POME: Post Optimization Model Edit via Muon-style Projection

Yong Liu[1]*, Di Fu[2], Yang Luo[2], Zirui Zhu[2], Minhao Cheng[3], Cho-Jui Hsieh[4], Yang You[2]
[1]Georgia Institute of Technology [2]National University of Singapore
[3]Pennsylvania State University [4]University of California, Los Angeles
{liuyong, youy}@comp.nus.edu.sg, mmc7149@psu.edu, chohsieh@cs.ucla.edu

Abstract

We introduce Post-Optimization Model Edit (POME), a new algorithm that enhances the performance of fine-tuned large language models using only their pretrained and fine-tuned checkpoints, without requiring extra data or further optimization. The core idea is to apply a Muon-style projection to ΔW, the difference between the fine-tuned and pretrained weights. This projection uses truncated singular value decomposition (SVD) to equalize the influence of dominant update directions and prune small singular values. As a simple post-processing step, POME is completely decoupled from the training pipeline. It requires zero modifications to training and no optimizer states. POME delivers consistent gains, boosting average performance by +2.5% on GSM8K and +1.0% on code generation. Its broad applicability—from 7B foundation models to 72B RLHF-instructed models—establishes it as a practical, zero-cost enhancement for fine-tuning pipelines. Code is available at https://github.com/NUS-HPC-AI-Lab/POME.

1 Introduction

The pretrain–finetune paradigm has been widely adopted: given pretrained model weights, a model can be adapted to domain-specific datasets (e.g., math or coding) via supervised fine-tuning (SFT) or reinforcement learning (RL) to enhance specialized abilities (Schulman et al., 2017; Ouyang et al., 2022). Let W_{pre} denote the pretrained weights and W_{ft} the weights after fine-tuning, we pose the following question:

Can we further improve the model using only W_{pre} and W_{ft}, without any additional data or training?

At first glance, this seems impossible since no new information is introduced. Surprisingly, we show that it is achievable through a novel algorithm we call **P**ost **O**ptimization **M**odel **E**dit (**POME**).

The key insight comes from recent advances in matrix-based optimizers, which often apply transformations to gradients or momentum to produce better updates (Jordan et al., 2024; Liu et al., 2025a; Team et al., 2025). For example, the Muon optimizer (Jordan et al., 2024) performs orthogonal projections on momentum at each step, ensuring that update directions contribute more uniformly rather than being dominated by a few principal components—a method validated in several recent LLM training works. Intuitively, we can treat $\Delta W = W_{\text{ft}} - W_{\text{pre}}$ as an aggregated update from the base optimizer (e.g., Adam across many steps), and it becomes plausible to refine this direction through a similar matrix-based transformation.

We instantiate this idea by applying truncated orthogonal projection, as in the Muon optimizer, to post-training weight edits. The intuition is that the benefits of orthogonalization in Muon do not

*Work done while a Ph.D. student at the National University of Singapore.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

fundamentally require per-step enforcement. The essential goal—ensuring that parameter changes span the space more uniformly—can also be achieved by projecting the final accumulated update. Therefore, given ΔW, our method computes its orthogonal basis and quantizes its singular values to two levels (constant and 0). This retains the most significant update directions while suppressing the influence of extremely high or low singular values. Crucially, this simple post-processing step requires no changes to training pipelines and no optimizer states. Because POME operates on checkpoints, it also applies to publicly released models whose training cannot be changed.

Empirical results demonstrate that POME consistently improves over standard fine-tuning across mathematical reasoning, code generation, and commonsense benchmarks, with ablations highlighting both its robustness and low computational cost. Specifically, POME achieves +2.5% on GSM8K and +0.7% on MATH for LLaMA2-7B (Touvron et al., 2023), +1.0% average improvement on HumanEval/MBPP code generation suites (Liu et al., 2023), and consistent gains across model scales from 7B to 72B parameters. The method also generalizes to RLHF-optimized models, improving Qwen2.5-Math (Yang et al., 2024) by +0.7% and Dr.GRPO (Liu et al., 2025c) by +2.1% on mathematical benchmarks.

Building on the theoretical insights and empirical findings, the main contributions of this paper are:

- We formalize post-training orthogonalization as a practical complement to per-step matrix-aware training, showing that the key benefits can be achieved through one-time geometric correction of accumulated weight changes;
- We present POME, a training-free method that applies truncated SVD and spectrum equalization to weight deltas, requiring zero training-time overhead and no infrastructure modifications;
- We demonstrate consistent improvements over standard fine-tuning across mathematical reasoning, code generation, and commonsense benchmarks, with comprehensive ablations characterizing the method's robustness and computational cost.

2 Preliminaries

2.1 Model Edit

Model editing (Yao et al., 2023) enables targeted modifications to a model's knowledge while avoiding adverse effects on other inputs. Methods mainly fall into two groups: Modify internal parameters and Add extra parameters.

Add extra parameters: Keep the original weights fixed and add new parameters to handle the revised facts (Mitchell et al., 2022; Huang et al., 2023; Li et al., 2025); adapters such as LoRA (Hu et al., 2022) likewise train added low-rank parameters. **Modify internal parameters**: Update some weights with a Δ matrix. This can be done with "Locate-Then-Edit". Locate-Then-Edit first finds the key weights to change, then edits them (Sundararajan et al., 2017; Vig et al., 2020; Singh & Jaggi, 2020; Geva et al., 2020; Mitchell et al., 2021; Dai et al., 2021; Meng et al., 2022a,b; Tan et al., 2023; Hase et al., 2023; Jiang et al., 2025; Liu et al., 2025b). However, these methods primarily address knowledge editing across multiple tasks, focusing on updating large language models with knowledge derived from diverse task domains. Weight-space methods are another paradigm for post-optimization model editing: model soups (Wortsman et al., 2022), ensembles (Dietterich, 2002), task arithmetic (Ilharco et al., 2022), TIES (Yadav et al., 2023), and DARE (Yu et al., 2024) combine multiple fine-tuned models, and EMA (Polyak & Juditsky, 1992) averages iterates along one trajectory. POME instead edits a single fine-tuning delta.

2.2 Muon

Muon (Jordan et al., 2024) is an optimizer designed for the 2D parameters of the hidden layers in neural networks. It processes the updates generated by SGD with momentum by applying a Newton-Schulz iteration as a post-processing step to each update before applying them to the parameters (Liu et al., 2025a; Amsel et al., 2025).

Following Liu et al. (2025a), given a parameter $W \in \mathbb{R}^{m\times n}$, gradient $g \in \mathbb{R}^{m\times n}$, and momentum $M \in \mathbb{R}^{m\times n}$, the update rule is:

$$M_t = \mu M_{t-1} + g_t, \tag{1}$$

$$U_t = \text{NewtonSchulz}(M_t), \tag{2}$$

$$W_t = W_{t-1} - \eta(U_t + \lambda W_{t-1}). \tag{3}$$

Here U_t approximates the closest semi-orthogonal matrix to M_t, normalizing all singular values to 1 so that each update direction receives equal scaling regardless of its original magnitude, preventing any single direction from dominating during training.

2.3 Matrix Orthogonalization

As previously discussed, Muon employs the Newton-Schulz iteration to compute an approximately semi-orthogonal matrix from M_t by solving:

$$\text{Orthogonal}(M_t) = \arg\min_{U_t^\top U_t = I} \|M_t - U_t\|_F^2, \tag{4}$$

where U_t is a semi-orthogonal matrix satisfying $U_t^\top U_t = I$, and $\|\cdot\|_F$ denotes the Frobenius norm.

