TACO: TERNARY ABSOLUTE-MAX COLUMN-WISE ONE-SPARSE OPTIMIZER FOR LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Full-parameter fine-tuning of large language models (LLMs) incurs substantial optimizer state memory overhead, limiting the model sizes that fit on modern GPUs. Existing approaches either compress optimizer state, abandon first-order gradients, or change the update geometry while retaining dense state. The recently introduced Muon optimizer reduces optimizer memory through matrix-valued updates. Still, its geometry differs from AdamW and can lead to performance degradation when fine-tuning AdamW-pretrained models. To reduce optimizer memory without sacrificing accuracy or computational efficiency in LLM fine-tuning, we propose **T**ernary **A**bsolute-max **C**olumn-wise **O**ne-sparse optimizer, or **TACO**, which follows Muon's operator-norm steepest-descent view but takes the geometric route further. TACO computes the exact steepest-descent direction under a dimension-normalized $1 \to 1$ operator norm by selecting the sign of the largest magnitude entry in each column of two-dimensional weight matrices. This retains first-order gradients while making optimizer state memory nearly negligible. Our practical TACO optimizer maintains only a small set of low precision gradient components per column, reducing persistent optimizer state by $174\times$ relative to AdamW8bit (from 27.7 GB to 0.16 GB) and peak training memory by $2.9\times$ (from 80.6 GB to 27.5 GB) on OPT-13B, while achieving comparable accuracy and runtime. TACO further enables full-parameter fine-tuning of 30-32B-parameter models on a single 80 GB H100 GPU across multiple model families and tasks.

1 INTRODUCTION

Full-parameter fine-tuning adapts all model weights and remains a strong baseline for transferring large language models (LLMs) to downstream tasks (Houlsby et al., 2019; Hu et al., 2021); however, it often suffers from memory overhead. For instance, a 13B-parameter model occupies roughly 26 GB of memory in `BF16` before allocating any training state. During AdamW (Loshchilov & Hutter, 2019) fine-tuning, the two `FP32` moment buffers add another ~ 104 GB of memory. At the same time, gradients contribute ~ 26 GB, yielding roughly 156 GB before accounting for activations, temporary workspaces, or an optional `FP32` master copy; see the *Pareto frontier memory-accuracy trade-off* graph in Figure 1. Thus, at this scale, AdamW's persistent optimizer states alone require roughly $4\times$ the `BF16` parameter memory.

Several memory-efficient first-order optimizers reduce this overhead while retaining backpropagation, including state-quantizers (AdamW8bit quantizes optimizer states (Dettmers et al., 2021); FlashAdamW combines 8-bit optimizer states with a compact master-weight representation (Gonzalez Ortiz et al., 2026)), factorized second moment methods (Adafactor uses factorized second-moment estimates (Shazeer & Stern, 2018)), and low-rank gradient methods (GaLore projects gradients and optimizer states into a low-rank subspace (Zhao et al., 2024), and APOLLO-Mini uses a rank-1 subspace to estimate adaptive scaling factors (Zhu et al., 2025)). In contrast, zeroth-order (ZO) methods avoid backpropagation and approach inference-level memory by estimating updates from forward evaluations (Bergou et al., 2020; Bibi et al., 2020; Gorbunov et al., 2020; Cai et al., 2021; Malladi et al., 2023; Boucherouite et al., 2024). Recent variants incorporate low-rank subspaces or curvature information to improve query efficiency (Yu et al., 2025; Zhao et al., 2025; Kim et al., 2025), but still require additional forward queries.

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.

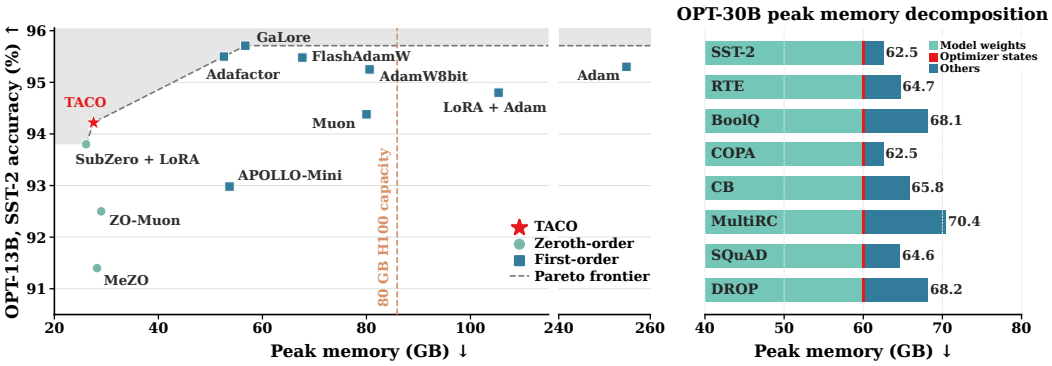

Figure 1: **Performance–memory overview of TACO and other optimizers.** *Left:* Pareto frontier memory-accuracy trade-off for fine-tuning OPT-13B on the SST-2 dataset on an 80 GB H100 GPU. TACO lies at the low-memory end of the Pareto frontier, achieving 94.22% accuracy with 27.54 GB of peak training memory. The dashed line connects the Pareto-optimal methods. The vertical line marks the 80 GB H100 memory limit. Adam achieves 95.3% accuracy with 254.8 GB of peak memory. *Right:* Peak GPU memory decomposition of TACO on OPT-30B across 8 downstream tasks. The bars show model state, optimizer state, and other transient memory across all eight downstream tasks. TACO's optimizer state occupies only 0.267 GB, less than 0.5% of peak memory on each task, while the transient memory accounts for the variation across tasks.

The recently introduced Muon optimizer (Jordan, 2024; Bernstein & Newhouse, 2024; Shah et al., 2025) asks a different question: *what steepest-descent direction under a matrix geometry makes the update useful?* Muon treats two-dimensional weights as matrices, retains a single momentum matrix, and orthogonalizes it to approximate a steepest-descent step under *spectral geometry*. This reduces optimizer memory relative to AdamW, although Muon's state remains dense. Additionally, switching an AdamW-pretrained model to Muon during fine-tuning can degrade performance, a phenomenon called *optimizer mismatch* that is attributed to the different update geometries and implicit biases of the two optimizers (Qu et al., 2026).

Zeroth-Order Muon (ZO-Muon) (Lang et al., 2026) improves query efficiency, but as a zeroth-order method, it requires additional forward evaluations. Sequential evaluation for ZO methods can keep memory close to inference, but the additional function evaluations can increase computational and communication overhead, especially in distributed training or with larger batches.

Thus, existing approaches, including Muon, expose a fundamental trade-off between the memory footprint of first-order training and the optimization efficiency of standard gradients, prompting a natural question:

Can full-parameter fine-tuning retain first-order information and competitive wall-clock efficiency while making persistent optimizer-state memory nearly negligible?

To answer this, we propose **TACO**, a **T**ernary **A**bsolute-max **C**olumn-wise **O**ne-sparse optimizer. Using the same operator-norm steepest-descent viewpoint as Muon, *TACO is not just another memory-efficient optimizer*; TACO's conceptual centerpiece is an optimization geometry whose exact steepest-descent direction has extreme structural sparsity. Under a dimension-normalized $1 \to 1$ operator norm, the corresponding dual geometry makes the steepest-descent direction separable across columns, and each column selects the sign of its largest-magnitude gradient coordinate. Thus, an $m \times n$ update contains at most n nonzeros.

Although the TACO steepest-descent direction is one-sparse per column, minibatch gradients can make the selected coordinates unstable across training. A conventional dense gradient history can stabilize coordinate selection but erodes the memory advantage. TACO instead retains only a small, low-precision set of dynamic gradient heavy hitters per column. This sparse gradient history keeps persistent optimizer-state memory negligible even as model size grows. For example, TACO can fine-tune OPT-30B on a single 80 GB H100 GPU, with peak GPU memory of 62.5 GB on SST-2, 64.7 GB on RTE, and 68.1 GB on BoolQ; see Peak GPU memory decomposition in Figure 1. Adam does not fit this model on a single H100 80 GB GPU. Across our study, TACO enables full-parameter fine-tuning of models up to 32B parameters on a single 80 GB H100 and transfers across multiple model families and tasks.

Taken together, our main contributions are:

(*i*) **Geometry-induced sparse optimization.** We propose TACO, a new first-order optimizer for memory-efficient full-parameter LLM fine-tuning. TACO is derived from the exact steepest-descent solution under a dimension-normalized $1 \to 1$ operator norm, whose geometry naturally yields a ternary, column-wise Top-1 update. Thus, TACO's sparsity is intrinsic to its optimization geometry rather than imposed by post-hoc gradient sparsification; see Section 3.

(*ii*) **Convergence and fine-tuning theory.** We prove that vanilla TACO reaches an ε-stationary point in $\mathcal{O}(\varepsilon^{-2})$ iterations under standard smoothness assumptions. We further show that, in our theoretical setting, TACO shares the continuation geometry and limiting solution of an Adam proxy while differing from Muon, and satisfies standard μP spectral scaling under balanced winner counts. These results connect TACO to optimizer mismatch and the learning rate transfer problem; see Sections 3 and 4.

(*iii*) **TACO-specific memory-efficient design.** TACO does not require an accurate dense model of the gradient history, but only enough information to recover the dominant coordinate in each column. Exploiting this structure, we develop a practical implementation based on dynamic `FP8` sparse gradient histories and immediate gradient processing, reducing persistent state from $\mathcal{O}(mn)$ to $\mathcal{O}(n)$ while retaining first-order gradients and the column-wise Top-1 update; see Section 5.

(*iv*) **Empirical validation at scale.** We evaluate TACO across model scales, architectures, and downstream tasks. TACO achieves competitive task performance with substantially reduced training memory, enables full-parameter fine-tuning of OPT-30B and Qwen3-32B on a single 80 GB H100 GPU, and transfers across OPT, Qwen3, Pythia, Llama, and Mistral model families. Ablations further characterize the roles of the gradient history EMA decay factor, heavy hitter threshold, update sparsity, and optimizer step budget; see Section 6.

2 Related Work

Zeroth-order optimization has become a practical way to reduce the memory cost of LLM fine-tuning. These methods construct an update direction from the change in loss under a parameter perturbation. MeZO applies this idea by regenerating its perturbations in place using a seeding trick (Malladi et al., 2023). Sparse MeZO reduces the number of perturbed parameters, preserving sparsity (Liu et al., 2025). ZO-Muon projects the estimated gradient into a lower-dimensional subspace and applies polar orthogonalization to recover more useful structure from the noisy estimate (Lang et al., 2026). HiZOO instead incorporates diagonal Hessian information into zeroth-order optimization, using curvature-aware preconditioning to improve convergence (Zhao et al., 2025).

Memory-efficient first-order optimization methods retain backpropagation while reducing parameter-associated training state. Adafactor represents second moments using row and column factors (Shazeer & Stern, 2018), while CAME adds a confidence-guided correction to improve the stability of this representation (Luo et al., 2023). Alternatively, 8-bit Adam uses block-wise quantization to store optimizer statistics at lower precision (Dettmers et al., 2021). FlashOptim combines quantized optimizer states with a compact master-weight representation and provides FlashAdamW as a drop-in AdamW variant (Gonzalez Ortiz et al., 2026). GaLore projects gradients and optimizer states into a lower-dimensional space (Zhao et al., 2024), whereas APOLLO approximates adaptive gradient scaling through a random low-rank auxiliary space; its APOLLO-Mini variant reduces this space to rank one and applies tensor-wise scaling (Zhu et al., 2025).

