POME: POST OPTIMIZATION MODEL EDIT VIA MUON-STYLE PROJECTION

Anonymous authors
Paper under double-blind review

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, which often represent noise. As a simple post-processing step, POME is completely decoupled from the training pipeline. It requires zero modifications and imposes no overhead, making it universally compatible with any optimizer or distributed framework. 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 any fine-tuning pipeline.

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 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 noise from extremely high or low singular values. Crucially, this simple post-processing step requires no changes to training pipelines and remains compatible with any optimizer or distributed framework.

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., 2024), 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 alternative 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

The enormous computational costs of large models make updating their knowledge far from straightforward. Ideally, as the world rapidly evolves in complex ways, a large model should keep pace in real time. Yet the heavy computing required to train a brand-new model makes instant updating infeasible. Consequently, a new paradigm "Model Edit" has emerged, enabling targeted modifications to a model's knowledge within specific domains while avoiding adverse effects on its behavior for other inputs. (Yao et al., 2023) Model-editing 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; Min, 2016; Li et al., 2025). **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. In addition, model soup (Wortsman et al., 2022), ensemble learning (Dietterich et al., 2002), or EMA is another paradigm for post optimization model editing, but it relies on training multiple models.

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.

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, the Newton-Schulz function is an iterative algorithm that approximates the closest semi-orthogonal matrix to M_t, and the final update U_t represents a modified version of M_t, where all singular values of M_t are normalized to 1.

This approach ensures that, regardless of the original magnitude of the singular values, each independent direction receives equal scaling. This adjustment prevents the overlooking of critical directions during training, thereby enhancing the efficiency of parameter space exploration.

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

This section introduces our post-training orthogonalization approach. We first establish the problem setup and theoretical motivation in Section 3.1, then present our core method including SVD decomposition, rank truncation, and orthogonalization in Section 3.2. Finally, we analyze which layers benefit most from post-training orthogonalization through systematic layer-wise evaluation in Section 3.3.

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 *data-free* post-hoc transformation of the fine-tuning update aimed at improving evaluation performance while controlling drift.

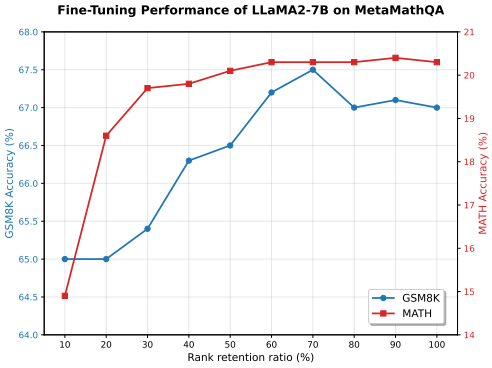

Figure 1: Effect of rank retention ratio on POME performance for LLaMA2-7B fine-tuned on MetaMathQA.

Our motivation builds on Muon (Sec. 2.2). Muon improves stability and generalization by equalizing the singular values of step wise updates, which shapes the update subspace during 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 finetuned model and truncate the trailing singular values while keep 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 selected layer, decomposes the update by singular value decomposition, keeps a compact subspace to suppress noise, 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 Our Method: POME

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

$$\|\Delta W\|_{\text{RMS}\to\text{RMS}} := \max_{x\neq 0} \frac{\|\Delta W x\|_{\text{RMS}}}{\|x\|_{\text{RMS}}} = \sqrt{\frac{\text{fan-in}}{\text{fan-out}}}\,\|\Delta W\|_*, \tag{7}$$

where $\|\cdot\|_*$ represents the spectral norm. Therefore, for the post-training edit, we define an optimization problem to stay close to the original weight matrix while ensuring the RMS-RMS norm is bounded by η (Bernstein, 2025):

Definition: Let ΔW be the layer-wise weight matrix. We hope to find a matrix P, which can preserve RMS-RMS norm as Muon while being low-rank:

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

where $\langle\cdot,\cdot\rangle$ represents the Frobenius inner product. The closed form solution can be written as

$$P^* = \eta\sqrt{\frac{\text{fan-out}}{\text{fan-in}}} U_k V_k^T,$$

where fan-out and fan-in represent 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 introduction of the low-rank constraint, which is important based on our empirical results.