Although Muon has demonstrated promising results (Jordan et al., 2024; Liu et al., 2025a; Team et al., 2025), its per-step matrix operations are impractical for large-scale distributed fine-tuning. In ZeRO/FSDP-style training (Rajbhandari et al., 2020; Paszke et al., 2019; Zhao et al., 2023), parameters are sharded across devices; even approximate orthogonalization (e.g., Newton–Schulz) must operate on full matrices, requiring frequent all-gathers to reassemble shards and subsequent redistributions. Moreover, the approximation introduces iteration and stabilization hyperparameters, adding numerical fragility and engineering overhead. In contrast, our method applies a Muon-like operator only once after post-training, avoiding the challenges associated with applying Muon at every step.

3 POME: Post-Optimization Model Edit

We first establish the problem setup and motivation, then present our core method: SVD decomposition, rank truncation, and orthogonalization.

3.1 Motivation and Problem Setup

We consider the post-training model edit setting. Formally, let W_{pre} denote the parameters of the pretrained base model. After fine-tuning with a standard optimizer (e.g., Adam), we obtain

$$W_{\text{ft}} = W_{\text{pre}} + \Delta W, \quad \Delta W = W_{\text{ft}} - W_{\text{pre}}, \tag{5}$$

where $\Delta W = \{\Delta W_\ell\}_{\ell=1}^{L}$ collects the layer-wise weight updates. Our goal is to produce an edited model

$$W_E = W_{\text{pre}} + \text{Edit}(\Delta W), \tag{6}$$

where $\text{Edit}(\cdot)$ is a post-hoc transformation of the fine-tuning update aimed at improving evaluation performance while controlling drift.

Our motivation builds on Muon (Sec. 2.2). Muon improves stability and generalization by equalizing the singular values of stepwise updates, which shapes the update subspace during

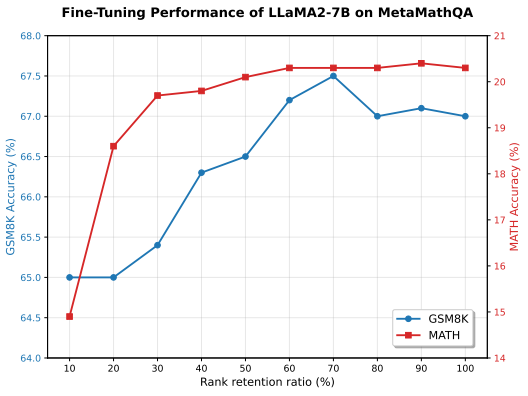

Figure 1: Effect of rank retention ratio on POME performance for LLaMA2-7B fine-tuned on MetaMathQA.

training. The same geometric effect can be applied once to the final accumulated update ΔW. The central idea is to act directly on ΔW so that update energy is distributed more uniformly across principal directions, reducing interference from nearly collinear components without touching the training loop.

We next examine the spectrum of ΔW. In this experiment, we take a fine-tuned model and truncate the trailing singular values while keeping the leading singular values/vectors unchanged. The results in Figure 1 indicate that the trailing singular vector spaces do not contribute much to the performance and the adaptation signal is concentrated in the leading subspace. This motivates a post-training orthogonalization method that, for each edited layer, decomposes the update by singular value decomposition, keeps a compact subspace, and replaces the singular values of the retained components with a common level to rebalance directions. The full procedure is presented in Sec. 3.2.

3.2 Method

Inspired by the above empirical observations and the optimization objective from Muon (Yang et al., 2023; Bernstein, 2025), we formulate POME as a low-rank post-training edit under an RMS-to-RMS operator-norm constraint. Specifically, the RMS-to-RMS operator norm measures the maximum factor by which a matrix amplifies the RMS norm of its input:

$$\begin{aligned} \|\Delta W\|_{\mathrm{RMS}\to\mathrm{RMS}} &:= \max_{x\neq 0} \frac{\|\Delta W x\|_{\mathrm{RMS}}}{\|x\|_{\mathrm{RMS}}} \\ &= \sqrt{\frac{d_{\mathrm{in}}}{d_{\mathrm{out}}}}\,\|\Delta W\|_2. \end{aligned} \tag{7}$$

where $\|\cdot\|_2$ represents the spectral norm.

Definition: Given a layer-wise fine-tuning delta ΔW and an RMS-to-RMS operator-norm budget η (Bernstein, 2025), we seek an edit P of rank at most k that maximizes alignment with ΔW, measured by the Frobenius inner product:

$$\max_{P:\mathrm{rank}(P)\leq k} \langle \Delta W, P\rangle \quad \text{s.t. } \|P\|_{\mathrm{RMS}\to\mathrm{RMS}} \leq \eta,$$

where $\langle\cdot,\cdot\rangle$ represents the Frobenius inner product. By Theorem G.2, a closed-form optimizer is given by

$$P^* = \eta\sqrt{\frac{d_{\mathrm{out}}}{d_{\mathrm{in}}}} U_k V_k^T,$$

where d_{out} and d_{in} are the output and input dimensions of each layer, and $U_k V_k^T$ corresponds to setting the top-k singular values to unity while zeroing out the rest. This is similar to the Muon formulation, except for the low-rank constraint, which adds a smaller gain on top of equalization (Table 7).

The closed-form solution naturally suggests a practical algorithm: perform SVD decomposition on layer-wise weight matrix ΔW, retain the top-k singular vectors (as guided by Figure 1), and reconstruct with equalized singular values. This leads us to POME, a simple post-training procedure that implements this principle.

Concretely, for a target layer ℓ with delta $\Delta W_\ell \in \mathbb{R}^{m_\ell \times d_\ell}$ from standard fine-tuning, POME decomposes the update via SVD, retains the top-k components based on our empirical analysis, and reconstructs with equalized singular values through the following procedure:

Step 1: SVD decomposition. For a target layer delta weight $\Delta W \in \mathbb{R}^{m\times d}$, compute its singular value decomposition:

$$\Delta W \;=\; U\,\Sigma\,V^\top, \tag{8}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{d\times d}$ are orthogonal, and $\Sigma = \mathrm{diag}(\sigma_1 \geq \cdots \geq \sigma_r \geq 0)$.

Step 2: Rank truncation. Retain only the top-k components to suppress insignificant directions:

$$U_k \;=\; U[:,\,1{:}k], \qquad V_k \;=\; V[:,\,1{:}k], \tag{9}$$

Algorithm 1 POME

Require: Base weights W_{pre}, fine-tuning delta ΔW, edited layer set $\mathcal{L}$
Require: Retention fraction $0 < K \leq 1$; $k_\ell = \lfloor K r_\ell \rfloor$, $1 \leq k_\ell \leq r_\ell = \min(m_\ell, d_\ell)$; global $\beta > 0$
Ensure: Edited model weights W_E

$W_E \leftarrow W_{\text{pre}} + \Delta W$
for each $\ell \in \mathcal{L}$ **do**
  if $\|\Delta W_\ell\|_F > 0$ **then**
    $\Delta W_\ell = U_\ell \Sigma_\ell V_\ell^\top$
    $\widehat{\Delta W}_{\perp,\ell} = U_{\ell,k_\ell} V_{\ell,k_\ell}^\top$
    $\alpha_\ell = \beta \|\Delta W_\ell\|_F / \sqrt{k_\ell}$
    $\widehat{\Delta W}_\ell = \alpha_\ell \widehat{\Delta W}_{\perp,\ell}$
    $W_{E,\ell} \leftarrow W_{\text{pre},\ell} + \widehat{\Delta W}_\ell$
  end if
end for
return W_E

Step 3: Orthogonalization and Reconstruction. For each edited layer ℓ, equalize the retained singular values and apply a layer-dependent coefficient α_ℓ:

$$\widehat{\Delta W}_{\perp,\ell} = U_{\ell,k_\ell} V_{\ell,k_\ell}^\top, \qquad \widehat{\Delta W}_\ell = \alpha_\ell \widehat{\Delta W}_{\perp,\ell},$$
$$\alpha_\ell = \beta \frac{\|\Delta W_\ell\|_F}{\|\widehat{\Delta W}_{\perp,\ell}\|_F} = \beta \frac{\|\Delta W_\ell\|_F}{\sqrt{k_\ell}}. \tag{10}$$