Operator-norm and linear-minimization-oracle (LMO) optimization. Bernstein & Newhouse (2024) formulate matrix optimization as steepest-descent under operator norms and characterize the $1 \to p$ family through maximum column norms. Pethick et al. (2025) develop a stochastic LMO framework and instantiate it with layer-aware operator geometries in Scion.

Sign-based methods. In gradient compression, TernGrad maps gradient values to $\{-1, 0, +1\}$ to reduce the communication bottleneck in distributed training (Wen et al., 2017), while signSGD retains only the coordinate-wise gradient signs (Bernstein et al., 2018); see GRACE (Xu et al., 2021), a compressed communication framework that places these methods under one umbrella. SignMuon (Mishra et al., 2026) communicates the signs of locally constructed Muon directions.

3 TACO: TERNARY ABSOLUTE-MAX COLUMN-WISE ONE-SPARSE OPTIMIZER

Setup and notation. We consider the minimization problem

$$\min_{W \in \mathbb{R}^{m \times n}} F(W). \tag{1}$$

Let $G = \nabla F(W)$. We use d_j and g_j to denote the jth column of matrices D and G, respectively, and define $\langle G, D \rangle_F = \operatorname{tr}(G^\top D)$. For a vector $x \in \mathbb{R}^d$ and $r \in [1, \infty]$, we define the dimension-normalized r-norm as $\|x\|_{r,d} = d^{-1/r}\|x\|_r$, with the convention $d^{-1/\infty} = 1$. For a linear map $D : \mathbb{R}^n \to \mathbb{R}^m$, we define the induced operator norm, $\|D\|_{(p,n\to q,m)} = \sup_{x \neq 0} \|Dx\|_{q,m}/\|x\|_{p,n}$. Here, the domain $\mathbb{R}^n$ is equipped with $\|\cdot\|_{p,n}$ and the codomain $\mathbb{R}^m$ with $\|\cdot\|_{q,m}$. The indices in $(p, n \to q, m)$ record the norm orders and dimensions of the input and output spaces; here n and m determine normalization factors, not additional tensor indices. We use $\odot$ to denote the element-wise Hadamard product and, with a slight abuse of notation, let $\operatorname{sign}(\cdot)$ act element-wise on vectors and matrices. For $q \in [1, \infty]$, we write $\|A\|_{1\to q} = \max_{\|x\|_1 \leq 1} \|Ax\|_q$.

From Muon to TACO. A first-order Taylor approximation of (1) gives

$$F(W + \eta D) = F(W) + \eta \langle G, D \rangle_F + \mathcal{O}(\eta^2).$$

From this steepest-descent viewpoint, Muon solves the local minimization problem

$$\min_D \langle G, D \rangle_F, \quad \|D\|_{(2,n\to 2,m)} \leq 1, \tag{2}$$

where the geometry used to constrain D determines the steepest local descent direction. In this view, Muon uses the normalized $2 \to 2$ (spectral) operator norm; if G has the SVD decomposition $G = U\Sigma V^\top$, then D is proportional to $UV^\top$.

TACO makes a different geometric choice. Since the ℓ_1-norm encourages sparsity, we equip both spaces with normalized ℓ_1-norms, for which

$$\|D\|_{(1,n\to 1,m)} = \frac{n}{m}\|D\|_{1\to 1} = \frac{n}{m} \max_j \|d_j\|_1,$$

and we reformulate (2) as

$$\min_D \langle G, D \rangle_F, \quad \|D\|_{(1,n\to 1,m)} \leq 1. \tag{3}$$

The maximum-column identity is the $p = 1$ specialization of the general $1 \to p$ characterization; TACO's specific choice is to use this geometry as the basis of a column-wise one-sparse, low-state optimizer. Thus, $1 \to 1$ is not a global element-wise ℓ_1 constraint. It gives every column its own ℓ_1 budget. Its dual norm under the Frobenius pairing is

$$\|G\|_{(1,n\to 1,m)^*} = \frac{m}{n} \sum_j \|g_j\|_\infty.$$

Because the constraint separates across columns, each column spends its budget on an extreme point of an ℓ_1 ball, its largest-magnitude gradient coordinate. Let $i_j \in \arg\max_i |G_{ij}|, j \in [n]$, with any fixed rule for breaking ties, define the operator $\mathcal{S} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ as

$$\mathcal{S}(G)_{ij} := \begin{cases} \operatorname{sign}(G_{ij}), & i = i_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Therefore, an exact minimizer of (3) is

$$D^\star = -\frac{m}{n}\mathcal{S}(G).$$

TACO's geometry produces a **ternary, column-wise, and one-sparse** update; an $m \times n$ update has at most n nonzeros. The factor m/n is not an additional layerwise scaling hyperparameter; it follows directly from dimension normalization. This distinguishes TACO from Top-1 sparsification in conventional optimization (Sahu et al., 2021; Dutta et al., 2020). TACO's sparsity follows from the *exact closed-form solution of the local problem.*

Given W_t and $G_t = \nabla F(W_t)$, vanilla TACO computes $D_t = \frac{m}{n}\mathcal{S}(G_t)$ and updates the sequence $\{W_t\}_{t=0}^T$ as

$$W_{t+1} = W_t - \eta_t D_t. \tag{5}$$

The term vanilla distinguishes the history-free TACO iteration from the noise-robust practical implementation in Section 5.

Convergence of Vanilla TACO. We state the convergence guarantee of vanilla TACO using the same primal-dual norm pair that defines the update.

Assumption 1. *(Lower boundedness and smoothness) The objective function is lower bounded and L-smooth. That is, $F_\star := \inf_W F(W) > -\infty$, and for all $W_1, W_2 \in \mathbb{R}^{m\times n}$, the following holds: $\|\nabla F(W_1) - \nabla F(W_2)\|_{(1,n\to1,m)^*} \le L\|W_1 - W_2\|_{(1,n\to1,m)}$.*

Theorem 1. *(Nonconvex convergence of vanilla TACO) Let Assumption 1 hold and define $\Delta_F := F(W_0) - F_\star$. The iterates $\{W_t\}_{t=0}^T$ in (5) with step sizes $\eta_t > 0$ satisfy*

$$\min_{0\le t<T} \|\nabla F(W_t)\|_{(1,n\to1,m)^*} \le \frac{\Delta_F + \frac{L}{2}\sum_{t=0}^{T-1}\eta_t^2}{\sum_{t=0}^{T-1}\eta_t}.$$

In particular, with $\eta = \sqrt{\frac{2\Delta_F}{LT}}$, we have $\min_{0\le t<T}\|\nabla F(W_t)\|_{(1,n\to1,m)^*} \le \sqrt{\frac{2L\Delta_F}{T}}$.

We call W an ε-stationary point in the TACO geometry if $\|\nabla F(W)\|_{(1,n\to1,m)^*} \le \varepsilon$. Theorem 1 shows that vanilla TACO reaches an ε-stationary point in $\mathcal{O}(\varepsilon^{-2})$ iterations.

4 Continuation Geometry, Implicit Bias, and μP-scaling of TACO

Recent work shows that Adam and Muon can exhibit different fine-tuning behavior because their update geometries induce different implicit biases (Qu et al., 2026). We study where TACO lies relative to these geometries in a simplified fixed-input setting, and then analyze how the spectral norm of its update scales with matrix dimensions.

Let $z \in \mathbb{R}^n, z \neq 0$, denote a fixed input. Ties in TACO's column-wise selections are resolved using a fixed deterministic rule. For $W_0 \in \mathbb{R}^{m\times n}$ and a vector $w \in \mathbb{R}^n$, let the continuation set generated by w at W_0 be represented by $\mathcal{C}(W_0, w) := \{W_0 + aw^\top : a \in \mathbb{R}^m\}$.

We include the Adam and Muon proxies in (10) and (11) respectively; both can be found in Appendix A. To make the optimizer comparison precise, we consider a setting in which the objective depends on a weight matrix only through its action on a fixed input and ask how different update geometries move through parameter space while producing the same target output.

Assumption 2. *(Coherent objective) The objective depends on W through the layer output, i.e., $F(W) = \phi(Wz)$ for a differentiable function $\phi : \mathbb{R}^m \to \mathbb{R}$.*

Assumption 3. *(Common target output) Starting from the same W_0, all compared methods converge to limits $W^\star$ satisfying $W^\star z = u^\star$ for the same optimal output $u^\star$.*

With these conditions in place, we now ask how each optimizer moves through parameter space. We first show that, despite very different update rules, TACO and Adam follow the same continuation geometry and converge to the same parameter-space solution. We then contrast this behavior with Muon, whose spectral geometry induces a different continuation path.

Theorem 2. *(Adam and TACO continuation equivalence) Let Assumptions 2 and 3 hold, and let the Adam Proxy follow (10). Then both the TACO and Adam sequences lie in $\mathcal{C}(W_0, \operatorname{sign}(z))$ for all $t \ge 0$, and $W^\star_{\mathrm{TACO}} = W^\star_{\mathrm{Adam}} = W_0 + \frac{(u^\star - W_0 z)\operatorname{sign}(z)^\top}{\|z\|_1}$.*

Theorem 3. *(Muon continuation separation) Let Assumptions 2 and 3 hold and let the Muon proxy follow (11). Further let $r = u^\star - W_0 z$. Then Muon's iterates lie in $\mathcal{C}(W_0, z)$ and $W^\star_{\mathrm{Muon}} = W_0 + \frac{rz^\top}{\|z\|_2^2}, W^\star_{\mathrm{Muon}} - W^\star_{\mathrm{TACO}} = r\left(\frac{z}{\|z\|_2^2} - \frac{\operatorname{sign}(z)}{\|z\|_1}\right)^\top$.*

Theorems 2 and 3 demonstrate that, under this setting, TACO behaves more like Adam than Muon when fine-tuning a model, even though TACO was motivated by the same geometric framework as Muon. In our setting, TACO and Adam are forced to move through the same family of parameter updates and arrive at the exact same solution, whereas Muon generally moves in a different direction and ends at a different solution. This matters because models are usually pretrained with AdamW, so preserving Adam-like behavior during fine-tuning may help avoid the optimizer mismatch problem that appears when switching to Muon.

Theoretical μP. The μP problem seeks update scalings that preserve well-behaved optimization dynamics as model width increases, allowing the learning rate to transfer across model scales rather than requiring repeated hyperparameter tuning; Muon is one solution to the framework (Bernstein, 2025). Since TACO is derived from the same steepest-descent perspective under a different operator geometry, we ask whether its update preserves the spectral scaling required for μP transfer.