The closed-form solution naturally suggests a practical algorithm: perform SVD decomposition on 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 (per selected layer). 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 $\Sigma = \text{diag}(\sigma_1 \geq \cdots \geq \sigma_r \geq 0)$.

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

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

Step 3: Orthogonalization and Reconstruction. Construct the orthogonalized delta by setting all retained singular values to unity and, optionally, apply a global scale α:

$$\widehat{\Delta W}_\perp = U_k\,I_k\,V_k^\top \;=\; U_k\,V_k^\top, \quad \widehat{\Delta W} = \alpha\,\widehat{\Delta W}_\perp, \tag{10}$$

where I_k is the $k\times k$ identity, k can be set to $0.5\cdot\text{rank}(\Delta W)$ since Figure 1 indicates that retaining the top 50% singular components recovers nearly all performance across most datasets, and α is

selected on a small held-out validation set to optimize performance. We provide the sensitivity analysis of k and α in sec. A.2.

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

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

This simple post-processing procedure requires no modifications to the training process and can be applied to any fine-tuned model, making it highly practical for real-world deployment.

3.3 Layer Selection

Given the data-free editing setup in Sec. 3.1 and our proposed edit method in Sec. 3.2, a practical question is *which layers to edit*. We therefore conduct a controlled, per–layer-type study to quantify sensitivity.

We evaluate on LLaMA2-7B, LLaMA3-8B and Gemma2-9B fine-tuned on MetaMathQA dataset. Only the target layer type is edited, and all others remain unchanged. We report accuracy deltas relative to the fine-tuned baseline.

Model / Task	Q_{proj}	K_{proj}	V_{proj}	O_{proj}	$Gate_{\text{proj}}$	Up_{proj}	$Down_{\text{proj}}$	Baseline
LLaMA2-7B/GSM8K	67.1	68.2	67.9	68.1	**68.9**	**69.7**	68.2	67.2
LLaMA2-7B/MATH	19.7	19.3	19.4	20.0	**19.8**	**19.7**	19.7	19.4
Average	43.4	43.8	43.7	44.1	**44.4**	**44.7**	44.0	43.3
LLaMA3-8B/GSM8K	80.3	81.3	80.4	80.8	**81.0**	**81.4**	80.1	80.3
LLaMA3-8B/MATH	32.3	31.5	31.5	32.0	**31.9**	**32.7**	31.6	31.5
Average	56.3	56.4	56.0	56.4	**56.5**	**57.1**	55.9	55.9
Gemma2-9B/GSM8K	81.7	82.4	82.7	82.7	82.7	**83.3**	**82.8**	82.2
Gemma2-9B/MATH	36.0	36.7	36.8	37.2	37.3	**37.3**	**38.9**	36.1
Average	58.9	59.6	59.8	60.0	60.0	**60.3**	**60.9**	59.2

Table 1: Layer-wise post-training orthogonalization on LLaMA2-7B (MetaMathQA, lr=1e-5, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$), LLaMA3-8B (MetaMathQA, lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$) and Gemma2-9B (MetaMathQA, lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$). Each column edits only the specified layer type and others remain unchanged.

Table 1 reports the layer-wise interventions. Two themes emerge consistently across tasks. First, attention projections ($Q_{\text{proj}}, K_{\text{proj}}, V_{\text{proj}}, O_{\text{proj}}$) are largely insensitive to post-hoc orthogonalization compared with FFN layers, suggesting their deltas are already well-conditioned. Second, the *FFN expansion* maps (Up_{proj}, $\text{Down}_{\text{proj}}$ and $\text{Gate}_{\text{proj}}$) benefit the most, aligning with our observation that high-dimensional expansions concentrate update energy into a few directions that orthogonalization can rebalance. Guided by these findings, our main experiments default to editing FFN expansion layers—using Up_{proj} as the primary target.

4 Experiments

We conduct comprehensive evaluations of POME across multiple experimental settings. We first evaluate supervised fine-tuning on mathematical reasoning, code generation, and commonsense reasoning tasks to establish baseline effectiveness. We then demonstrate the method's broader applicability by applying POME to publicly available fine-tuned models and models optimized with reinforcement learning from human feedback (RLHF). Finally, we provide detailed ablation studies to understand the contribution of key components.

