9_{\pm 0.3}$	$69.6_{\pm 0.0}$
SNIP	$46.9_{\pm 0.5}$	$\mathbf{77.4}_{\pm 0.5}$	$\mathbf{54.1}_{\pm 0.0}$	$\mathbf{76.6}_{\pm 0.2}$	$\mathbf{68.9}_{\pm 0.5}$	$\mathbf{67.4}_{\pm 0.2}$	$\mathbf{69.9}_{\pm 0.0}$

Table 9: SFT probe tasks (accuracy %, mean ± seed standard deviation over three seeds).

run	BoolQ	Copa	RACE	LogiQA	ASDiv	SAT-Math	NQ
uniform references							
uniform NVFP4	$77.1_{\pm 0.2}$	$84.0_{\pm 1.7}$	$39.5_{\pm 0.7}$	$40.2_{\pm 0.7}$	$27.9_{\pm 0.2}$	$43.6_{\pm 2.0}$	$8.7_{\pm 0.4}$
uniform MXFP8	$81.2_{\pm 0.3}$	$83.0_{\pm 0.0}$	$40.5_{\pm 0.1}$	$42.8_{\pm 0.6}$	$26.6_{\pm 0.5}$	$48.5_{\pm 2.1}$	$11.2_{\pm 0.7}$
budget 5, menu $\{4, 6, 8, 16\}$							
NVIDIA	$79.3_{\pm 0.4}$	$\mathbf{85.0}_{\pm 1.7}$	$39.8_{\pm 1.0}$	$40.3_{\pm 1.5}$	$26.2_{\pm 0.3}$	$44.2_{\pm 1.4}$	$8.8_{\pm 0.2}$
Q-PACE	$\mathbf{80.7}_{\pm 0.1}$	$84.0_{\pm 1.0}$	$39.7_{\pm 0.7}$	$41.1_{\pm 1.6}$	$29.6_{\pm 0.5}$	$44.5_{\pm 0.9}$	$8.8_{\pm 0.5}$
SNIP	$80.3_{\pm 0.3}$	$83.0_{\pm 2.0}$	$39.9_{\pm 0.6}$	$40.5_{\pm 1.5}$	$\mathbf{32.4}_{\pm 0.8}$	$42.9_{\pm 0.9}$	$\mathbf{9.4}_{\pm 0.3}$
budget 5.5, menu $\{4, 6, 8, 16\}$							
NVIDIA	$80.0_{\pm 0.5}$	$\mathbf{86.0}_{\pm 1.0}$	$39.6_{\pm 0.5}$	$39.9_{\pm 0.9}$	$26.6_{\pm 1.0}$	$47.1_{\pm 1.6}$	$9.2_{\pm 0.4}$
Q-PACE	$\mathbf{82.1}_{\pm 0.6}$	$85.0_{\pm 0.0}$	$\mathbf{40.3}_{\pm 0.4}$	$\mathbf{42.7}_{\pm 1.3}$	$27.8_{\pm 0.9}$	$\mathbf{49.1}_{\pm 1.4}$	$9.6_{\pm 1.0}$
SNIP	$79.8_{\pm 0.2}$	$84.3_{\pm 1.5}$	$39.7_{\pm 0.4}$	$41.7_{\pm 1.1}$	$31.4_{\pm 3.9}$	$45.5_{\pm 1.2}$	$\mathbf{9.7}_{\pm 0.6}$
budget 6.5, menu $\{4, 8, 16\}$							
NVIDIA	$79.6_{\pm 0.5}$	$83.3_{\pm 2.1}$	$39.3_{\pm 0.6}$	$39.6_{\pm 1.4}$	$26.2_{\pm 0.8}$	$44.1_{\pm 2.1}$	$9.6_{\pm 0.5}$
Q-PACE	$\mathbf{82.3}_{\pm 0.6}$	$\mathbf{85.0}_{\pm 0.0}$	$\mathbf{40.3}_{\pm 0.3}$	$40.8_{\pm 1.3}$	$30.4_{\pm 0.6}$	$47.7_{\pm 1.6}$	$\mathbf{9.8}_{\pm 0.4}$
SNIP	$79.9_{\pm 0.2}$	$85.0_{\pm 1.0}$	$39.7_{\pm 0.5}$	$42.1_{\pm 0.9}$	$\mathbf{33.7}_{\pm 1.3}$	$\mathbf{48.2}_{\pm 0.9}$	$9.3_{\pm 0.1}$
budget 6.5, menu $\{4, 16\}$							
NVIDIA	$79.6_{\pm 0.5}$	$83.3_{\pm 2.1}$	$39.3_{\pm 0.6}$	$39.6_{\pm 1.4}$	$26.2_{\pm 0.8}$	$44.1_{\pm 2.1}$	$\mathbf{9.6}_{\pm 0.5}$
Q-PACE	$80.1_{\pm 0.1}$	$84.0_{\pm 2.0}$	$\mathbf{41.3}_{\pm 0.6}$	$38.9_{\pm 2.4}$	$27.9_{\pm 0.3}$	$\mathbf{48.5}_{\pm 1.6}$	$9.1_{\pm 0.6}$
SNIP	$80.4_{\pm 0.2}$	$84.3_{\pm 1.2}$	$40.4_{\pm 0.1}$	$40.8_{\pm 1.0}$	$\mathbf{32.8}_{\pm 0.9}$	$43.8_{\pm 2.7}$	$8.9_{\pm 0.1}$