Assumption 4. *(Balanced TACO Winners) For $G \in \mathbb{R}^{m\times n}$, let c_i denote the number of columns whose TACO winner lies in row i. We assume $\max_i c_i = \Theta(n/m)$ for all iterates W_t.*

Theorem 4. *(TACO μP-scaling) Let Assumption 4 hold. Then the TACO direction $D = \frac{m}{n}\mathcal{S}(G)$ satisfies $\|D\|_2 = \frac{m}{n}\sqrt{\max_i c_i}$, for every G. Hence D satisfies the standard μP spectral scaling $\|D\|_2 = \Theta\left(\sqrt{\frac{m}{n}}\right)$.*

Theorem 4 shows that TACO satisfies the standard μP spectral scaling law when the largest-magnitude coordinates selected across columns remain reasonably balanced across rows, rather than becoming disproportionately concentrated in only a few rows. Hence, the theorem provides a theoretical basis for learning rate transfer across model widths. We complement this result with empirical μP experiments in Appendix B, where the same learning rate remains optimal across model scales.

Together with the continuation analysis above, these theoretical developments position TACO as an unusually favorable combination: it inherits the reduced hyperparameter sensitivity associated with μP scaling, follows the same continuation geometry as AdamW under our theoretical setting, and retains these properties while requiring almost no persistent optimizer state.

5 Practical Design of TACO

Although vanilla TACO updates are sparse by design, applying $\mathcal{S}$ directly to a single minibatch gradient makes the selected coordinates sensitive to minibatch noise. A conventional dense gradient history can stabilize coordinate selection but also reintroduce the dense mn-entry state TACO was meant to avoid. Exploiting TACO's special geometric update structure, our implementation resolves this tension with a sparse gradient history, low-precision state, and immediate gradient processing. This motivates the following design principle:

> TACO does not need an accurate model of the gradient history. TACO only needs enough information to recover its dominant coordinate in each column.

Gradient history and dynamic heavy hitters. A dense exponential moving average (EMA) is $\tilde{M}_t = \mu\tilde{M}_{t-1} + (1-\mu)G_t, \mu = 0.95$. However, the next TACO direction only needs a reliable winner: the entry with the largest absolute value in each column, rather than a dense reconstruction of $\tilde{M}_t$. Let $\mathcal{H}_k(X)$ retain the k largest-magnitude entries of every column of X and set all remaining entries to zero. Practical TACO maintains the following update rule:

$$M_t = \mathcal{H}_k(\mu M_{t-1} + (1-\mu)G_t)\,, \qquad \mu = 0.95, \quad k = 16,$$

where the unretained entries of the sparse M_{t-1} are treated as zero. The update remains $D_t = \frac{m}{n}\mathcal{S}(M_t)$. In our experiments, we set $k = 16$ to control how many entries are retained in each column for future winner selection; it does *not* change the column-wise Top-1 TACO update. For additional discussion of sparse gradient history approximation, see Theorem 5 and ablations in Appendix B. The retained history values are stored in `FP8 E4M3` format, which uses four exponent bits and three mantissa bits (Micikevicius et al., 2022), together with `int32` for row indices. Winner selection depends primarily on relative magnitude and sign and therefore requires less numerical fidelity than an adaptive moment used to rescale every coordinate.

Persistent state memory. Each retained entry uses one byte for its `E4M3` value and four bytes for its `int32` row index. Thus, for an $m \times n$ matrix, TACO stores $5kn$ bytes, compared with mnb_{dense} bytes for one dense state matrix whose entries use b_{dense} bytes. When $m \le 2^{15}$, `int16` indices suffice, reducing the history storage to $3kn$ bytes. But some model matrices have more rows than signed `int16` can index, so we use `int32`. With `int32`, the relative history cost is $5k/(mb_{\text{dense}})$, which decreases as the matrix height m grows and is small when $k \ll m$. Thus, TACO reduces persistent state memory complexity from $\mathcal{O}(mn)$ to $\mathcal{O}(kn)$.

Backward pass memory management. TACO is applied to matrix-valued parameters, while scalar and vector parameters use an auxiliary AdamW optimizer. PyTorch `autograd hooks` (Paszke

Algorithm 1 Practical implementation of TACO with gradient history and dynamic heavy hitters

Require: $W_0 \in \mathbb{R}^{m\times n}$; learning rate schedule $(\eta_t)_{t\geq 0}$; gradient history EMA decay factor $\mu = 0.95$; heavy hitter threshold $k = 16$; optimizer step budget T.
Ensure: Final parameters W_T.
1: Initialize an empty sparse gradient history M_{-1}
2: **for** $t = 0, \ldots, T-1$ **do**
3: Compute $G_t \leftarrow \nabla F(W_t)$ ▷ `Backpropagation`
4: $G_t \leftarrow \mu M_{t-1} + (1-\mu)G_t$ ▷ `In-place history update`
5: $M_t \leftarrow \mathcal{H}_k(G_t)$ ▷ `Dynamic heavy hitters`
6: Store the retained values of M_t in `E4M3` and their row indices in `int32`
7: $W_{t+1} \leftarrow W_t - \eta_t \frac{m}{n}\mathcal{S}(M_t)$ ▷ $\mathcal{S}$ `is defined in (4)`
8: Release the dense G_t
9: **end for**
10: **return** W_T

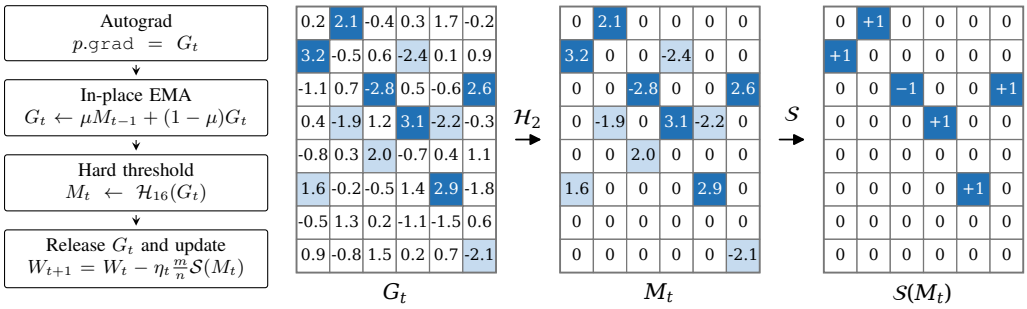

Figure 2: **Illustrative workflow of the memory-efficient practical implementation of TACO.** We use $\mathcal{H}_2$ for illustration; in our experiments, we use $\mathcal{H}_{16}$. In this example, we highlight the largest- and second-largest-magnitude elements for $\mathcal{H}_2$ in each column. Here, G_t is the dense gradient via `autograd`, M_t is obtained via $\mathcal{H}_2(G_t)$ by retaining the largest- and second-largest-magnitude elements in each column, and finally, the operator $\mathcal{S}$ is used, which keeps the sign of the largest-magnitude element in each column. Only the sparse gradient history M_t persists across iterations.

et al., 2019) make each matrix gradient G_t available as soon as it is computed. TACO updates G_t in-place using the sparse gradient history, extracts $\mathcal{H}_k(G_t)$ as the new history, applies the parameter update, and releases G_t. We set gradient accumulation to 1, so dense gradients do not persist across microbatches. Gradient checkpointing reduces activation memory. Together, these techniques bring peak training memory close to inference memory. Peak training memory includes model parameters, sparse optimizer state, checkpointed activations, and the current gradient. `Autograd hooks` and checkpointing are standard systems techniques rather than contributions of this work.

6 Experiments

Implementation details and models used. We evaluate TACO for full-parameter fine-tuning across five model families: OPT, Qwen3, Pythia, Llama, and Mistral, covering model sizes from 1.3B to 32B parameters. Our experiments include SST-2, RTE, BoolQ, MultiRC, SQuAD, DROP, COPA, and CB, spanning classification, multiple-choice reasoning, and question answering. We first compare TACO with a range of optimizers on OPT-13B, then study its scaling behavior across the OPT and Qwen3 model families and its transfer across different model architectures. Finally, we summarize several ablations characterizing TACO's optimizer step budget, gradient history EMA decay factor, heavy hitter threshold, and update sparsity. Unless otherwise specified, TACO uses a shared hyperparameter configuration across the non-ablation experiments; see complete experimental configurations, dataset construction, prompt formats, hyperparameters, and results in Appendix B.

Memory, trained model quality, and runtime on OPT-13B. Figure 3 compares TACO with memory-efficient and conventional first-order optimizers across eight downstream tasks. TACO consistently requires substantially less peak GPU memory while maintaining competitive task performance. The SST-2 efficiency comparison in Figure 4 isolates this advantage: beyond the 25.7 GB required for model weights, TACO uses only 1.8 GB of additional training memory, including

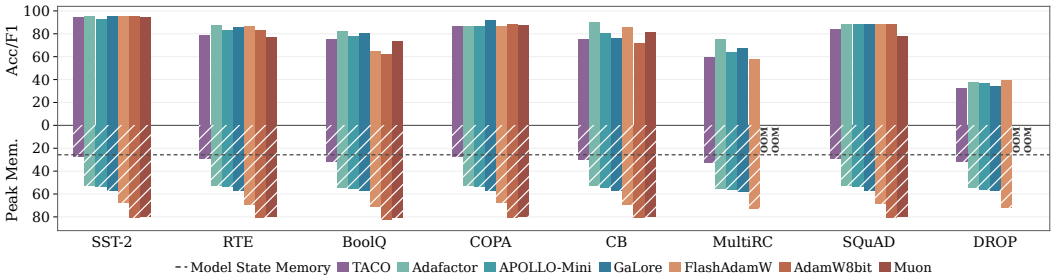

Figure 3: **OPT-13B performance and peak-memory comparison across downstream tasks.** Upper bars report test accuracy or F1; lower mirrored bars report peak GPU memory. **OOM** denotes configurations that do not fit within the available GPU memory. Across all tasks, TACO consistently requires the least GPU memory while maintaining competitive performance; see Table 3 in Appendix B for full results.

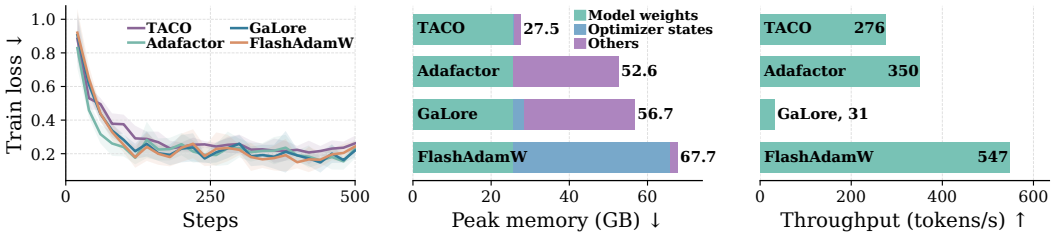

Figure 4: **Efficiency comparison** in terms of optimization trajectory, peak memory decomposition, and training throughput for fine-tuning OPT-13B on SST-2. For the Train loss vs. Steps plot, we show only the first 500 steps, as all optimizers plateaued; the shaded region shows ± sample standard deviation across five training seeds; see Table 4 in Appendix B for full results.

just 0.16 GB of persistent optimizer state. In comparison, Adafactor, GaLore, and FlashAdamW require 26.9, 31.0, and 42.0 GB of additional memory, respectively, resulting in peak memory of 52.6, 56.7, and 67.7 GB versus 27.5 GB for TACO. Although Adafactor has the smallest persistent optimizer state, its substantially larger transient memory results in much higher peak GPU memory than TACO. TACO reaches 94.2% SST-2 accuracy at 276 tokens/second, compared with 95.5% at 350 tokens/second for Adafactor, 95.7% at 31 tokens/second for GaLore, and 95.5% at 547 tokens/second for FlashAdamW. TACO therefore occupies the low-memory end of the accuracy–memory trade-off while retaining competitive performance and practical throughput.