Algorithm 1 POME

Require: Base weights W_{pre}, delta ΔW, scaling factor α
Ensure: Edited model weights W_E
1: **SVD decomposition**:
2: $\Delta W = U \Sigma V^\top$
3: **Rank truncation**:
4: $U_k = U[:, 1{:}k], \quad V_k = V[:, 1{:}k],$
5: **Orthogonalization and Reconstruction**:
6: $\widehat{\Delta W}_\perp = U_k I_k V_k^\top = U_k V_k^\top, \quad \widehat{\Delta W} = \alpha \widehat{\Delta W}_\perp$
7: **Final edited model**:
8: $W_E = W_{\text{pre}} + \widehat{\Delta W}$
9: **return** W_E

4.1 Experimental Setting

Mathematical Reasoning. We fine-tune LLaMA2-7B (Touvron et al., 2023), LLaMA3-8B (Dubey 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., 2024).

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)).

To ensure robustness, we conduct multiple runs with different random seeds and report averaged results where applicable.

4.2 Mathematical Reasoning

We evaluate POME on mathematical reasoning tasks by fine-tuning three model families on MetaMathQA. To ensure robustness, we train each model with three different random seeds and analyze performance across runs.

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)**

Table 2: Accuracy of Fine-Tuning LLaMA2-7b, LLaMA3-8B and Gemma2-9B on MetaMathQA. Adam_{42} represents the results with seed=42 and Adam optimizer.

Table 2 illustrates that POME achieves consistent improvements across models and seeds. For instance, on LLaMA2-7B with seed 87, POME improves GSM8K accuracy from 67.2% to 69.7%

and MATH accuracy from 19.4% to 19.7%. Similar gains are observed across LLaMA3-8B and Gemma2-9B, demonstrating the method's generalizability.

To further validate our approach beyond the models trained by ourselves, we apply POME to publicly available fine-tuned models. Table 3 demonstrates that POME improves MetaMath-Mistral-7B [1] performance from 77.9% to 79.5% on GSM8K and from 28.1% to 28.5% on MATH, confirming effectiveness on externally trained checkpoints.

Model	Adam	+POME	GSM8K	MATH	Average
MetaMath-Mistral-7B	✓		77.9	28.1	53.0
MetaMath-Mistral-7B	✓	✓	**79.5(↑ 1.6)**	**28.5(↑ 0.4)**	**54.0(↑ 1.0)**

Table 3: Effect of POME on the publicly available MetaMath-Mistral-7B model, demonstrating generalization to externally trained checkpoints.

4.3 Code Generation

We assess POME's effectiveness on code synthesis tasks using two training datasets. Table 4 presents results for models trained on Code-Feedback (Zheng et al., 2024) and Magicoder-Evol-Instruct-110K (Wei et al., 2023) datasets.

On Code-Feedback, POME consistently improves performance across all evaluation metrics. For LLaMA2-7B, we observe gains of 0.6% on HumanEval, 1.2% on HumanEval+, 0.8% on MBPP, and 1.3% on MBPP+, yielding a 1.0% average improvement. On Magicoder-Evol-Instruct-110K, POME achieves more substantial gains, with a 1.7% average improvement across metrics.

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**)

Table 4: Accuracy of Fine-Tuning LLaMA2-7B and LLaMA3-8B (lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$) on Code Generation Task.

4.4 Commonsense Reasoning

We evaluate POME on commonsense reasoning by fine-tuning LLaMA2-7B and LLaMA3-8B on Commonsense-170k and testing across eight downstream tasks. Table 5 presents comprehensive results across diverse reasoning benchmarks.

Model	Method	BoolQ	PIQA	SIQA	HellaS	WinoG	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**)

Table 5: Accuracy of Fine-Tuning LLaMA2-7b and LLaMA3-8B (lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$) on Commonsense-170k.