Here $\beta > 0$ is a single global multiplier shared by all edited layers; α_ℓ is computed analytically and is not tuned independently. Since $\|U_{\ell,k_\ell} V_{\ell,k_\ell}^\top\|_F = \sqrt{k_\ell}$, the edited delta satisfies $\|\widehat{\Delta W}_\ell\|_F = \beta \|\Delta W_\ell\|_F$, preserving the relative layer-wise update magnitudes. A zero delta is left unchanged. This rule corresponds to the layer-dependent RMS budget $\eta_\ell = \alpha_\ell \sqrt{d_{\text{in},\ell}/d_{\text{out},\ell}}$ in Theorem G.2. We use a common retention fraction $K = 0.5$ as the default and set $r_\ell = \min(m_\ell, d_\ell)$ and $k_\ell = \lfloor K r_\ell \rfloor$. Here, r_ℓ is the number of singular values returned by the reduced SVD, and the retained count k_ℓ is rounded down. For models we fine-tune, β is selected from a small grid on a subset of the fine-tuning data; for the third-party RL checkpoints, on a small public math proxy set disjoint from the test benchmarks; in the checkpoint-only setting (Appendix B), $\beta = 2.0$. Sensitivity to K and β is reported in Appendix E.

Final edited model. We keep unedited layers unchanged and replace edited layers by their orthogonalized deltas:

$$W_E = W_{\text{pre}} + \widehat{\Delta W}. \tag{11}$$

POME is applied uniformly to all 2D weight matrices (attention projections and FFN layers); embeddings, RMSNorm weights, and biases are left unchanged. A layer-wise ablation showing that FFN layers are the primary contributor is provided in Appendix D.

4 Experiments

We evaluate POME on supervised fine-tuning across mathematical reasoning, code generation, and commonsense tasks, then extend to publicly available fine-tuned models and RLHF-optimized models, with ablations isolating the contribution of key components.

4.1 Experimental Setting

Mathematical Reasoning. We fine-tune LLaMA2-7B (Touvron et al., 2023), LLaMA3-8B (Grattafiori et al., 2024), and Gemma2-9B models on MetaMathQA (Yu et al., 2023) and evaluate on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) benchmarks, which assess mathematical problem-solving capabilities.

Code Generation. We employ Code-Feedback (Zheng et al., 2024) as the training dataset for LLaMA2-7B and LLaMA3-8B, evaluating on HumanEval (Chen et al., 2021), HumanEval+, MBPP, and MBPP+ from EvalPlus (Liu et al., 2023).

Table 1: POME on mathematical reasoning. Rows 1–6: three models fine-tuned on MetaMathQA, Mean±Std over seeds 42/87/100, POME improves all 18 (model, task, seed) combinations. Per-seed details in Appendix A.

Model	Task	Adam (Mean±Std)	+POME (Mean±Std)	Gain
LLaMA2-7B	GSM8K	67.0 ± 0.1	$\mathbf{69.5 \pm 0.4}$	+2.5
LLaMA2-7B	MATH	19.1 ± 0.2	$\mathbf{19.8 \pm 0.1}$	+0.7
LLaMA3-8B	GSM8K	80.7 ± 0.5	$\mathbf{81.3 \pm 0.4}$	+0.6
LLaMA3-8B	MATH	31.6 ± 0.2	$\mathbf{32.8 \pm 0.1}$	+1.2
Gemma2-9B	GSM8K	81.6 ± 0.5	$\mathbf{82.9 \pm 0.3}$	+1.3
Gemma2-9B	MATH	36.6 ± 0.5	$\mathbf{37.5 \pm 0.6}$	+0.9
MetaMath-Mistral-7B	GSM8K	77.9	**79.5**	+1.6
MetaMath-Mistral-7B	MATH	28.1	**28.5**	+0.4

Commonsense Reasoning. Following LLM-adapters (Hu et al., 2023), we train on Commonsense-170k and evaluate across eight benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-C (Clark et al., 2018), ARC-E (Clark et al., 2018), and OBQA (Mihaylov et al., 2018).

RLHF: We investigate POME's applicability beyond supervised fine-tuning and evaluate on models optimized with reinforcement learning from human feedback (RLHF), such as open-sourced Qwen2.5-Math-72B-Instruct (Yang et al., 2024) model and variants of GRPO (Shao et al., 2024) (Dr.GRPO (Liu et al., 2025c)). These are publicly released third-party checkpoints; their results are single point estimates.

Reproducibility. We repeat our controlled MetaMathQA fine-tuning experiments with three random seeds (42, 87, and 100), which affect data order and optimization randomness. Table 1 reports the mean and standard deviation across these runs; full per-seed results are provided in Appendix A. Single-seed results are identified in the corresponding table captions. Unless otherwise stated, we use the same training data, number of optimization steps, and evaluation configuration across compared methods. Each POME result is compared with its unedited fine-tuned checkpoint, without additional optimization. Training hyperparameters are detailed in Table 12.

4.2 Mathematical Reasoning

We train each model family with three random seeds to ensure robustness against initialization variance. Table 1 summarizes results across three model families and three random seeds. POME yields consistent improvements on both GSM8K and MATH across all models and seeds—e.g., LLaMA2-7B GSM8K rises from 67.0 ± 0.1 to 69.5 ± 0.4—confirming the gains are not driven by a particular initialization. The bottom two rows show that POME also improves MetaMath-Mistral-7B, a publicly released third-party checkpoint with an unknown training recipe, showing that the gain does not depend on our training pipeline. POME also generalizes to heavily-optimized base→instruct checkpoints (+0.3–0.6%, no regressions); see Appendix B for full results.

4.3 Code Generation

We evaluate POME on code generation using two training corpora (Code-Feedback and Magicoder-Evol-Instruct-110K) and standard functional benchmarks (HumanEval, HumanEval+, MBPP, MBPP+). As shown in Table 2, POME improves the average score in all three settings and 11 of the 12 benchmark scores over standard fine-tuning. On Code-Feedback, POME improves the average score of LLaMA2-7B by approximately 1.0% (e.g., +0.6% on HumanEval, +1.2% on HumanEval+, +0.8% on MBPP, and +1.3% on MBPP+), with similar gains observed for LLaMA3-8B. On Magicoder-Evol-Instruct-110K, the improvements are more pronounced, reaching an average gain of around 1.7% across functional benchmarks.

Table 2: Code generation results on LLaMA2-7B and LLaMA3-8B (lr=5e-6, seed=87, $K = 0.5$).

Model	Dataset	Method	HumanEval	HumanEval+	MBPP	MBPP+	Average
LLaMA2-7B	CodeFeedback	Adam	34.8	32.3	40.2	32.8	35.0
LLaMA2-7B	CodeFeedback	+POME	**35.4 (↑0.6)**	**33.5 (↑1.2)**	**41.0 (↑0.8)**	**34.1 (↑1.3)**	**36.0 (↑1.0)**
LLaMA3-8B	CodeFeedback	Adam	57.3	52.4	64.6	56.1	57.6
LLaMA3-8B	CodeFeedback	+POME	**58.6 (↑1.3)**	**53.6 (↑1.2)**	**65.2 (↑0.6)**	**58.3 (↑2.2)**	**58.9 (↑1.3)**
LLaMA2-7B	Magicoder-Evol	Adam	47.0	43.3	31.5	27.2	37.3
LLaMA2-7B	Magicoder-Evol	+POME	**49.4 (↑2.4)**	**43.3 (↑0.0)**	**33.9 (↑2.4)**	**29.4 (↑2.2)**	**39.0 (↑1.7)**

In contrast to mathematical reasoning tasks, improvements on code generation are more evenly distributed across evaluation metrics.