Scaling across model families. Figure 5 shows that TACO remains trainable within the memory limit of a single 80 GB H100 as OPT scales from 1.3B to 30B parameters and Qwen3 scales from 8B to 32B parameters. Within OPT, SST-2 accuracy remains broadly stable, RTE generally improves with scale, and BoolQ is non-monotonic. Qwen3 similarly maintains strong performance across model sizes. Peak memory scales with model size and stays within the capacity of a single H100 80 GB GPU, even for OPT-30B and Qwen3-32B. Importantly, these experiments use the same TACO hyperparameter configuration across model sizes rather than model-specific tuning.

Transfer across architectures. TACO is agnostic to model architecture. It achieves strong performance on Llama-3.1-8B and Qwen3-32B across all three tasks, while Pythia-12B and Mistral-24B exhibit larger task-dependent variation, particularly on RTE and BoolQ. Peak GPU memory remains close to the model footprint across architectures, consistent with TACO's small persistent optimizer state. Figure 6 reports architecture transfer results and examines learning rate transfer across OPT model scales, where all four model sizes attain their best validation accuracy at 3×10^{-5}. The broader sweep in Appendix B shows the same optimum across all evaluated model-dataset configurations, further supporting the use of a shared TACO configuration across model scales.

Gradient history and heavy hitter threshold. Practical TACO uses an EMA gradient history with sparse heavy-hitter. Comparing dense gradient history with $k \in \{64, 32, 16, 8, 4, 2, 1\}$, we find moderate sparse histories mostly retain performance comparable to dense gradient history while substantially reducing persistent optimizer state. In particular, $k = 16$ uses only 38.3 MB of optimizer state, a $34.4\times$ reduction from dense gradient history, while achieving the strongest BoolQ and RTE performance in the sweep. Hence, we use $k = 16$ as the default heavy-hitter threshold. Across

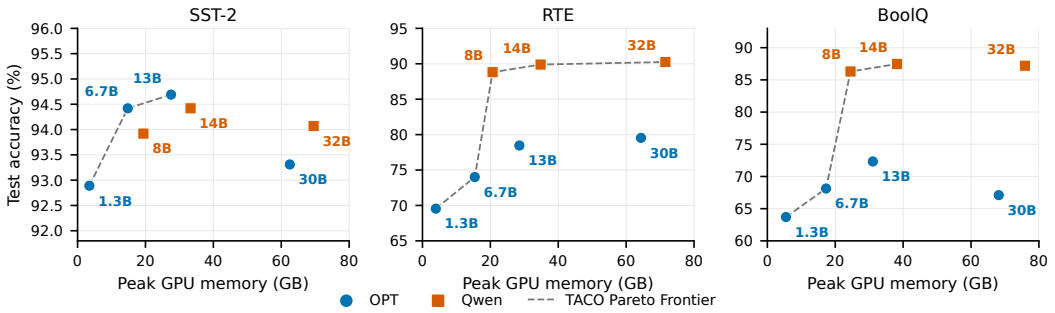

Figure 5: **Scaling behavior of TACO across the OPT and Qwen3 model families.** We plot the test performance against peak GPU memory as model size increases; see Table 5 in Appendix B for full results.

Model	SST-2		RTE		BoolQ	
	Acc.	Mem. (GB)	Acc.	Mem. (GB)	Acc.	Mem. (GB)
Pythia-12B	92.6	25.2	65.1	26.2	66.7	28.6
Llama-3.1-8B	94.2	18.7	87.0	19.8	86.6	22.9
Qwen3-32B	93.1	69.6	90.4	71.6	88.0	75.9
Mistral-24B	87.8	50.5	54.4	52.0	69.5	55.1

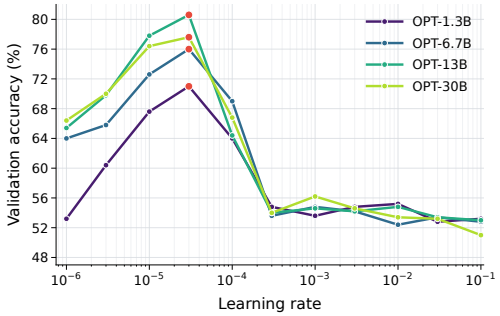

Figure 6: **Transfer across model architectures and learning-rate stability across model scales.** *Left:* TACO performance and peak GPU memory across Pythia-12B, Llama-3.1-8B, Qwen3-32B, and Mistral-24B on SST-2, RTE, and BoolQ, showing consistent memory efficiency with task-dependent performance. *Right:* Learning rate sweep on RTE for OPT-1.3B, OPT-6.7B, OPT-13B, and OPT-30B. All four models attain their best validation accuracy at 3×10^{-5}, supporting learning rate transfer across model scales.

eight tasks, $\mu = 0.95$ improves six, including classification and multiple-choice tasks, with the largest gains on MultiRC (+8.9 F1A) and CB (+7.1%); SQuAD and DROP are the two exceptions. We therefore retain $\mu = 0.95$ as default; see Tables 7 and 8 in Appendix B for full results.

Optimizer step budget. TACO continues to improve beyond the default 2K-step budget. Increasing the budget from 2K to 16K steps improves BoolQ and RTE by +6.5% and +4.3%, respectively, and SQuAD by +2.1 F1. These results show that the default 2K-step budget does not represent TACO's performance ceiling; see Table 6 in Appendix B for full results.

Update sparsity. We relax TACO's theoretically derived column-wise Top-1 update and evaluate denser Top-k variants. Top-1 achieves the strongest final performance across all three evaluated tasks, while dense updates substantially degrade performance, particularly at Top-8 and beyond. These results support retaining Top-1 as the standard TACO update, consistent with the exact steepest-descent solution under the $1 \to 1$ geometry; see Table 9 in Appendix B for full results.

7 Conclusion

We introduced the **T**ernary **A**bsolute-max **C**olumn-wise **O**ne-sparse (TACO) optimizer for memory-efficient full-parameter LLM fine-tuning. TACO is derived from the exact minimizer of a local optimization problem under a dimension-normalized $1 \to 1$ operator norm, yielding a ternary, column-wise one-sparse update. Exploiting this structure, practical TACO maintains a sparse, low-precision gradient history for matrix-valued parameters, making persistent optimizer-state memory nearly negligible. We establish a nonconvex convergence guarantee for vanilla TACO and, in a simplified fixed-input setting, show that it shares the continuation geometry and limiting solution of an Adam proxy while generally differing from Muon. We further establish μP spectral scaling under balanced winner counts and give a sufficient condition under which sparse history preserves the dense-history TACO update. Across model scales, architectures, and downstream tasks, TACO maintains competitive performance with a small memory footprint, enabling full-parameter fine-tuning of models up to 32B parameters on a single 80 GB H100 GPU. More broadly, our results suggest that optimization geometry can provide a useful framework for designing memory-efficient first-order optimizers.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, as research and writing assistants. ChatGPT was used to improve the clarity and organization of the manuscript, provide feedback on mathematical exposition and experimental methodology, assist with literature search, and help inspect and summarize experimental results. Codex was used to assist with code refinement, debugging, and preparing auxiliary experiment scripts, including Slurm job scripts. The core research ideas, mathematical methodology, experimental design, and initial implementation were developed by the authors. All AI-assisted text, experimental interpretations, and code were reviewed by the authors, and code changes were tested against the corresponding experiments. The authors take full responsibility for the final content and results of this work.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproduction of our theoretical and empirical results. The TACO algorithm and its practical implementation are described in Sections 3 to 5, with complete proofs and additional theoretical details in Appendix A. Appendix B reports the datasets, prompt construction, model configurations, optimizer-specific hyperparameters, hardware setup, learning-rate selection protocol, and complete experimental results. Anonymous source code for TACO and all experiments, including the training and evaluation scripts needed to reproduce the reported results, is available at `https://anonymous.4open.science/r/iclr27-TACO-923D/`.

REFERENCES

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Text Analysis Conference*, 2009.

El Houcine Bergou, Eduard Gorbunov, and Peter Richtárik. Stochastic Three Points Method for Unconstrained Smooth Minimization. *SIAM Journal on Optimization*, 30(4):2726–2749, 2020.

Jeremy Bernstein. Deriving muon, 2025. URL `https://jeremybernste.in/writing/deriving-muon`.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291*, 2018.

Adel Bibi, El Houcine Bergou, Ozan Sener, Bernard Ghanem, and Peter Richtárik. A stochastic derivative-free optimization method with importance sampling: Theory and Learning to Control. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 3275–3282, 2020.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430, 2023.

Soumia Boucherouite, Grigory Malinovsky, Peter Richtárik, and El Houcine Bergou. Minibatch Stochastic Three Points Method for Unconstrained Smooth Minimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20344–20352, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

HanQin Cai, Yuchen Lou, Daniel Mckenzie, and Wotao Yin. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 1193–1203, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936, 2019.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, 2005.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *Sinn und Bedeutung*, volume 23, pp. 107–124, 2019.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 2368–2378, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Agam Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Akhoundi, Akbar Alibek, Tanel Alumae, and Anirudh *et al.* Anichini. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3817–3824, 2020.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

Jose Javier Gonzalez Ortiz, Abhay Gupta, Christopher Rinard, and Davis Blalock. FlashOptim: Optimizers for memory-efficient training. In *International Conference on Machine Learning*, 2026.

Eduard Gorbunov, Adel Bibi, Ozan Sener, El Houcine Bergou, and Peter Richtarik. A stochastic derivative free optimization method with momentum. In *International Conference on Learning Representations*, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. GitHub repository, 2024.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 252–262, 2018.

Bumsu Kim, Daniel McKenzie, HanQin Cai, and Wotao Yin. Curvature-aware derivative-free optimization. *Journal of Scientific Computing*, 103(43):1–28, 2025.

Yicheng Lang, Changsheng Wang, Yihua Zhang, Mingyi Hong, Zheng Zhang, Wotao Yin, and Sijia Liu. Powering up zeroth-order training via subspace gradient orthogonalization, 2026.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 21139–21169, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Yang Luo, Xiaozhe Ren, Zangwei Zheng, Zhuo Jiang, Xin Jiang, and Yang You. Came: Confidence-guided adaptive memory efficient optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 4442–4453, 2023.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Neel Mishra, Kushagara Trivedi, and Pawan Kumar. Signmuon: Communication-efficient distributed muon optimization. *arXiv preprint arXiv:2605.16311*, 2026.

Mistral AI Team. Mistral-small-24b-base-2501 foundation model. `https://huggingface.co/mistralai/Mistral-Small-24B-Base-2501`, 2025. Released January 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49069–49104, 2025.

Xingyu Qu, Peigeng Huang, and Samuel Horvath. Can muon fine-tune adam-pretrained models? *arXiv preprint arXiv:2605.10468*, 2026.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. 2011.

Atal Sahu, Aritra Dutta, Ahmed M Abdelmoniem, Trambak Banerjee, Marco Canini, and Panos Kalnis. Rethinking gradient sparsification as total error minimization. *Advances in Neural Information Processing Systems*, 34:8133–8146, 2021.