[1] https://huggingface.co/meta-math/MetaMath-Mistral-7B

POME achieves consistent improvements across both model scales. For LLaMA2-7B, we observe notable improvements on HellaS (+2.2%) and ARC-E (+1.8%), yielding an overall average improvement of 0.8%. Similarly, LLaMA3-8B shows improvements on all eight benchmarks, with the largest gain on ARC-C (+3.1%) and an average improvement of 1.0%. The consistent improvements across diverse reasoning tasks demonstrate POME's effectiveness in enhancing response for commonsense inference.

4.5 RLHF MODEL

To investigate POME's applicability beyond supervised fine-tuning, we evaluate on models optimized with reinforcement learning from human feedback (RLHF). We apply POME to Qwen-2.5-Math models by computing weight deltas relative to their base checkpoints.

Table 6 demonstrates that POME improves performance on 6 out of 7 tasks for the 7B model and 5 out of 7 tasks for the 72B model, confirming effectiveness across model scales and optimization paradigms. For Qwen2.5-Math-7B-Instruct [2], POME increases average performance from 62.7% to 63.4%.

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**)

Table 6: Performance of Qwen2.5-Math-72B-Instruct models with and without POME across mathematical reasoning benchmarks.

Additionally, we evaluate POME on recent variants of GRPO (Shao et al., 2024), specifically Dr.GRPO [3] (Liu et al., 2025c) models. Table 7 shows that POME consistently improves performance across mathematical reasoning benchmarks. For Dr.GRPO, we observe notable improvements on AIME (43.3% → 46.7%) and AMC (59.0% → 66.3%), with gains also observed on Olympiad Bench. The average performance increases from 51.2% to 53.3%, demonstrating POME's effectiveness on advanced RL-optimized models.

Model	AIME24	AMC	MATH500	Minerva	OlympiadBench	Average
Dr.GRPO	43.3	59.0	**80.2**	**31.6**	41.9	51.2
Dr.GRPO + POME	**46.7**	**66.3**	79.6	31.3	**42.5**	**53.3**(↑ **2.1**)

Table 7: Accuracy comparison between baseline Dr.GRPO and Dr.GRPO with our proposed POME method on Qwen2.5-Math.

4.6 GENERALIZATION ANALYSIS

To understand why POME improves model performance, we analyze its effect on generalization by examining both training accuracy and test accuracy. Table 8 presents results for LLaMA3-8B and Gemma2-9B trained on MetaMathQA datasets.

Our analysis reveals that POME consistently improves test accuracy *almost* without affecting training accuracy, suggesting that the method enhances generalization rather than simply overfitting to training data. For MetaMathQA, while training accuracy remains comparable between baseline and POME-edited models, test accuracy shows notable improvements. These findings suggest that

[2] https://huggingface.co/Qwen/Qwen2.5-Math-7B-Instruct
[3] https://huggingface.co/sail/Qwen2.5-Math-7B-Oat-Zero

Model	Task	POME	Training Accuracy	Test Accuracy
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)

Table 8: Generalization analysis of POME on LLaMA3-8B and Gemma2-9B fine-tuned on MetaMathQA (lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$). Training accuracy and test accuracy demonstrate the method's effect on model generalization.

POME's effectiveness stems from improved parameter space utilization rather than memorization of training examples, providing insight into the mechanism underlying the observed performance gains.

4.7 Ablation Study

Rank Truncation and Equalization Analysis. We examine the effect of retaining only top-k singular components versus using all components (Truncation) and setting all retained singular values to unity (Equalization). Table 9 shows that while equalization yields improvements (e.g., 67.2% → 69.0% on GSM8K for LLaMA2-7B), rank truncation provides additional gains (67.2% → 69.7%), confirming that noise suppression contributes to performance.

Model	Truncation	Equalization	GSM8K	MATH	Average
LLaMA2-7B			67.2	19.4	43.3
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		✓	81.0	32.1	**56.6(↑ 0.7)**
LLaMA3-8B	✓	✓	81.4	32.7	**57.1(↑ 1.2)**

Table 9: Ablation study on the effects of truncation and POME on LLaMA2-7B (MetaMathQA, lr=1e-5, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$) and LLaMA3-8B (MetaMathQA, lr=5e-6, seed=87, $k = 0.5 \cdot \text{rank}(\Delta W)$). The checkmarks indicate: Truncation (selecting top-50% principal components) and Equalization.