4.4 Commonsense Reasoning

We fine-tune LLaMA2-7B and LLaMA3-8B on Commonsense-170k and evaluate them on eight downstream commonsense reasoning benchmarks. As shown in Table 3, POME consistently improves average accuracy for both models, with gains distributed over most tasks rather than driven by a single outlier. The exception is BoolQ, which decreases from 73.4 to 72.0 for LLaMA2-7B. More pronounced improvements are observed on ARC-C and HellaSwag, which emphasize compositional and counterfactual reasoning (e.g., +2.2% on HellaSwag and a +0.8% average gain for LLaMA2-7B, and up to +3.1% on ARC-C with a +1.0% average gain for LLaMA3-8B).

Table 3: Accuracy of Fine-Tuning LLaMA2-7B and LLaMA3-8B (lr=5e-6, seed=87, $K = 0.5$) on Commonsense-170k.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-C	ARC-E	OBQA	Average
LLaMA2-7B	**Adam**	**73.4**	83.4	**80.1**	89.4	83.3	70.4	83.9	81.4	80.7
LLaMA2-7B	**+POME**	72.0	**84.3**	**80.1**	**91.6**	**84.4**	**71.2**	**85.7**	**82.8**	**81.5** (↑ **0.8**)
LLaMA3-8B	**Adam**	75.7	88.0	79.4	95.1	85.5	78.5	90.1	86.4	84.8
LLaMA3-8B	**+POME**	**75.9**	**88.3**	**79.9**	**95.5**	**86.7**	**81.6**	**90.5**	**87.6**	**85.8** (↑ **1.0**)

4.5 RLHF Model

Table 4 reports results on Qwen2.5-Math models at two scales (7B and 72B), where the weight deltas come from a post-training pipeline that includes RL, across a diverse suite of mathematical reasoning benchmarks. POME improves average performance for both model sizes (+0.7% for 7B and +0.8% for 72B), with gains distributed across most tasks rather than concentrated on a single benchmark. The largest gains are on Minerva Math, Gaokao, and OlympiadBench.

Table 4: Performance of Qwen2.5-Math models (7B and 72B) with and without POME across mathematical reasoning benchmarks.

Model	GSM8K	MATH	Minerva Math	Gaokao 2023 EN	Olympiad Bench	College Math	MMLU STEM	Average
Qwen2.5-Math-72B	**95.8**	86.0	43.8	71.9	48.3	**49.6**	80.2	67.9
Qwen2.5-Math-72B + POME	95.5	**86.7**	**44.5**	**73.8**	**50.4**	49.3	**80.4**	**68.7** (↑ **0.8**)
Qwen2.5-Math-7B	95.8	**83.5**	33.8	67.8	41.5	46.9	**69.3**	62.7
Qwen2.5-Math-7B + POME	**96.0**	**83.5**	**36.0**	**69.6**	**42.2**	**47.1**	69.2	**63.4** (↑ **0.7**)

We further evaluate POME on a Qwen2.5-Math checkpoint trained with Dr.GRPO, which uses group-relative policy optimization. As shown in Table 5, POME improves three of five benchmarks, raising average accuracy from 51.2% to 53.3% (+2.1 percentage points). Gains are largest on AIME24, from 43.3% to 46.7% (+3.4 points), and AMC, from 59.0% to 66.3% (+7.3 points). OlympiadBench improves from 41.9% to 42.5% (+0.6 points), while MATH500 decreases from 80.2% to 79.6% (-0.6 points) and Minerva from 31.6% to 31.3% (-0.3 points). The reported average gives equal weight to each of the five benchmarks, regardless of evaluation-set size. These single-checkpoint point estimates do not measure variability across training runs.

Table 5: Dr.GRPO accuracy (%) with and without POME on Qwen2.5-Math. Δ: change in percentage points; Average: five-task mean. Results are single-checkpoint point estimates.

Benchmark	**Dr.GRPO**	**+POME**	**Δ (pp)**
AIME24	43.3	**46.7**	$+3.4$
AMC	59.0	**66.3**	$+7.3$
MATH500	**80.2**	79.6	-0.6
Minerva	**31.6**	31.3	-0.3
OlympiadBench	41.9	**42.5**	$+0.6$
Average	51.2	**53.3**	$+2.1$

4.6 Ablation Study

Generalization Analysis. To assess whether POME improves generalization, a goal shared by training-time methods (Liu et al., 2022; Luo et al., 2025), we compare training and test accuracy before and after the edit. Table 6 reports results for LLaMA3-8B and Gemma2-9B fine-tuned on MetaMathQA across GSM8K and MATH. Across both model families and tasks, training accuracy remains nearly unchanged after applying POME (within $\pm 0.3\%$), while test accuracy consistently improves. For example, on LLaMA3-8B, test accuracy increases from 80.3% to 81.4% on GSM8K and from 31.5% to 32.7% on MATH, despite a negligible decrease in training accuracy. Similar trends are observed for Gemma2-9B, where test accuracy improves by around 1.1–1.2% on both benchmarks with minimal impact on training performance.

Table 6: Generalization analysis of LLaMA3-8B and Gemma2-9B fine-tuned on MetaMathQA (lr=5×10^{-6}, seed=87, $K = 0.5$). Checkmarks denote POME editing. Accuracies are percentages; arrows indicate percentage-point changes relative to the unedited checkpoint. POME leaves training accuracy nearly unchanged (absolute change ≤ 0.3 percentage points) while consistently improving test accuracy.

Model	**Task**	**POME**	**Train Acc**	**Test Acc**
LLaMA3-8B	GSM8K		94.3	80.3
LLaMA3-8B	GSM8K	✓	94.2 (↓0.1)	**81.4** (↑1.1)
LLaMA3-8B	MATH		54.7	31.5
LLaMA3-8B	MATH	✓	54.5 (↓0.2)	**32.7** (↑1.2)
Gemma2-9B	GSM8K		95.8	82.2
Gemma2-9B	GSM8K	✓	95.9 (↑0.1)	**83.3** (↑1.1)
Gemma2-9B	MATH		63.1	36.1
Gemma2-9B	MATH	✓	62.8 (↓0.3)	**37.3** (↑1.2)

This divergence between stable training accuracy and improved test accuracy is consistent with a reduced generalization gap. These gains are obtained without additional optimization.

Rank Truncation and Equalization Analysis. Table 7 isolates the contribution of each component by testing all four combinations: no operation (baseline), truncation only, equalization only, and both together. The result is clear: *truncation alone* provides only marginal improvement (+0.2% average for both models), confirming that simple low-rank projection of the weight delta does not drive POME's gains. *Equalization alone* yields a 3.5–5$\times$ larger gain (+0.7–1.0%), identifying spectrum equalization—replacing retained singular values with a uniform constant—as the primary mechanism. This is motivated by Muon's RMS-to-RMS norm formulation (Theorem G.2): equalization directly addresses the spectral imbalance in accumulated ΔW where a few dominant directions overshadow

Table 7: Ablation of truncation and equalization on MetaMathQA (seed=87), with lr=10^{-5} for LLaMA2-7B and lr=5×10^{-6} for LLaMA3-8B. Trunc. retains the top 50% singular components ($K = 0.5$); Equal. sets the retained singular values to a common magnitude. Checkmarks indicate enabled operations; blank entries indicate disabled operations. Avg averages GSM8K and MATH accuracies, and parenthesized values are percentage-point gains over the corresponding unedited baseline. For LLaMA2-7B, equalization yields +1.0 points and adding truncation provides a further +0.4 points.