Ishaan Shah, Anthony M. Polloreno, Karl Stratos, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604, 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, volume 32, 2019.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.

Hang Xu, Chen-Yu Ho, Ahmed M Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. Grace: A compressed communication framework for distributed machine learning. In *2021 IEEE 41st international conference on distributed computing systems*, pp. 561–572, 2021.

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. *arXiv preprint arXiv:2505.09388*, 2025.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of LLMs in random subspaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143, 2024.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 43496–43520, 2025.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z. Pan, Zhangyang Wang, and Jinwon Lee. APOLLO: SGD-like memory, AdamW-level performance. In *Proceedings of Machine Learning and Systems*, 2025.

Supplementary Material for TACO: TERNARY ABSOLUTE-MAX COLUMN-WISE ONE-SPARSE OPTIMIZER FOR LLM FINE-TUNING

CONTENTS

1 Introduction **1**

2 Related Work **3**

3 TACO: Ternary Absolute-max Column-wise One-sparse Optimizer **4**

4 Continuation Geometry, Implicit Bias, and μP-scaling of TACO **5**

5 Practical Design of TACO **6**

6 Experiments **7**

7 Conclusion **9**

A Theoretical Analysis and Proofs **14**

A.1 Vanilla TACO Convergence 15

A.1.1 Multilayer analysis 15

A.2 Continuation Geometry, Implicit Bias, and μP-scaling Analysis 16

A.3 Gradient History and Heavy Hitter Analysis 19

B Experiment Setup and Additional Experiment Results **19**

B.1 Optimizer comparison 20

B.2 Model Scaling 22

B.3 Ablation studies 22

B.4 Architecture Transfer 24

Organization of the Appendix. We organize the Appendix as follows. Appendix A contains the proofs of the theorems in Sections 3 and 4. In particular, it contains the proofs of Theorem 1 to Theorem 4. Appendix B describes the experimental setup, including the models and datasets, implementation details, hyperparameter settings, and hardware configuration. Appendix B also provides additional experimental results, including the complete optimizer comparison, model scaling results, ablation studies, and architecture transfer results.

A THEORETICAL ANALYSIS AND PROOFS

This section provides proofs and additional theoretical context for the results in Sections 3 and 4. We first establish the convergence guarantee for vanilla TACO, then examine its continuation geometry and μP scaling, and conclude with a condition under which sparse gradient histories preserve the dense-history TACO update.

A.1 VANILLA TACO CONVERGENCE

We begin by proving the convergence guarantee in Theorem 1 for the history-free iteration. The proof uses $\langle G_t, D_t \rangle_F = \|G_t\|_{(1,n\to 1,m)^*}$ to express the first-order decrease exactly, while smoothness contributes the step-size penalty $L\eta_t^2/2$.

Theorem 1. *(Nonconvex convergence of vanilla TACO) Let Assumption 1 hold and define $\Delta_F = F(W_0) - F_\star$. The iterates $\{W_t\}_{t=0}^T$ in (5) with step size $\eta_t > 0$ follow*

$$\min_{0\le t<T} \|\nabla F(W_t)\|_{(1,n\to 1,m)^*} \le \frac{\Delta_F + \frac{L}{2}\sum_{t=0}^{T-1}\eta_t^2}{\sum_{t=0}^{T-1}\eta_t}.$$

In particular, with $\eta = \sqrt{\frac{2\Delta_F}{LT}}$, we have $\min_{0\le t<T} \|\nabla F(W_t)\|_{(1,n\to 1,m)^*} \le \sqrt{\frac{2L\Delta_F}{T}}$.

Proof. By Assumption 1, using the L-smoothness of F, for any update direction D, we have

$$F(W+D) \le F(W) + \langle \nabla F(W), D\rangle_F + \frac{L}{2}\|D\|_{(1,n\to 1,m)}^2. \tag{6}$$

Since $\nabla F(W) = G$, with $W_{t+1} = W_t - \eta_t D_t$, we have $\|D_t\|_{(1,n\to 1,m)} \le 1$, $\langle G_t, D_t\rangle_F = \|G_t\|_{(1,n\to 1,m)^*}$, and (6) reduces to

$$F(W_{t+1}) \le F(W_t) - \eta_t \|G_t\|_{(1,n\to 1,m)^*} + \frac{L}{2}\eta_t^2. \tag{7}$$

Summing (7) with t varying from 0 to $T-1$ gives

$$F(W_T) - F(W_0) \le -\sum_{t=0}^{T-1} \eta_t \|G_t\|_{(1,n\to 1,m)^*} + \frac{L}{2}\sum_{t=0}^{T-1}\eta_t^2. \tag{8}$$

Note that $F \ge F_\star$. Reordering the inequality, we obtain

$$\left(\min_{t<T} \|G_t\|_{(1,n\to 1,m)^*}\right)\sum_{t=0}^{T-1}\eta_t \le \Delta F + \frac{L}{2}\sum_{t=0}^{T-1}\eta_t^2, \tag{9}$$

which proves the result. □

The resulting bound balances the initial objective gap Δ_F against the accumulated step-size penalty. Choosing $\eta = \sqrt{2\Delta_F/(LT)}$ gives the $T^{-1/2}$ stationarity bound.

A.1.1 MULTILAYER ANALYSIS

For neural network training, consider matrix parameters

$$W = \left(W^{(1)}, \cdots, W^{(L)}\right), W^{(\ell)} \in \mathbb{R}^{m_\ell \times n_\ell},$$

and the objective $\mathcal{L} : \mathbb{R}^{m_1\times n_1} \times \cdots \times \mathbb{R}^{m_L \times n_L} \to \mathbb{R}$. For each layer, define $N_\ell(D^{(\ell)}) := \|D^{(\ell)}\|_{(1,n_\ell\to 1,m_\ell)}$. On the product space $\mathbb{R}^{m_1\times n_1} \times \cdots \times \mathbb{R}^{m_L\times n_L}$, define $\mathcal{N}(D) = \max_\ell \|D^{(\ell)}\|_{(1,n_\ell\to 1,m_\ell)}$. The dual norm of $\mathcal{N}$ is given by

$$\mathcal{N}^*(G) = \sum_{\ell=1}^L \|G^{(\ell)}\|_{(1,n_\ell\to 1,m_\ell)^*} = \sum_{\ell=1}^L \frac{m_\ell}{n_\ell}\sum_j \|g_j^{(\ell)}\|_\infty.$$

The local minimization problem under this multilayer setting is

$$\min_D \sum_{\ell=1}^L \left\langle G^{(\ell)}, D^{(\ell)}\right\rangle_F, \quad \text{s.t. } \mathcal{N}(D) \le 1.$$

The exact minimizer is obtained simultaneously in every layer,

$$D^{(\ell)} = \frac{m_\ell}{n_\ell} S(G^{(\ell)}), \quad \ell = 1, \cdots, L.$$

Thus, the parameter update is

$$W_{t+1}^{(\ell)} = W_t^{(\ell)} - \eta_t D_t^{(\ell)}, \quad \ell = 1, \cdots, L.$$

Note that every layer receives its geometry scaling factor $\frac{m_\ell}{n_\ell}$, and the optimization learning rate η_t remains a single shared scalar across all layers. This distinction is important; it suggests that the learning rate is an optimization hyperparameter that is invariant to the parameter shape.

A.2 Continuation Geometry, Implicit Bias, and μP-scaling Analysis

Our implicit-bias analysis extends the implicit-bias study of Qu et al. (2026) beyond their least-squares setting. Rather than restricting the objective to a squared loss, we consider the more general class $F(W) = \phi(Wz)$ for a differentiable function ϕ, and characterize the continuation geometry and limiting solutions induced by TACO, the Adam proxy, and the Muon proxy under a common target output. The least-squares problem studied in Qu et al. (2026) is recovered as a specific instance of this framework; we return to this case after the general results and show how our solutions reduce to their previously derived SignGD and Muon endpoints.

We consider $F(W) = \phi(Wz)$ and ask which weights are selected among matrices producing the same target output $u^\star$. For the implicit-bias analysis, we use the following optimizer proxies:

The **Adam proxy** is initialized with $M_{-1} = V_{-1} = 0$ and follows

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)G_t \quad V_t = \beta_2 V_{t-1} + (1-\beta_2)G_t \odot G_t \quad \hat{M}_t = \frac{M_t}{1-\beta_1^{t+1}}$$
$$\hat{V}_t = \frac{V_t}{1-\beta_2^{t+1}} \qquad W_{t+1} = W_t - \eta_t \frac{\hat{M}_t}{\sqrt{\hat{V}_t}} \tag{10}$$

The **Muon proxy** is initialized with $B_{-1} = 0$ and follows

$$B_t = \beta B_{t-1} + G_t, \quad W_{t+1} = W_t - \eta_t \mathcal{P}(B_t), \quad \mathcal{P}(B_t) = UV^\top, \quad B_t = U\Sigma V^\top. \tag{11}$$

Here, $G_t = \nabla F(W_t)$.

We note that both proxies use momentum; however, their resulting theoretical developments are invariant to momentum. We use momentum for a more realistic case.

The proofs of Theorems 2 and 3 begin with $G_t = g_t z^\top$, where $g_t = \nabla\phi(W_t z)$. As a result, the gradient columns are scaled copies of the same vector. We track this structure through the optimizer histories where TACO's and Adam proxy's winners yield the factor $\operatorname{sign}(z)^\top$, whereas the Muon proxy obtains $z^\top$.

Theorem 2. *(Adam and TACO continuation equivalence) Let Assumptions 2 and 3 hold, and let the Adam Proxy follow (10). Then both the TACO and Adam sequences lie in $\mathcal{C}(W_0, \operatorname{sign}(z))$ for all $t \geq 0$, and*

$$W^\star_{\mathrm{TACO}} = W^\star_{\mathrm{Adam}} = W_0 + \frac{(u^\star - W_0 z)\operatorname{sign}(z)^\top}{\|z\|_1}.$$

Proof. Assumption 2 and the chain rule give $G_t = \nabla F(W_t) = g_t z^\top$ with $g_t = \nabla\phi(W_t z)$.

For TACO, $M_t = p_t z^\top$ for some $p_t \in \mathbb{R}^m$. This holds at $t = 0$ since $M_{-1} = 0$. If $M_{t-1} = p_{t-1} z^\top$, then $\mu M_{t-1} + (1-\mu)G_t = \tilde{p}_t z^\top$ with $\tilde{p}_t = \mu p_{t-1} + (1-\mu)g_t$.

Column j of $\tilde{p}_t z^\top$ equals $z_j \tilde{p}_t$, so its entry magnitudes are $|z_j||\tilde{p}_{t,i}|$ and their ordering does not depend on j.

The nonzero columns have the same magnitude ordering, with equal-magnitude entries ordered by row index. Hence, H_k retains the same row set $\mathcal{I}_t$ in each nonzero column, while zero columns remain zero, giving $M_t = p_t z^\top$.

The same ordering argument applied to S selects a single row $i_t^\star$ in every column with $z_j \neq 0$ and returns a zero column whenever $z_j = 0$, so $S(M_t) = \operatorname{sign}(p_{t,i_t^\star}) e_{i_t^\star} \operatorname{sign}(z)^\top$ and $D_t = \frac{m}{n} S(M_t)$ has the form $a \operatorname{sign}(z)^\top$.