run	SciQ	OBQA	CSQA	AQuA	LSAT-ar	LSAT-lr
uniform references						
uniform NVFP4	$94.2_{\pm 0.4}$	$41.2_{\pm 0.9}$	$79.2_{\pm 0.8}$	$39.5_{\pm 1.6}$	$18.0_{\pm 1.8}$	$43.1_{\pm 1.2}$
uniform MXFP8	$94.5_{\pm 0.2}$	$40.3_{\pm 0.6}$	$82.1_{\pm 0.5}$	$40.6_{\pm 2.2}$	$19.4_{\pm 0.5}$	$48.6_{\pm 0.2}$
budget 5, menu $\{4, 6, 8, 16\}$						
NVIDIA	$94.1_{\pm 0.1}$	$40.1_{\pm 0.8}$	$78.8_{\pm 0.1}$	$37.9_{\pm 4.3}$	$17.7_{\pm 1.6}$	$42.7_{\pm 2.2}$
Q-PACE	$94.8_{\pm 0.2}$	$40.5_{\pm 0.7}$	$79.6_{\pm 0.4}$	$\mathbf{38.3}_{\pm 1.9}$	$20.1_{\pm 0.9}$	$43.7_{\pm 1.0}$
SNIP	$\mathbf{94.9}_{\pm 0.3}$	$41.4_{\pm 0.3}$	$79.6_{\pm 0.2}$	$37.5_{\pm 1.2}$	$\mathbf{22.5}_{\pm 0.9}$	$43.5_{\pm 1.5}$
budget 5.5, menu $\{4, 6, 8, 16\}$						
NVIDIA	$94.3_{\pm 0.3}$	$\mathbf{41.4}_{\pm 0.5}$	$79.1_{\pm 0.3}$	$39.9_{\pm 0.2}$	$19.6_{\pm 2.6}$	$41.4_{\pm 0.5}$
Q-PACE	$\mathbf{94.8}_{\pm 0.2}$	$40.6_{\pm 0.2}$	$79.2_{\pm 0.7}$	$37.4_{\pm 0.4}$	$18.6_{\pm 2.8}$	$43.3_{\pm 1.2}$
SNIP	$94.7_{\pm 0.3}$	$41.3_{\pm 0.5}$	$79.9_{\pm 0.8}$	$\mathbf{40.3}_{\pm 0.5}$	$20.0_{\pm 0.4}$	$43.3_{\pm 1.4}$
budget 6.5, menu $\{4, 8, 16\}$						
NVIDIA	$94.3_{\pm 0.2}$	$40.3_{\pm 0.6}$	$78.6_{\pm 0.6}$	$39.2_{\pm 4.7}$	$19.3_{\pm 1.0}$	$42.0_{\pm 0.7}$
Q-PACE	$\mathbf{94.6}_{\pm 0.1}$	$\mathbf{40.8}_{\pm 0.9}$	$79.7_{\pm 0.8}$	$38.1_{\pm 1.9}$	$19.6_{\pm 2.3}$	$44.4_{\pm 1.5}$
SNIP	$94.4_{\pm 0.2}$	$40.7_{\pm 0.1}$	$\mathbf{80.8}_{\pm 0.8}$	$\mathbf{39.6}_{\pm 2.2}$	$\mathbf{20.7}_{\pm 0.7}$	$43.3_{\pm 0.7}$
budget 6.5, menu $\{4, 16\}$						
NVIDIA	$94.3_{\pm 0.2}$	$40.3_{\pm 0.6}$	$78.6_{\pm 0.6}$	$39.2_{\pm 4.7}$	$19.3_{\pm 1.0}$	$42.0_{\pm 0.7}$
Q-PACE	$\mathbf{94.9}_{\pm 0.5}$	$\mathbf{41.5}_{\pm 0.9}$	$79.1_{\pm 0.5}$	$39.0_{\pm 1.2}$	$20.4_{\pm 1.9}$	$\mathbf{43.5}_{\pm 2.1}$
SNIP	$94.6_{\pm 0.3}$	$40.9_{\pm 0.3}$	$79.5_{\pm 0.3}$	$38.5_{\pm 1.2}$	$20.6_{\pm 2.1}$	$42.4_{\pm 1.6}$