Model	Trunc.	Equal.	GSM8K	MATH	Avg
LLaMA2-7B			67.2	19.4	43.3
LLaMA2-7B	✓		67.5	19.4	43.5 (+0.2)
LLaMA2-7B		✓	69.0	19.6	**44.3** (+**1.0**)
LLaMA2-7B	✓	✓	69.7	19.7	**44.7** (+**1.4**)
LLaMA3-8B			80.3	31.5	55.9
LLaMA3-8B	✓		80.6	31.6	56.1 (+0.2)
LLaMA3-8B		✓	81.0	32.1	**56.6** (+**0.7**)
LLaMA3-8B	✓	✓	81.4	32.7	**57.1** (+**1.2**)

others. Adding truncation on top of equalization provides a further +0.4% (total +1.4%), as it prevents low-energy components from being amplified to the same uniform level during equalization.

Together, the two operations play complementary but asymmetric roles: equalization is the primary driver of improvement, while truncation serves as a modest safeguard that prevents low-energy directions from being amplified during equalization. In our POME ablation, applying only low-rank truncation to the fine-tuning weight delta yields a modest gain. GaLore (Zhao et al., 2024) applies low-rank projection to gradients during training.

Table 8: Comparison of POME with training-time baselines and alternative optimizers on LLaMA2-7B (MetaMathQA, seed=42). Bold marks the best result in each column.

Category	Method	GSM8K	MATH
Training-time	Adam (baseline)	66.9	19.0
	NEFTune	67.7	19.7
	EMA	67.5	18.8
Alt. optimizer / arch.	LoRA (r=64)	65.7	17.8
	LoRA (r=256)	66.4	18.5
	Muon	67.2	19.4
+ POME	LoRA (r=64) + POME	67.6	19.2
	Adam + POME	69.0	**19.8**
	Muon + POME	**69.4**	**19.6**

Comparison with Baselines. Table 8 compares POME against training-time regularizers (NEFTune (Jain et al., 2023), exponential moving average (EMA)) and alternative training methods (LoRA (Hu et al., 2022), Muon (Jordan et al., 2024)) on LLaMA2-7B (MetaMathQA, seed=42; single-run point estimates). NEFTune yields modest improvements, while EMA shows mixed effects: it improves GSM8K accuracy but reduces MATH accuracy. On LLaMA3-8B, NEFTune and EMA reach 80.9/31.8 and 80.3/31.5 on GSM8K/MATH, versus 80.3/31.5 for Adam and 81.4/32.7 for POME. POME outperforms both baselines on both benchmarks for both models.

These results highlight differences in how the methods operate on model parameters. NEFTune introduces stochastic regularization during optimization, whereas EMA averages checkpoints along the same training trajectory. EMA's limited gains may reflect the limited diversity among these correlated

checkpoints. Neither method explicitly targets the singular-value spectrum of the accumulated update. POME instead directly modifies the post-training weight delta through rank truncation and spectrum equalization, addressing spectral imbalance without requiring training-time modifications.

LoRA underperforms full fine-tuning at both ranks, and Muon matches Adam due to the optimizer-mismatch effect on Adam-pretrained models (Liu et al., 2025a). Critically, POME is *complementary* to Adam, Muon, and LoRA training: Muon+POME achieves the best overall result, confirming that post-hoc spectral correction addresses residual imbalance that even per-step orthogonalization leaves in ΔW. POME further applies to LoRA's $\Delta W{=}BA$, yielding +1.9% GSM8K over LoRA alone. Appendix F further reports matched post-hoc controls on the same delta, Muon- and Lion-trained checkpoints, multi-seed code and commonsense results, shared versus layer-dependent budgets, and a LayerNorm-based model.

5 Conclusion

In this paper, we introduce POME, a training-free post-optimization method that applies matrix orthogonalization to fine-tuned language models through truncated SVD and spectrum equalization of weight deltas. By shifting orthogonalization from per-step training operations to a one-shot post-processing approach, POME eliminates the scalability issues of methods like Muon while preserving their geometric benefits. Experiments across mathematical reasoning, code generation, and commonsense tasks demonstrate consistent improvements over standard fine-tuning with zero training overhead. POME's simplicity and broad applicability establish post-hoc subspace shaping as a practical complement to matrix-aware training methods.

6 Limitations

Our evaluation focuses on text-only Transformer language models, with extensions to other modalities left for future work. POME's gains vary with the global multiplier β, for which we provide sensitivity analysis and selection guidance in Appendix E.

Acknowledgments and Disclosure of Funding

We would like to acknowledge that computational work involved in this research work is supported by NUS IT's Research Computing group. Yang You's research group is being sponsored by ByteDance grant, Alibaba grant, Meta Research Award, AliCloud grant, Apple grant, Google Research Award, Microsoft Research grant, and Adobe gift. This work is partially supported by NSF 2048280, 2325121, 2244760, 2331966 and ONR N00014-23-1- 2300:P00001.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The Falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Thomas G. Dietterich. Ensemble learning. In Michael A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 405–408. MIT Press, Cambridge, MA, 2nd edition, 2002.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783v3.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*, 2023.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*, 2025.

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/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. *arXiv preprint arXiv:2502.05759*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12350–12360. IEEE, 2022.

Yong Liu, Di Fu, Shenggan Cheng, Zirui Zhu, Yang Luo, Minhao Cheng, Cho-Jui Hsieh, and Yang You. SeedloRA: A fusion approach to efficient LLM fine-tuning. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=7QH48TtFZX.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

Yang Luo, Zangwei Zheng, Ziheng Qin, Zirui Zhu, Yong Liu, and Yang You. MERIT: Maximum-normalized element-wise ratio for language model large-batch training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41566–41585. PMLR, 2025. URL https://proceedings.mlr.press/v267/luo25u.html.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Leon Mirsky. A trace inequality of John von Neumann. *Monatshefte für Mathematik*, 79:303–306, 1975. doi: 10.1007/BF01647331.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pp. 15817–15831. PMLR, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. *arXiv preprint arXiv:2311.04661*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

John von Neumann. Some matrix-inequalities and metrization of matric-space. *Tomsk University Review*, 1:286–300, 1937.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=fq0NaiU8Ex.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

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. PMLR, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

A Per-Seed Results on MetaMathQA

Table 9 reports the full per-seed breakdown for the results summarised in Table 1 of the main text. POME yields positive improvement in all 18 (model, task, seed) combinations.

Table 9: Per-seed results on MetaMathQA for LLaMA2-7B, LLaMA3-8B and Gemma2-9B. Subscripts denote random seed.

Model	Task	Adam_{42}	+POME	Adam_{87}	+POME	Adam_{100}	+POME
LLaMA2-7B	GSM8K	66.9	**69.0 (↑2.1)**	67.2	**69.7 (↑2.5)**	66.9	**69.9 (↑3.0)**
LLaMA2-7B	MATH	19.0	**19.8 (↑0.8)**	19.4	**19.7 (↑0.3)**	19.0	**20.0 (↑1.0)**
LLaMA3-8B	GSM8K	80.6	**80.9 (↑0.3)**	80.3	**81.4 (↑1.1)**	81.3	**81.7 (↑0.4)**
LLaMA3-8B	MATH	31.8	**32.7 (↑0.9)**	31.5	**32.7 (↑1.2)**	31.6	**32.9 (↑1.3)**
Gemma2-9B	GSM8K	81.0	**82.6 (↑1.6)**	82.2	**83.3 (↑1.1)**	81.5	**82.7 (↑1.2)**
Gemma2-9B	MATH	36.4	**36.9 (↑0.5)**	36.1	**37.3 (↑1.2)**	37.3	**38.3 (↑1.0)**

B POME on Base→Instruct Checkpoints

Table 10 evaluates POME on publicly released base→instruct checkpoint pairs with a fixed global multiplier β=2.0 (Equation (10); no benchmark tuning). Gains are smaller than in controlled fine-tuning experiments (+0.3–0.6%), and no benchmark decreases.