For the Adam proxy, $M_t = p_t z^\top$ and $V_t = q_t (z \odot z)^\top$ with $q_t \geq 0$ entrywise. Both hold at $t = 0$ and propagate, since $G_t \odot G_t = (g_t \odot g_t)(z \odot z)^\top$. Bias correction preserves the two forms, so the (i,j) entry of $\hat{M}_t / \sqrt{\hat{V}_t}$ equals $c_t (p_{t,i}/\sqrt{q_{t,i}}) \operatorname{sign}(z_j)$ with $c_t = \sqrt{1-\beta_2^{t+1}}/(1-\beta_1^{t+1})$, giving an update of the form $a \operatorname{sign}(z)^\top$ as well.

Hence, both sequences lie in $\mathcal{C}(W_0, \operatorname{sign}(z))$.

For $W = W_0 + a\operatorname{sign}(z)^\top$ we have $\operatorname{sign}(z)^\top z = \|z\|_1$, hence $Wz = W_0 z + a\|z\|_1$. Imposing $Wz = u^\star$ and using $\|z\|_1 > 0$ determines $a = (u^\star - W_0 z)/\|z\|_1$. □

Theorem 3. *(Muon continuation separation) Let Assumptions 2 and 3 hold and let the Muon proxy follow (11). Further let $r = u^\star - W_0 z$. Then Muon's iterates lie in $\mathcal{C}(W_0, z)$ and*

$$W^\star_{\mathrm{Muon}} = W_0 + \frac{rz^\top}{\|z\|_2^2}, \qquad W^\star_{\mathrm{Muon}} - W^\star_{\mathrm{TACO}} = r\left(\frac{z}{\|z\|_2^2} - \frac{\operatorname{sign}(z)}{\|z\|_1}\right)^\top.$$

Proof. The gradient can be represented via $G_t = g_t z^\top$, and $B_t = b_t z^\top$ by induction from $B_{-1} = 0$.

If $b_t \neq 0$, then

$$b_t z^\top = (b_t/\|b_t\|_2)(\|b_t\|_2\|z\|_2)(z/\|z\|_2)^\top$$

is a singular value decomposition, so $\mathcal{P}(B_t) = b_t z^\top/(\|b_t\|_2\|z\|_2)$.

Hence, every update has the form $az^\top$, so the iterates and their limit lie in the closed set $\mathcal{C}(W_0, z)$.

For $W = W_0 + az^\top$ we have $Wz = W_0 z + a\|z\|_2^2$, and $Wz = u^\star$ forces $a = r/\|z\|_2^2$, which gives the endpoint.

Subtracting the endpoint of Theorem 2 gives the displayed difference, and $z^\top z/\|z\|_2^2 = \operatorname{sign}(z)^\top z/\|z\|_1 = 1$ gives $(W^\star_{\mathrm{Muon}} - W^\star_{\mathrm{TACO}})z = 0$. □

The common target fixes the output correction $r = u^\star - W_0 z$, but the continuation set determines how this correction is represented in the weight space. We illustrate this distinction next through least squares, connecting our formulas to the optimizer mismatch analysis of Qu et al. (2026).

Remark 4. *(Extension to deep neural networks) Theorem 2 extends directly to deep networks under sequential layerwise fine-tuning. Consider*

$$h_0 = x, \qquad h_\ell = \sigma_\ell(W^{(\ell)} h_{\ell-1}), \qquad \ell = 1, \ldots, L,$$

and optimize one $W^{(\ell)}$ at a time while keeping all other layers fixed. During each stage, $h_{\ell-1}$ is fixed, so the objective depends on $W^{(\ell)}$ only through $W^{(\ell)} h_{\ell-1}$ and therefore satisfies Assumption 2 with $z = h_{\ell-1}$. Theorem 2 then implies that TACO and the Adam proxy reach the same endpoint for that layer. Since the resulting networks are identical, they produce the same input to the next layer, and repeating this argument over $\ell = 1, \ldots, L$ yields identical final networks.

Theorems 2 and 3 prove the general case. Below, we illustrate Theorems 2 and 3 through a specific example.

Least-squares example. Consider the least-squares objective function

$$F_{\mathrm{LS}}(W) = \frac{1}{2}\|Wz - u^\star\|_2^2,$$

where $W \in \mathbb{R}^{m\times n}$, $z \in \mathbb{R}^n$ is nonzero, and $u^\star \in \mathbb{R}^m$. This satisfies Assumption 2 with $\phi(u) = \frac{1}{2}\|u - u^\star\|_2^2$. Let $r = u^\star - W_0 z$. Proposition D.4 of Qu et al. (2026) gives the solutions

$$W^\star_{\mathrm{SignGD}} = W_0 + \frac{r\operatorname{sign}(z)^\top}{\|z\|_1}, \qquad W^\star_{\mathrm{Muon}} = W_0 + \frac{rz^\top}{\|z\|_2^2}$$

for SignGD and momentum-free Muon with exact orthogonalization from arbitrary initialization W_0. Their convergence result uses positive step sizes satisfying $\eta_t \to 0$ and $\sum_{t=0}^{\infty} \eta_t = \infty$.

Under Assumption 3, Theorem 2 gives

$$W^\star_{\mathrm{TACO}} = W^\star_{\mathrm{Adam}} = W^\star_{\mathrm{SignGD}},$$

while Theorem 3 gives the same Muon endpoint for the momentum-based proxy. Thus, TACO and the Adam proxy select the same limiting weights as SignGD in this setting.

These endpoints also characterize different minimum-norm corrections, where TACO and the Adam proxy inherit the minimum-max-norm correction identified for SignGD, whereas Muon selects a minimum-spectral-norm correction. For nonzero initialization, these statements concern the displacement $W - W_0$, rather than the norm of the final weights themselves.

The distinction between the continuation sets comes from their fixed right factors. In $\mathcal{C}(W_0, w)$, the coefficient $a \in \mathbb{R}^m$ varies, while $w \in \mathbb{R}^n$ is fixed. Every row of $W - W_0 = aw^\top$ is therefore a scalar multiple of $w^\top$. Changing a changes these row coefficients but cannot change their direction.

Both $\mathcal{C}(W_0, z)$ and $\mathcal{C}(W_0, \operatorname{sign}(z))$ have dimension m, but they coincide only when $z = c\operatorname{sign}(z)$ for some $c > 0$. A common nonzero correction

$$az^\top = b\operatorname{sign}(z)^\top$$

has a nonzero row satisfying $a_i z = b_i \operatorname{sign}(z)$, which forces this proportionality. Otherwise,

$$\mathcal{C}(W_0, z) \cap \mathcal{C}(W_0, \operatorname{sign}(z)) = \{W_0\},$$

and the two endpoints differ whenever $r \neq 0$.

The two factors are not interchangeable. In particular, $z\operatorname{sign}(z)^\top$ and $\operatorname{sign}(z)z^\top$ are both $n \times n$ matrices, so they are not admissible $m \times n$ corrections when $m \neq n$. Even when $m = n$, these products are generally different matrices.

For example, let

$$W_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad z = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, \quad u^\star = \begin{pmatrix} 3 \\ 0 \end{pmatrix}.$$

Then $W_0 z = (1, 2)^\top$, $r = (2, -2)^\top$, $\|z\|_1 = 4$, and $\|z\|_2^2 = 6$. Substituting into the endpoint formulas gives

$$W^\star_{\mathrm{TACO}} = W^\star_{\mathrm{Adam}} = \frac{1}{2}\begin{pmatrix} 3 & 1 & -1 \\ -1 & 1 & 1 \end{pmatrix},$$

$$W^\star_{\mathrm{Muon}} = \frac{1}{3}\begin{pmatrix} 4 & 2 & -1 \\ -1 & 1 & 1 \end{pmatrix}.$$

Direct multiplication verifies $W^\star_{\mathrm{TACO}} z = W^\star_{\mathrm{Adam}} z = W^\star_{\mathrm{Muon}} z = u^\star$.

Hence, both endpoints achieve zero least-squares loss. Their corrections exhibit their implicit biases:

$$\|W^\star_{\mathrm{TACO}} - W_0\|_{\max} = \frac{1}{2} < \frac{2}{3} = \|W^\star_{\mathrm{Muon}} - W_0\|_{\max},$$

whereas

$$\|W^\star_{\mathrm{Muon}} - W_0\|_2 = \frac{2}{\sqrt{3}} < \frac{\sqrt{6}}{2} = \|W^\star_{\mathrm{TACO}} - W_0\|_2.$$

More generally, $W_0 z$ is the initial output on the fixed input z, so $r = u^\star - W_0 z$ is precisely the output correction required by fine-tuning. The dependence on z records which input is being fitted; changing z changes the optimization problem. For $n > 1$, the target output does not determine a unique parameter matrix; the complete solution family is

$$\{W : Wz = u^\star\} = W^\star_{\mathrm{Muon}} + \{N \in \mathbb{R}^{m \times n} : Nz = 0\}.$$

Each continuation set contains exactly one matrix producing $u^\star$, so the update geometry selects one member of this solution family.

We next turn from the limiting weights to the width dependence of the update. For μP scaling, the relevant quantity is $\|D\|_2$, which is determined by how concentrated TACO's column-wise winners are across rows. Assumption 4 controls this concentration as m and n grow.

Theorem 4. *(TACO μP-scaling) Let Assumption 4 hold. Then the TACO direction $D = \frac{m}{n}\mathcal{S}(G)$ satisfies $\|D\|_2 = \frac{m}{n}\sqrt{\max_i c_i}$, for every G. Hence D satisfies the standard μP spectral scaling $\|D\|_2 = \Theta\left(\sqrt{\frac{m}{n}}\right)$.*

Proof. Since each column of $\mathcal{S}(G)$ contains one nonzero entry at row i_j, we have

$$\mathcal{S}(G)\mathcal{S}(G)^\top = \operatorname{diag}(c_1, \ldots, c_m).$$

Hence,

$$\|\mathcal{S}(G)\|_2 = \sqrt{\max_i c_i},$$

and the TACO direction $D = \frac{m}{n}\mathcal{S}(G)$ satisfies

$$\|D\|_2 = \frac{m}{n}\sqrt{\max_i c_i}.$$

This identity holds for every G and uses no assumptions. Dividing by $\sqrt{m/n}$ gives

$$\frac{\|D\|_2}{\sqrt{m/n}} = \sqrt{\frac{m}{n}\max_i c_i},$$

so the left side is $\Theta(1)$ exactly when $\max_i c_i = \Theta(n/m)$, which proves both directions. □

The proof reduces a spectral-norm question to a counting question, since $\|\mathcal{S}(G)\|_2 = \sqrt{\max_i c_i}$ exactly. The degenerate case makes the condition concrete. If a single row wins every column, then $\max_i c_i = n$ and $\|D\|_2 = m/\sqrt{n}$, which exceeds the μP target by a factor of $\sqrt{m}$.

A.3 Gradient History and Heavy Hitter Analysis

Finally, we connect the update rule to the sparse-history design in Section 5. The dense gradient history $\tilde{M}_t$ and sparse gradient history M_t need not agree entrywise. Rather than requiring their selected coordinates to agree directly, we give a sufficient condition based on the approximation error and the separation between the largest and second-largest entries of the dense gradient history.