Comparison with Baseline Methods. Table 10 compares POME against training-time alternatives including NEFTune (Jain et al., 2023) and Exponential Moving Average (EMA). NEFTune achieves modest improvements across both models and tasks, while EMA shows mixed results with some performance degradation on certain benchmarks. POME consistently outperforms both methods, achieving the largest improvements across all settings. The limited effectiveness of EMA can be attributed to the similarity of models from the same training trajectory, providing minimal additional information for ensemble averaging. NEFTune's modest gains suggest that noise injection during training provides some regularization benefits, but lacks the targeted geometric correction that POME provides. In contrast, POME's superior performance stems from its ability to reshape the learned parameter subspace post-training, addressing directional imbalances that accumulate during optimization without requiring training-time modifications or hyperparameter tuning.

Model	Task	Adam	NEFTune	EMA	POME
LLaMA2-7B	**GSM8K**	67.2	67.7	67.5	**69.7**
LLaMA2-7B	**MATH**	19.4	**19.7**	18.8	**19.7**
LLaMA3-8B	**GSM8K**	80.3	80.9	80.3	**81.4**
LLaMA3-8B	**MATH**	31.5	31.8	31.5	**32.7**

Table 10: Comparison between POME and baseline methods on LLaMA2-7B and LLaMA3-8B.

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. Our ablation studies confirm the method's robustness and identify FFN expansion layers as optimal targets for orthogonalization. POME's simplicity and broad applicability establish post-hoc subspace shaping as a practical alternative to matrix-aware training methods.

6 Usage of Large Language Models

We declare that all core contributions of this work, including the research ideas, methodology design, experimental setup, and result analysis, were conceived and executed entirely by the authors through our own thinking and discussions. Large language models were used solely for language polishing and improving the clarity of our written presentation. Specifically, we used LLMs to refine sentence structure, grammar, and word choice in our manuscript, but no LLM-generated content was used for the technical substance, experimental design, or scientific insights presented in this paper.

References

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 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.

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 et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

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

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.

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.

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, You Jiacheng, 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, 2024.

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, 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.

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.

Byeong June Min. Application of monte carlo simulations to improve basketball shooting strategy. *Journal of the Korean Physical Society*, 69(7):1139–1143, 2016.

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.

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.

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.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*, 3, 2023.

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.

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.

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.

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 APPENDIX

A.1 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.

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

Table 11: The Implementation Details of fine-tuning on LLaMA2-7B, LLaMA3-8B and Gemma2-9B.

A.2 THE SENSITIVITY ANALYSIS OF K AND α

In this section, we provide the sensitivity analysis of k (top-k components) and scale factor α.

The selection of K: In our experiments, we define $k = K \cdot \text{rank}(\Delta W)$.

Model	Task	K = 0.25	K = 0.5	K = 0.75	K = 1.0	Baseline
LLaMA2-7B	GSM8K	69.7	69.7	69.1	69.2	67.2
LLaMA2-7B	MATH	19.9	19.7	19.6	19.7	19.4
Gemma2-9B	GSM8K	83.2	83.3	83.5	83.9	82.2
Gemma2-9B	MATH	37.8	37.3	37.3	36.9	36.1

Table 12: The sensitivity analysis of k on MetaMathQA.

From the results, we can observe that POME is not sensitive to the selection of k. For example, in table 12, We observe that POME achieves consistent improvements when K ranges from 0.25 to 0.75.

The selection of α:

In our experiments, we first normalize the weight matrix and then apply a scaling factor β to the weights:

$$\alpha = \beta * \|\Delta W\| \frac{\widehat{\Delta W}_{\perp}}{\|\widehat{\Delta W}_{\perp}\|} \tag{12}$$

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	/	/	/	/

Table 13: The sensitivity analysis of α on MetaMathQA ($k = 0.5 \cdot \text{rank}(\Delta W)$).

A.3 The selection of α

In this section, we investigate how to determine α (defined in Equation 12) using training data. As shown in Table 14, the optimal α consistently corresponds to strong training performance. Therefore, we can simplify the selection process by using training performance as a metric to determine α.

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 14: The analysis (Train/Test) of α selection on LLaMA2-7B and Gemma2-9B.