Table 10: POME on publicly released base→instruct pairs (fixed β=2.0, no tuning).

Model	Benchmark	Instruct	+POME
LLaMA-2-7B → Chat	GSM8K	23.9	**24.5** (+0.6)
LLaMA-2-7B → Chat	MMLU	48.1	**48.4** (+0.3)
LLaMA-3-8B → Instruct	GSM8K	79.6	**80.1** (+0.5)
LLaMA-3-8B → Instruct	MMLU	68.4	**68.8** (+0.4)
Qwen2.5-7B → Instruct	GSM8K	82.4	**82.9** (+0.5)
Qwen2.5-7B → Instruct	HumanEval	84.8	**85.4** (+0.6)

C Implementation Details

Training is conducted on Nvidia H100 and H200 GPUs using BFloat16 precision. We set weight decay to 0 and employ a cosine learning rate scheduler with a 0.03 ratio linear warmup. For evaluation, we utilize vLLM (Kwon et al., 2023) to conduct our tests, ensuring efficient and scalable inference. All Adam parameters not listed in Table 12 use their default values, and our Muon experiments use Nesterov-style momentum following Liu et al. (2025a). MetaMathQA augments the GSM8K and MATH training sets; we evaluate on their official test splits.

Computational cost of POME. POME consists of two stages: (1) computing $\Delta W = W_{\text{ft}} - W_{\text{pre}}$ (elementwise subtraction), and (2) performing per-layer SVD, truncation, and reconstruction. Table 11 reports wall-clock time and peak GPU memory for two representative model scales. POME processes each layer independently, so memory scales with the largest single layer rather than the full model; in practice, weights can be offloaded to CPU so that only one layer's ΔW resides on GPU at a time. The total runtime is negligible compared to fine-tuning (typically hours to days).

D Layer Selection Ablation

Table 13 reports the effect of restricting POME to different layer types on LLaMA2-7B (MetaMathQA, seed=87). FFN layers are the primary contributor. Attention-only editing yields limited improvement,

Table 11: Wall-clock time and peak GPU memory for applying POME to all Attention + FFN layers.

Model	Hardware	Wall-clock time	Peak GPU memory
LLaMA2-7B	H100 $\times$ 1	6 min 55 s	28.3 GB $\times$ 1
Qwen2.5-Math-72B	H100 $\times$ 8	33 min 29 s	46.6 GB $\times$ 8

Table 12: The Implementation Details of fine-tuning on LLaMA2-7B, LLaMA3-8B and Gemma2-9B.

Model	Dataset	LR	LR Scheduler	Warmup	Epochs	BS	K
LLaMA2-7B	MetaMathQA	1e-5	cosine	300	3	128	0.5
LLaMA2-7B	commonsense-170k	1e-5	cosine	300	3	32	0.5
LLaMA2-7B	Code-Feedback	1e-5	cosine	300	3	32	0.5
LLaMA3-8B	MetaMathQA	5e-6	cosine	300	3	128	0.5
LLaMA3-8B	commonsense-170k	5e-6	cosine	300	3	32	0.5
LLaMA3-8B	Code-Feedback	5e-6	cosine	300	3	32	0.5
Gemma2-9B	MetaMathQA	5e-6	cosine	300	3	128	0.5

while combining both layer types achieves the best performance, justifying our default strategy of applying POME uniformly to all 2D weight matrices.

Table 13: Layer selection ablation on LLaMA2-7B (MetaMathQA, lr=1e-5, seed=87, $K = 0.5$). POME is applied to the specified layer type only; embeddings, RMSNorm weights, and biases are always left unchanged.

Edited Layers	GSM8K	MATH
None (baseline)	67.2	19.4
Attention only	67.9	19.3
FFN only	69.4	19.7
All (POME)	**69.7**	**19.7**

E Sensitivity Analysis of the Retention Fraction and Global Scale

We examine the retention fraction K, which determines each layer's retained rank k_ℓ, and the global multiplier β. The layer-dependent coefficient α_ℓ is determined analytically by Equation (10) and is not an independent hyperparameter.

The selection of K: In our experiments, we set $r_\ell = \min(m_\ell, d_\ell)$ and retain $k_\ell = \lfloor K r_\ell \rfloor$ singular components. Because the edited layers differ in shape, a common fraction K retains a comparable proportion of singular components in each layer.

From the results, we can observe that POME is not sensitive to the selection of K. For example, in Table 14, we observe that POME achieves consistent improvements when K ranges from 0.25 to 1.0.

The selection of β:

Table 15 varies only the global multiplier β; each α_ℓ then follows from Equation (10).

When the original fine-tuning data are available, β may be selected by inference on a subset of those data, after which all α_ℓ are determined by Equation (10). Table 16 reports training and test accuracies for candidate β values. This data-assisted selection is distinct from checkpoint-only use with a fixed β; training performance need not identify the test-optimal scale.

Table 14: The sensitivity analysis of K on MetaMathQA (mean $\pm$ std over three seeds).

Model	Task	K = 0.25	K = 0.5	K = 0.75	K = 1.0	Baseline
LLaMA2-7B	GSM8K	69.0 ± 0.4	69.5 ± 0.4	69.0 ± 0.4	69.1 ± 0.4	67.0 ± 0.1
LLaMA2-7B	MATH	19.6 ± 0.2	19.8 ± 0.1	19.7 ± 0.2	19.7 ± 0.2	19.1 ± 0.2
Gemma2-9B	GSM8K	82.7 ± 0.4	82.9 ± 0.3	82.8 ± 0.4	82.7 ± 0.4	81.6 ± 0.5
Gemma2-9B	MATH	37.3 ± 0.6	37.5 ± 0.6	37.4 ± 0.6	37.1 ± 0.6	36.6 ± 0.5

Table 15: Sensitivity to the global multiplier β on MetaMathQA ($K = 0.5$).

Model	Task	β = 1.0	β = 1.3	β = 1.5	β = 1.8	β = 2.0	β = 2.3	β = 2.5	β = 3.0
LLaMA2-7B	GSM8K	/	/	/	/	68.9	68.3	68.9	68.2
LLaMA2-7B	MATH	/	/	/	/	19.6	19.2	19.6	19.7
LLaMA3-8B	GSM8K	/	80.6	81.4	81.2	81.4	80.7	/	/
LLaMA3-8B	MATH	/	/	32.7	32.1	32.1	31.4	/	/
Gemma2-9B	GSM8K	83.0	83.4	82.9	82.3	/	/	/	/
Gemma2-9B	MATH	37.3	36.9	36.6	35.9	/	/	/	/

F Additional Comparisons and Robustness Results

Each result below is labeled as a multi-seed result or a single-run point estimate.

Matched post-hoc controls. To test whether other transformations of the same fine-tuning delta reproduce POME's gain, we apply four post-hoc controls to the same LLaMA2-7B checkpoints fine-tuned on MetaMathQA (three seeds), and select each control's hyperparameter with the same training-data protocol used for POME. Task-vector scaling (Ilharco et al., 2022) multiplies ΔW by a global scalar. DARE-style (Yu et al., 2024) drop-and-rescale randomly drops entries of ΔW and rescales the remaining ones. TIES-style (Yadav et al., 2023) trimming keeps only the largest-magnitude entries of ΔW. Spectral clipping caps the largest singular values of each layer's delta. DARE and TIES were designed for merging multiple deltas; here we apply their single-delta operations as matched controls. As shown in Table 17, task-vector scaling improves GSM8K/MATH by 0.3/0.3 points, whereas POME is a further 2.2/0.4 points higher. The DARE- and TIES-style controls remain close to the unedited checkpoint, and spectral clipping leaves it essentially unchanged. Global rescaling, sparsification of the delta, and clipping only the largest singular values therefore do not reproduce POME's gain. Together with Table 7, this identifies equalization of the retained singular directions as the main empirical source of the gain.