Let $\tilde{M}_t = \mu\tilde{M}_{t-1} + (1-\mu)G_t$ and $M_t = \mathcal{H}_k(\mu M_{t-1} + (1-\mu)G_t)$ denote the dense and sparse gradient history. For every column j and iteration t, define the sparse-history approximation error as $\varepsilon_{t,j} := \|M_{t,:,j} - \tilde{M}_{t,:,j}\|_\infty$.

Theorem 5. *(Winner-margin preservation of TACO) Let $i^\star_{t,j} = \arg\max_i |\tilde{M}_{t,ij}|$ and define the dense-history winner margin as $\gamma_{t,j} = |\tilde{M}_{t,i^\star_{t,j},j}| - \max_{i \neq i^\star_{t,j}} |\tilde{M}_{t,ij}|$. If $\gamma_{t,j} > 2\varepsilon_{t,j}$ for every column j and iteration t, then $\mathcal{S}(M_t) = \mathcal{S}(\tilde{M}_t)$ for all $t \geq 0$.*

Proof. Fix a column j and iteration t, and let $i^\star = i^\star_{t,j}$. Then,

$$|M_{t,i^\star j}| \geq |\tilde{M}_{t,i^\star j}| - \varepsilon_{t,j}.$$

For any $i \neq i^\star$,

$$|M_{t,ij}| \leq |\tilde{M}_{t,ij}| + \varepsilon_{t,j}.$$

Since $\gamma_{t,j} > 2\varepsilon_{t,j}$,

$$|\tilde{M}_{t,i^\star j}| - \varepsilon_{t,j} > \max_{i \neq i^\star} |\tilde{M}_{t,ij}| + \varepsilon_{t,j}.$$

Therefore,

$$|M_{t,i^\star j}| > |M_{t,ij}|$$

for every $i \neq i^\star$, so the dense and sparse gradient histories select the same winning row.

Moreover, $\gamma_{t,j} > 2\varepsilon_{t,j}$ implies $|\tilde{M}_{t,i^\star j}| > \varepsilon_{t,j}$. Hence,

$$|M_{t,i^\star j} - \tilde{M}_{t,i^\star j}| \leq \varepsilon_{t,j} < |\tilde{M}_{t,i^\star j}|,$$

so the selected entries also have the same sign. By the definition of $\mathcal{S}$ in equation (4),

$$\mathcal{S}(M_t)_{:,j} = \mathcal{S}(\tilde{M}_t)_{:,j}.$$

Since this holds for every column j, we obtain

$$\mathcal{S}(M_t) = \mathcal{S}(\tilde{M}_t)$$

for all $t \geq 0$, which proves the result. □

The result shows that exact reconstruction of the dense gradient history is not required to preserve the TACO update. It is sufficient for the approximation error to remain smaller than half of the dense-history winner margin in every column. Thus, when the dominant coordinate is sufficiently separated from the remaining coordinates, sparse history compression can perturb the retained values while still producing exactly the same TACO direction as the dense gradient history. This provides a direct theoretical connection between TACO's winner-based update rule and the memory-efficient sparse-history design in Section 5.

B Experiment Setup and Additional Experiment Results

Model and datasets. For LLM experiments, we evaluate the OPT family (Zhang et al., 2022), including OPT-1.3B, OPT-6.7B, OPT-13B, and OPT-30B; the Qwen3 family (Yang et al., 2025), including Qwen3-8B, Qwen3-14B, and Qwen3-32B; as well as Pythia-12B (Biderman et al., 2023), Llama-3.1-8B (Dubey et al., 2024), and Mistral-Small-24B-Base-2501 (Mistral AI Team, 2025). We evaluate on SST-2 (Socher et al., 2013), RTE (Dagan et al., 2005; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), COPA (Roemmele et al., 2011), and MultiRC (Khashabi et al., 2018) from SuperGLUE (Wang et al., 2019), together with the question answering datasets SQuAD (Rajpurkar et al., 2016) and

Configuration	Setting
Fine-tuning	Full-parameter
Model precision	`BF16`
Maximum sequence length	1,024
Gradient accumulation	1
Learning rate	`3e-5`
Learning rate schedule	Linear
Warmup	50 steps
Gradient checkpointing	Enabled
Data seed	42
Default optimizer step budget	2,000 optimizer steps
Default batch size	4
TACO update	Column-wise Top-1
Gradient history EMA decay factor	$\mu = 0.95$
Heavy hitter threshold	$k = 16$ entries per column
Gradient history value format	`FP8 E4M3`
Gradient history index format	`int32`
Auxiliary AdamW	$\beta = (0.9, 0.999)$, $\varepsilon = 10^{-8}$, weight decay $= 0.01$

Table 1: **Shared experimental configuration.** All experiments use this setting unless otherwise specified.

DROP (Dua et al., 2019). Our experimental code is adapted from MeZO (Malladi et al., 2023), and we adopt the same task prompts as MeZO.

Implementation details. TACO treats model parameters as matrices and is applied to matrix-valued parameters. For ordinary attention and FFN linear weight matrices, TACO uses the geometry-derived m/n scaling in (5). Token embedding matrices are treated separately and use a scaling factor of 1, reflecting their one-hot lookup structure rather than an ordinary dense linear transformation. Following the common practice of matrix-based optimizers such as Muon (Jordan, 2024), scalar and vector parameters are optimized with an auxiliary AdamW optimizer. For models based on the GPT-NeoX architecture, such as Pythia, the query, key, and value projections are stored in a single fused QKV parameter. We respect this logical matrix structure by treating the fused parameter as three separate Q, K, and V projections for TACO history compression and coordinate selection, and applying TACO independently to each projection before writing the updates back to the fused parameter. This prevents the Q, K, and V projections from competing for a single selected coordinate in each column.

Hyperparameters. We group our experiments into four categories, E1-E4; see Table 2. Unless otherwise specified, all experiments use the shared hyperparameters summarized in Table 1. Experiment-specific hyperparameters and settings are summarized in Table 2.

E1 compares TACO against six baseline optimizers. For each optimizer, we perform a common learning rate sweep spanning 10^{-6} to 10^{-2} on SST-2, RTE, and BoolQ using the same training seed, batch size 1, and a 500-step budget, and select the final learning rate based on validation performance. The selected learning rate is then used in E1 experiments.

Hardware. All reported experiments are conducted on NVIDIA H100 80 GB HBM3 GPUs of the same model. Each device reports approximately 85.5 GB of total memory when expressed in decimal units (10^9 bytes). Peak GPU memory is measured as the maximum CUDA memory allocated during training and is reported in decimal GB. Runtime and throughput comparisons use the same GPU model and hardware configuration.

B.1 Optimizer comparison

We provide the complete OPT-13B comparison across all eight downstream tasks in Table 3. The corresponding SST-2 efficiency results, including persistent optimizer state, peak GPU memory, runtime, throughput, and validation behavior, are reported in Table 4. Each configuration is launched with five seeds; reported values average over successful runs. Failed runs are marked by $n/5$ or OOM. Although Adafactor has the smallest persistent optimizer state, its transient training memory is substantially higher, resulting in considerably larger peak GPU memory than TACO. FlashAdamW benefits from hardware-optimized fused kernels and gradient release, so its through-

Experiment	Model and task	Optimizer	Experiment specific hyperparameters
E1 Optimizer comparison Fix model, task	**Model:** OPT-13B **Tasks:** SST-2, RTE, BoolQ, COPA, CB, MultiRC, SQuAD, DROP	TACO	–
		AdamW8bit	LR = `3e-6`; 8-bit optimizer states; `FP32` embedding; weight decay $= 0.01$
		Adafactor	LR = `1e-4`; β_2 decay $= -0.8$; weight decay $= 0$
		APOLLO-Mini	LR = `3e-6`; rank $= 1$; projection gap $= 200$; tensor-wise scale $= 128$
		FlashAdamW	LR = `1e-6`; `BF16` weights; 8-bit weight correction; 8-bit moments; weight decay $= 0.01$
		GaLore	LR = `3e-5`; rank $= 128$; projection gap $= 200$; scale $= 0.25$; weight decay $= 0$
		Muon	LR = `3e-5`; momentum $= 0.95$; Nesterov; Newton-Schulz iteration = 5; auxiliary LR = `3e-4`
E2 Model scaling Fix optimizer, task	**Models:** OPT-{1.3B, 6.7B, 13B, 30B} Qwen3-{8B, 14B, 32B} **Tasks:** SST-2, RTE, BoolQ	TACO	–
E3.1 Optimizer step budget ablation	**Model:** OPT-1.3B **Tasks:** BoolQ, RTE, SQuAD	TACO	Steps $\in$ {2K, 4K, 8K, 16K}
E3.2 Gradient history EMA decay factor ablation			$\mu \in \{0, 0.95\}$; additionally evaluated on SST-2, COPA, CB, MultiRC, and DROP
E3.3 Heavy hitter threshold k ablation			$k \in \{\text{dense}, 64, 32, 16, 8, 4, 2, 1\}$
E3.4 Update sparsity Top-k ablation			Top-1, Top-2, Top-4, Top-8, Top-16, Top-32; heavy hitter threshold k = 32
E3.5 Batch size ablation	**Models:** Qwen3-{14B, 32B} **Tasks:** SST-2, RTE		BS $\in \{2, 4, 8, 16, 32, 64, 128\}$
E3.6 μP learning rate transfer study	**Models:** OPT-{1.3B, 6.7B, 13B, 30B} Qwen3-{8B, 14B, 32B} **Tasks:** SST-2, RTE		Learning rate sweep
E4 Architecture transfer Fix optimizer, task	**Models:** Pythia-12B, Llama-3.1-8B, Qwen3-32B, Mistral-24B **Tasks:** SST-2, RTE, BoolQ	TACO	–

Table 2: **Experiment configurations for E1-E4.** E1 compares optimizers with the model and tasks fixed. E2 studies TACO across model scales. E3 contains the ablation studies. E4 evaluates transfer across model architectures. Settings not explicitly listed follow Table 1. Note that TACO uses uniform hyperparameters in Table 1, except the ablation study category, E3, where only explicitly listed hyperparameters are varied.

put also reflects systems-level optimization (Gonzalez Ortiz et al., 2026). TACO remains the most memory-efficient method in terms of peak GPU memory while maintaining competitive downstream performance and practical training throughput. The main performance and memory observations are discussed in Section 6.

	Classification			Multiple Choice			Generation	
Method	**SST-2**	**RTE**	**BoolQ**	**COPA**	**CB**	**MultiRC**	**SQuAD**	**DROP**
TACO	94.2	80.8	71.5	89.4	87.1	61.2	82.6	30.8
AdamW8bit	95.3	83.5	62.1 (2/5)	88.2	71.8	OOM	88.1	OOM
Adafactor	95.5	87.3	82.0	87.0	90.0	75.1	88.6	37.7
APOLLO-Mini	93.0	83.5	77.8	87.0	80.7	64.2	88.8	36.7
FlashAdamW	95.5	86.3	64.8 (4/5)	86.8	85.7	57.5	88.4	39.8
GaLore	95.7	85.7	80.2	91.8	76.4	67.3	88.0	34.5
Muon	94.4	77.0 (4/5)	73.5 (4/5)	87.2	81.2 (2/5)	OOM	77.8	OOM

Table 3: **OPT-13B performance comparison across downstream tasks.** SST-2, RTE, BoolQ, COPA, and CB report accuracy; MultiRC reports F1A; SQuAD and DROP report F1. For entries marked $(n/5)$, only n of the five runs complete successfully and the remaining runs fail due to OOM.