A.4 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:

$$\max_{P:\text{rank}(P)\leq k} \langle \Delta W, P \rangle \quad \text{s.t. } \|P\|_{\text{RMS}\to\text{RMS}} \leq \eta. \tag{13}$$

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

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

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

where $\|\cdot\|_*$ is the spectral norm.

Proof. By definition, we can obtain:

$$\|P\|_{\text{RMS}} = \frac{1}{d} \|P\|_2 . \tag{14}$$

Then,

$$\|P\|_{\text{RMS}\to\text{RMS}} = \max_{x\neq 0} \frac{\|Px\|_{\text{RMS}}}{\|x\|_{\text{RMS}}} = \max_{x\neq 0} \frac{\|Px\|_2 / \sqrt{d_{\text{out}}}}{\|x\|_2 / \sqrt{d_{\text{in}}}} = \sqrt{\tfrac{d_{\text{in}}}{d_{\text{out}}}} \max_{x\neq 0} \frac{\|Px\|_2}{\|x\|_2} = \sqrt{\tfrac{d_{\text{in}}}{d_{\text{out}}}}\ \|P\|_* , \tag{15}$$

where $\|\cdot\|_*$ represents the spectral norm. By Lemma 1, letting

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

Problem (13) is equivalent to

$$\max_{P:\text{rank}(P)\le k} \langle \Delta W, P\rangle \quad \text{s.t. } \|P\|_* \le \tau. \tag{16}$$

Finally, we can transform the problem in equation 13 to equation 16 and solve the problem about equation 16 in Theorem 1.

Theorem 1. *Let the SVD of ΔW be $\Delta W = U\Sigma V^\top$, with singular values $\sigma_1 \ge \cdots$ and singular vectors $\{u_i, v_i\}$. Consider the problem*

$$\max_P \langle \Delta W, P\rangle \quad \textit{s.t. } \|P\|_* \le \tau,\ \text{rank}(P) \le k. \tag{17}$$

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

$$P^\star = \tau \sum_{i=1}^{k} u_i v_i^\top = \eta\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}\, U_k V_k^\top, \tag{18}$$

and the optimal value equals

$$\max_{\|P\|_2\le\tau,\ \text{rank}(P)\le k} \langle \Delta W, P\rangle = \tau \sum_{i=1}^{k} \sigma_i(\Delta W). \tag{19}$$

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

Proof. The feasible set $\{P : \|P\|_* \le \tau,\ \text{rank}(P) \le k\}$ is compact and nonempty, and the objective is linear, so a maximizer exists. Write the SVD $\Delta W = U\Sigma V^\top$ and set $Q := U^\top P V$. By unitary invariance of the inner product,

$$\langle \Delta W, P\rangle = \text{tr}(\Delta W^\top P) = \text{tr}(\Sigma Q).$$

By von Neumann's trace inequality (equivalently, Ky Fan's variational principle),

$$\text{tr}(\Sigma Q) \le \sum_{i\ge 1} \sigma_i(\Delta W)\, \sigma_i(Q).$$

Since $\text{rank}(Q) \le \text{rank}(P) \le k$, we have $\sigma_i(Q) = 0$ for $i > k$. Also $\|Q\|_* = \|P\|_* \le \tau$, hence $\sigma_i(Q) \le \tau$ for all i. Therefore

$$\langle \Delta W, P\rangle = \sum_{i=1}^{k} \sigma_i(\Delta W)\, \sigma_i(Q) \le \tau \sum_{i=1}^{k} \sigma_i(\Delta W). \tag{20}$$

The bound is tight when Q aligns with the top-k singular directions of ΔW and saturates the spectral-norm budget, i.e., $Q^\star = \tau\,\text{diag}(1, \ldots, 1, 0, \ldots)$ in the (U, V) basis. Mapping back gives the maximizer

$$P^\star = UQ^\star V^\top = \tau \sum_{i=1}^{k} u_i v_i^\top, \tag{21}$$

which attains the value in equation 19. 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 1 (namely $\tau = \eta\sqrt{d_{\text{out}}/d_{\text{in}}}$) gives the stated form equation 18 and finish the proof.