Training optimizers. POME depends only on the pretrained and fine-tuned checkpoints and needs no optimizer states. To test whether it remains useful when the training optimizer already shapes the update spectrum, we fine-tune LLaMA2-7B and Gemma2-9B on MetaMathQA with Adam, Muon, and Lion (Chen et al., 2023), and apply POME to each resulting checkpoint (Table 18). All entries are single-run point estimates; the LLaMA2-7B Adam and Muon entries are those of Table 8. POME improves all twelve optimizer–model–task settings, and Muon+POME gives the best result in three

Table 16: Training/test accuracy for selection of the global multiplier β on LLaMA2-7B and Gemma2-9B. Each α_ℓ is computed analytically.

Model	Task	β = 1.0	β = 1.3	β = 1.5	β = 1.8	β = 2.3	β = 2.5	β = 3.0
LLaMA2-7B	GSM8K	/	/	/	/	93.5/68.3	93.6/68.9	93.5/68.2
LLaMA2-7B	MATH	/	/	/	/	53.0/19.2	53.5/19.6	53.6/19.7
Gemma2-9B	GSM8K	95.5/83.0	95.9/83.4	95.9/82.9	95.8/82.3	/	/	/
Gemma2-9B	MATH	62.1/37.3	63.1/36.9	62.9/36.6	62.8/35.9	/	/	/

Table 17: Matched post-hoc controls applied to the same LLaMA2-7B/MetaMathQA fine-tuning deltas (mean ± std over three seeds).

Post-hoc transformation of ΔW	**GSM8K**	**MATH**
Unedited	67.0 ± 0.1	19.1 ± 0.2
Task-vector scaling	67.3 ± 0.3	19.4 ± 0.2
DARE-style drop-and-rescale	67.1 ± 0.3	19.2 ± 0.2
TIES-style magnitude trimming	67.1 ± 0.3	19.3 ± 0.2
Spectral clipping	66.9 ± 0.4	19.1 ± 0.2
POME	$\mathbf{69.5 \pm 0.4}$	$\mathbf{19.8 \pm 0.1}$

of the four model–task settings. The one-shot edit is therefore complementary to Muon's per-step orthogonalization.

Table 18: POME applied to checkpoints trained with different optimizers on MetaMathQA (single-run point estimates). Bold marks the best entry in each row.

Model	**Task**	**Adam**	**+POME**	**Muon**	**+POME**	**Lion**	**+POME**
LLaMA2-7B	GSM8K	66.9	69.0	67.2	**69.4**	67.1	69.2
LLaMA2-7B	MATH	19.0	**19.8**	19.4	19.6	19.2	19.7
Gemma2-9B	GSM8K	81.0	82.6	81.4	**82.8**	81.2	82.7
Gemma2-9B	MATH	36.4	36.9	36.4	**37.2**	36.4	37.0

Multi-seed code and commonsense results. The code and commonsense tables in the main text report seed 87. We reran the Code-Feedback and Commonsense-170k settings with seeds 42, 87, and 100; Table 19 reports the mean ± std of the average score across training seeds.

Table 19: Average scores on code generation and commonsense reasoning (mean ± std over seeds 42, 87, and 100).

Setting	**Model**	**Unedited**	**POME**	**Gain**
Code-Feedback (avg.)	LLaMA2-7B	34.8 ± 0.3	$\mathbf{36.0 \pm 0.5}$	$+1.2$
Code-Feedback (avg.)	LLaMA3-8B	57.9 ± 0.5	$\mathbf{58.9 \pm 0.4}$	$+1.0$
Commonsense-170k (avg.)	LLaMA2-7B	80.5 ± 0.2	$\mathbf{81.5 \pm 0.3}$	$+1.0$
Commonsense-170k (avg.)	LLaMA3-8B	84.9 ± 0.3	$\mathbf{85.6 \pm 0.3}$	$+0.7$

Shared versus layer-dependent budgets. Equation (10) corresponds to the layer-wise optimizer of Theorem G.2 with the induced budgets η_ℓ, which differ across layers. A shared budget η is an alternative cross-layer allocation that equalizes the absolute RMS amplification instead of preserving the relative layer-wise update magnitudes. The theorem holds for any choice of per-layer budgets, so we compare the two allocations empirically. All variants use the same LLaMA2-7B/MetaMathQA checkpoint (seed 87), $K = 0.5$, the same edited attention and FFN matrices, and the same retained singular subspaces; only the cross-layer budget allocation changes. POME uses Equation (10) with one global β selected on a subset of the MetaMathQA training data. The norm-matched shared-η control uses a single budget η for all edited matrices, chosen so that the aggregate Frobenius norm of the edit matches that of POME. The tuned shared-η control rescales this budget by a multiplier selected with the same training-data criterion. The two-group control selects separate multipliers for attention and FFN matrices in the same way. As shown in Table 20, neither shared-η control improves over the unedited checkpoint, the two-group control recovers part of the gain, and POME is best. In this single-checkpoint comparison, preserving the learned relative layer-wise update magnitudes outperforms a uniform RMS budget across heterogeneous layers.

LayerNorm-based architecture. All other models in this paper use RMSNorm. To test a different normalization scheme, we fine-tune the LayerNorm-based Falcon-7B (Almazrouei et al., 2023) on

Table 20: Cross-layer budget allocation on LLaMA2-7B/MetaMathQA (seed 87, $K = 0.5$).

Budget allocation	**GSM8K**	**MATH**
Unedited	67.2	19.4
Shared η, norm-matched	66.9	19.0
Shared η, tuned	67.0	19.4
Two-group η (attention / FFN)	68.7	19.6
POME (layer-wise, Eq. (10))	**69.7**	**19.7**

Alpaca (Taori et al., 2023) and evaluate three commonsense benchmarks (Table 21, single-run point estimates). POME improves all three benchmarks and the average by 1.2 points.

Table 21: POME on the LayerNorm-based Falcon-7B fine-tuned on Alpaca.

Model	**BoolQ**	**OpenBookQA**	**PIQA**	**Avg**
Falcon-7B (Alpaca)	74.6	32.6	79.7	62.3
Falcon-7B (Alpaca) + POME	**75.4**	**34.4**	**80.6**	**63.5**

G Theoretical Analysis

In this section, our theoretical analysis is based on the theory in Muon (Bernstein, 2025).

Problem. Given a layer weight delta $\Delta W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we seek a low-rank linear operator P that preserves the Muon RMS→RMS bound while aligning with ΔW:

$$\max_{P:\text{rank}(P)\le k} \langle \Delta W, P \rangle \quad \text{s.t. } \|P\|_{\text{RMS}\to\text{RMS}} \le \eta. \tag{12}$$

(Here $d_{\text{in}} =$ fan-in, $d_{\text{out}} =$ fan-out.)

Norms. For a vector $x \in \mathbb{R}^d$, the RMS norm is $\|x\|_{\text{RMS}} := \|x\|_2 / \sqrt{d}$. For a matrix $P \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\|P\|_{\text{RMS}\to\text{RMS}} := \max_{x\neq 0} \|Px\|_{\text{RMS}} / \|x\|_{\text{RMS}}$ is the operator norm induced by the RMS norms on $\mathbb{R}^{d_{\text{in}}}$ and $\mathbb{R}^{d_{\text{out}}}$, and $\|P\|_2 := \max_{x\neq 0} \|Px\|_2 / \|x\|_2 = \sigma_1(P)$ is the spectral norm.