Optimizer	Steps	State Mem. ↓ (GB)	Peak Mem. ↓ (GB)	Runtime ↓ (minutes)	Throughput ↑ (tokens/s)	Acc. ↑ (%)	Best Val. Step ↓
TACO	2K	0.159	27.54	14.92	275.8	94.22	1240
Adafactor	2K	0.013	52.59	11.77	349.7	95.50	840
AdamW8bit	2K	27.708	80.60	15.74	261.3	95.25	1380
APOLLO-Mini	2K	1.095	53.67	8.95	460.1	92.98	940
FlashAdamW	2K	40.167	67.68	7.55	546.7	95.48	940
GaLore	2K	2.655	56.71	132.34	31.1	95.71	1120
Muon	2K	26.248	80.01	101.98	40.3	94.38	680

Table 4: **OPT-13B fine-tuning efficiency on SST-2.** Best Val. Step reports the mean checkpoint selected by validation performance.

B.2 Model Scaling

Table 5 provides the complete scaling results for the OPT and Qwen3 model families on SST-2, RTE, and BoolQ. These results complement the scaling discussion in Section 6 and Figure 5 by reporting downstream performance and peak GPU memory for each model size. Results are averaged over three seeds.

Family	**Model Size**	**SST-2**		**RTE**		**BoolQ**	
		Acc.	**Mem. (GB)**	**Acc.**	**Mem. (GB)**	**Acc.**	**Mem. (GB)**
OPT	1.3B	92.9	3.4	69.6	4.0	63.7	5.5
	6.7B	94.4	14.8	74.0	15.5	68.1	17.3
	13B	94.7	27.6	78.5	28.6	72.3	31.0
	30B	93.3	62.5	79.5	64.4	67.1	68.2
Qwen3	8B	93.9	19.4	88.8	20.6	86.3	24.5
	14B	94.4	33.3	89.9	34.8	87.5	38.2
	32B	94.1	69.6	90.3	71.6	87.2	75.9

Table 5: Scaling behavior of TACO across the OPT and Qwen3 model families.

B.3 Ablation studies

E3.1: Optimizer step budget ablation. We vary the optimizer step budget from 2K to 16K on OPT-1.3B; results are reported in Table 6. TACO continues to improve beyond the default 2K-step

budget across all three tasks. Most of the improvement on RTE and SQuAD is obtained by 4K-8K steps, while BoolQ continues to improve through 16K steps. Relative to the 2K-step setting, the 16K-step budget improves BoolQ and RTE by 6.5 and 4.3 percentage points, respectively, and SQuAD by 2.1 F1 points. These results indicate that the default 2K-step budget does not represent TACO's performance ceiling. Results are averaged over three seeds.

Task	$T = 2\text{K}$	$T = 4\text{K}$	$T = 8\text{K}$	$T = 16\text{K}$	**Change**
BoolQ	68.6	72.6	73.3	**75.1**	+6.5 pp
RTE	72.6	76.5	76.7	**76.9**	+4.3 pp
SQuAD	78.6	80.0	**80.8**	80.7	+2.1 F1

Table 6: TACO optimizer step budget ablation on OPT-1.3B with $T \in \{2\text{K}, 4\text{K}, 8\text{K}, 16\text{K}\}$. The change is measured from $T = 2\text{K}$ to $T = 16\text{K}$.

E3.2: Gradient history EMA decay factor ablation. We examine the effect of the gradient history EMA decay factor by comparing the default $\mu = 0.95$ with the history-free setting $\mu = 0$ across all eight downstream tasks; results are reported in Table 7. The EMA gradient history improves performance on all six classification and multiple-choice tasks, with the largest gains on MultiRC (+8.9 F1A) and CB (+7.1 percentage points). The two generation tasks, SQuAD and DROP, favor $\mu = 0$ by 2.6 and 0.4 F1 points, respectively. Overall, these results support $\mu = 0.95$ as a uniform practical default. Results are averaged over three seeds.

Task	$\mu = 0$	$\mu = 0.95$	**Change**
SST-2	93.7	93.9	+0.3 pp
RTE	71.2	72.6	+1.3 pp
BoolQ	66.1	68.6	+2.5 pp
COPA	78.0	79.0	+1.0 pp
CB	79.2	86.3	+7.1 pp
MultiRC	53.5	62.5	+8.9 F1A
SQuAD	81.3	78.6	−2.6 F1
DROP	28.7	28.3	−0.4 F1

Table 7: Effect of gradient history EMA decay factor μ on OPT-1.3B. SST-2, RTE, BoolQ, COPA, and CB report accuracy; MultiRC reports F1A; SQuAD and DROP report F1.

E3.3: Heavy hitter threshold k ablation. We compare dense gradient histories with sparse gradient history retaining $k \in \{64, 32, 16, 8, 4, 2, 1\}$ entries per column; results are reported in Table 8. Moderately sparse gradient histories retain performance broadly comparable to dense gradient history while substantially reducing persistent optimizer state. In particular, $k = 16$ achieves the strongest BoolQ and RTE performance while using only 38.3 MB of optimizer state, a $34.4\times$ reduction from the 1317.7 MB required by dense gradient history. SQuAD is slightly better with dense or $k = 32$ history, while more aggressive compression at very small k begins to degrade performance. We therefore use $k = 16$ as the default heavy hitter threshold. Results are averaged over three seeds.

	Dense	$k = 64$	$k = 32$	$k = 16$	$k = 8$	$k = 4$	$k = 2$	$k = 1$
BoolQ	68.1	66.8	67.9	**68.6**	67.5	65.5	65.8	66.3
RTE	69.7	70.9	71.0	**72.6**	70.8	70.3	69.3	69.7
SQuAD	79.4	79.4	79.4	78.6	**78.9**	78.6	77.7	78.4
State memory (MB)	1317.7	145.4	74.0	38.3	20.4	11.5	7.0	4.8

Table 8: TACO heavy hitter threshold k ablation on OPT-1.3B.

E3.4: Update sparsity Top-k ablation. We relax TACO's theoretically derived column-wise Top-1 update and evaluate Top-k variants while fixing the heavy hitter threshold to $k = 32$; results are reported in Table 9. Top-1 achieves the strongest final performance across all three tasks. Top-2 remains broadly comparable, while performance begins to degrade for denser updates and deteriorates substantially at Top-8 and beyond, particularly on BoolQ and RTE. Although Top-2 and Top-4

can make faster progress during the early stage of training on some tasks, this advantage is transient and does not translate into better final performance. These results support retaining the column-wise Top-1 update used by TACO, which also corresponds exactly to the steepest-descent solution under the $1 \to 1$ geometry. Results are averaged over three seeds.

Task	Top-1 (TACO)	Top-2	Top-4	Top-8	Top-16	Top-32
BoolQ	**67.9**	66.6	67.8	61.9	61.5	61.5
RTE	**71.0**	70.4	69.3	53.6	56.1	47.1
SQuAD	**79.4**	79.0	76.8	76.0	73.0	70.7

Table 9: TACO update Top-k ablation on OPT-1.3B with heavy hitter threshold $k = 32$.

E3.5: TACO and batch sizes. We evaluate batch sizes from 2 to 128 on Qwen3-14B and Qwen3-32B using SST-2 and RTE; results are reported in Table 10. Larger batches increase peak GPU memory without consistently improving accuracy, so we retain batch size 4 as the default.

	Qwen3-14B				Qwen3-32B			
	SST-2		RTE		SST-2		RTE	
Batch	Acc.	Mem. (GB)	Acc.	Mem. (GB)	Acc.	Mem. (GB)	Acc.	Mem. (GB)
2	94.1 ± 0.9	33.1	89.5 ± 1.2	33.8	92.3 ± 2.6	69.3	89.4 ± 1.9	70.3
4	94.2 ± 1.0	33.3	89.2 ± 1.3	34.8	93.3 ± 1.6	69.6	$\mathbf{90.1} \pm 0.8$	71.5
8	93.9 ± 0.4	33.6	89.3 ± 1.9	36.6	93.7 ± 0.5	70.0	89.4 ± 0.9	74.0
16	94.5 ± 0.6	34.3	$\mathbf{89.8} \pm 0.8$	43.1	93.1 ± 0.9	71.0	89.3 ± 0.6	81.4
32	93.9 ± 1.7	35.8	89.0 ± 2.1	56.4	$\mathbf{94.0} \pm 0.7$	72.9	OOM	
64	$\mathbf{94.8} \pm 0.9$	40.1	OOM		93.5 ± 0.8	78.1	OOM	
128	94.1 ± 1.0	50.3	OOM		OOM		OOM	

Table 10: TACO batch size ablation on Qwen3-14B and Qwen3-32B. Accuracy is reported as mean $\pm$ sample standard deviation, and memory as peak GPU memory.

E3.6: μP Learning Rate Transfer in TACO.

We sweep learning rates across Qwen3-8B/14B/32B on SST-2 and OPT-1.3B/6.7B/13B/30B on SST-2 and RTE, using 1,000 optimizer steps, batch size 4, and five training seeds. As shown in Figure 7, every model-dataset configuration selects 3×10^{-5} as the best learning rate, providing empirical support for learning rate transfer across model scales. This contrasts with conventional scale-dependent tuning; for example, Brown et al. (2020) use progressively smaller learning rates as GPT-3 model size increases.

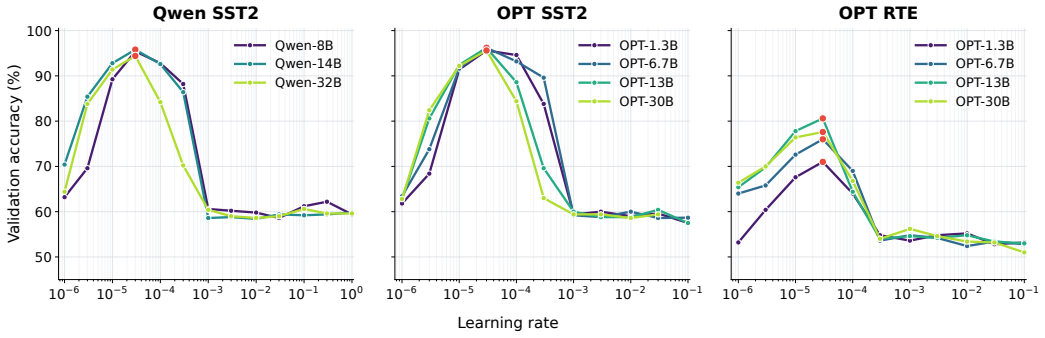

Figure 7: Every evaluated model and dataset configuration attains its best validation performance at 3×10^{-5}, empirically supporting the μP scaling result in Theorem 4

B.4 Architecture Transfer

The complete architecture transfer results for Pythia-12B, Llama-3.1-8B, Qwen3-32B, and Mistral-24B are reported and discussed in Section 6; see Figure 6.