Lemma G.1 (RMS → RMS norm)**.** *For any $P \in \mathbb{R}^{d_{out} \times d_{in}}$,*

$$\|P\|_{\text{RMS}\to\text{RMS}} = \sqrt{\frac{d_{in}}{d_{out}}}\ \|P\|_2 .$$

Proof. Applying the definition of the RMS norm in $\mathbb{R}^{d_{\text{out}}}$ and in $\mathbb{R}^{d_{\text{in}}}$,

$$\|P\|_{\text{RMS}\to\text{RMS}} = \max_{x\neq 0} \frac{\|Px\|_{\text{RMS}}}{\|x\|_{\text{RMS}}} = \max_{x\neq 0} \frac{\|Px\|_2 / \sqrt{d_{\text{out}}}}{\|x\|_2 / \sqrt{d_{\text{in}}}} = \sqrt{\tfrac{d_{\text{in}}}{d_{\text{out}}}}\ \max_{x\neq 0} \frac{\|Px\|_2}{\|x\|_2} = \sqrt{\tfrac{d_{\text{in}}}{d_{\text{out}}}}\ \|P\|_2 , \tag{13}$$

where the last equality is the definition of the spectral norm. □

By Lemma G.1, letting

$$\tau \ := \ \eta \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}},$$

Problem (12) is equivalent to

$$\max_{P:\text{rank}(P)\le k} \langle \Delta W, P \rangle \quad \text{s.t. } \|P\|_2 \le \tau. \tag{14}$$

Finally, we can transform the problem in equation 12 to equation 14 and solve the problem about equation 14 in Theorem G.2.

Theorem G.2. *Let the SVD of ΔW be $\Delta W = U\Sigma V^\top$, with singular values $\sigma_1 \geq \cdots$ and singular vectors $\{u_i, v_i\}$. Consider, for $1 \leq k \leq \min(d_{\text{out}}, d_{\text{in}})$, the problem*

$$\max_P \langle \Delta W, P \rangle \quad \textit{s.t.} \ \|P\|_2 \leq \tau, \ \ \operatorname{rank}(P) \leq k. \tag{15}$$

Let $U_k = [u_1, \ldots, u_k]$ and $V_k = [v_1, \ldots, v_k]$. Then an optimizer is

$$P^\star \ = \ \tau \sum_{i=1}^{k} u_i v_i^\top \ = \ \eta \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}\, U_k V_k^\top, \tag{16}$$

and the optimal value equals

$$\max_{\|P\|_2 \leq \tau, \ \operatorname{rank}(P) \leq k} \langle \Delta W, P \rangle \ = \ \tau \sum_{i=1}^{k} \sigma_i(\Delta W). \tag{17}$$

If $\sigma_k > \sigma_{k+1}$, the solution is unique up to sign flips of $\{u_i, v_i\}_{i \leq k}$; if there are ties at the k-th singular value, any orthonormal basis of the top-k singular subspace is optimal.

Proof. Upper bound. Let P be feasible for (15). By von Neumann's trace inequality (von Neumann, 1937; Mirsky, 1975),

$$\big|\langle \Delta W, P \rangle\big| = \big|\operatorname{tr}(\Delta W^\top P)\big| \ \leq \ \sum_{i \geq 1} \sigma_i(\Delta W)\, \sigma_i(P),$$

where singular values are sorted in non-increasing order. (The inequality is usually stated for square matrices; it applies here after padding ΔW and P with zeros to a common square size, which changes neither the trace nor the nonzero singular values.) Since $\operatorname{rank}(P) \leq k$, we have $\sigma_i(P) = 0$ for $i > k$. Also $\sigma_1(P) = \|P\|_2 \leq \tau$, hence $\sigma_i(P) \leq \tau$ for all i. Therefore

$$\langle \Delta W, P \rangle \ \leq \ \big|\langle \Delta W, P \rangle\big| \ \leq \ \sum_{i=1}^{k} \sigma_i(\Delta W)\, \sigma_i(P) \ \leq \ \tau \sum_{i=1}^{k} \sigma_i(\Delta W). \tag{18}$$

Attainment. The matrix $P^\star = \tau U_k V_k^\top$ is feasible: since U_k and V_k have orthonormal columns, $P^\star = U_k(\tau I_k)V_k^\top$ is a compact SVD whose singular values all equal τ, so $\operatorname{rank}(P^\star) \leq k$ and $\|P^\star\|_2 = \tau$. Moreover, since $\Delta W v_i = \sigma_i(\Delta W)\, u_i$ and $u_i^\top u_i = 1$,

$$\langle \Delta W, P^\star \rangle = \tau \sum_{i=1}^{k} \operatorname{tr}\big(\Delta W^\top u_i v_i^\top\big) = \tau \sum_{i=1}^{k} u_i^\top \Delta W v_i = \tau \sum_{i=1}^{k} \sigma_i(\Delta W).$$

Hence $P^\star$ attains the upper bound in (18), so it is an optimizer and the optimal value is given by (17). If $\sigma_k > \sigma_{k+1}$, the top-k singular subspace is unique, so $P^\star$ is unique up to sign flips of the pairs (u_i, v_i); under ties, any orthonormal basis of the top-k subspace yields an optimizer.

Finally, rewriting τ via Lemma G.1 (namely $\tau = \eta\sqrt{d_{\text{out}}/d_{\text{in}}}$) gives the stated form (16), which completes the proof. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state that POME is a post-optimization, training-free method that applies truncated SVD and spectrum equalization to weight deltas. All claimed improvements are supported by the experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the scope of the evaluation and sensitivity to the global multiplier β, with supporting analysis in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theorem G.2 and the supporting lemma are stated and proved in Appendix G. Section 3.2 defines the relevant norms and connects the closed-form optimizer to the implemented edit.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3 presents the POME algorithm, including retained ranks and layer-wise norm matching, and Appendix C reports model, dataset, and training configurations. The global-scale selection protocol is described in Section 3.2, with sensitivity results in Appendix E. The base models and training datasets are publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and instructions for reproducing the main experimental results are publicly available at https://github.com/NUS-HPC-AI-Lab/POME. The experiments use publicly available base models and training datasets; Appendix C reports implementation details.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Table 12 in Appendix C reports the learning rate, scheduler, warmup, batch size, epochs, and retention fraction K for the listed controlled fine-tuning configurations. Evaluation uses standard benchmark splits via vLLM with greedy decoding, and the global-scale selection protocol is described in Section 3.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 1 reports mean$\pm$std over training seeds 42, 87, and 100 for three MetaMathQA-fine-tuned models evaluated on GSM8K and MATH; the standard deviations

capture variation across training seeds. Appendix A reports all 18 paired model–benchmark–seed comparisons. Appendices E and F also report multi-seed retention-fraction sensitivity, matched-control, code-generation, and commonsense results. Single-run and third-party checkpoint results are interpreted as point estimates.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C specifies that training uses Nvidia H100/H200 GPUs with BFloat16 precision. Table 11 reports wall-clock time and peak GPU memory for applying POME at two scales (LLaMA2-7B: 7 min on 1×H100; Qwen2.5-Math-72B: 33 min on 8×H100).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research involves only publicly available models and datasets and does not raise any ethical concerns identified in the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: POME is a generic post-processing technique for improving fine-tuned model quality. It does not introduce new capabilities that create direct negative societal impacts beyond those already present in large language models. No specific broader impact discussion is required.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new models, datasets, or other assets with high misuse risk. POME is a lightweight post-processing utility that inherits the risk profile of the underlying models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models (LLaMA-2, LLaMA-3, Gemma-2, Qwen2.5) and datasets (MetaMathQA, commonsense-170k, Code-Feedback) are cited with their original papers. All are publicly available under licenses permitting academic research use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets, benchmarks, or model checkpoints as assets. The POME algorithm is the primary contribution, which is fully described in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the *subject* of study (models being fine-tuned and post-processed), not a component of the research methodology itself. No LLMs were used in designing the method, running experiments, or analyzing results.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.