NEGLIGIBLE IN SIZE, SIGNIFICANT IN EFFECT: ON SCALE VECTORS IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Normalization layers in modern large language models (LLMs) consist of a deterministic normalization operation and a learnable scale vector. While the normalization operation has been extensively studied, the scale vector remains poorly understood despite its ubiquitous use. In this work, we present a systematic study of scale vectors in LLMs from the perspectives of expressivity, optimization, and architectural structure. *First*, we show empirically that although scale vectors constitute only a negligible fraction of model parameters, removing them substantially degrades LLM pre-training. Our theory further shows that, in Pre-Norm architectures, scale vectors do not increase expressivity; instead, they improve optimization through a self-amplifying preconditioning effect on subsequent linear mappings. *Second*, we investigate the role of weight decay for scale vectors. By distinguishing Input-Norm and Output-Norm layers, we theoretically show that weight decay is beneficial for the former but harmful for the latter, due to their distinct roles in optimization and expressivity. *Third*, motivated by this understanding, we propose three lightweight and complementary improvements to scale vectors: branch-specific heterogeneity, improved placement around linear mappings, and magnitude-direction reparameterization. Both theory and experiments show that each improvement yields consistent gains. *Finally*, we combine these improvements into a unified scale-vector strategy and evaluate it through extensive LLM pre-training experiments on dense and mixture-of-experts models ranging from 0.12B to 2B parameters, across multiple optimizers and learning rate schedules, under industrial-scale token budgets. The unified strategy consistently achieves lower terminal loss than well-tuned baselines and exhibits more favorable scaling behavior, while adding negligible parameter and computational overhead.

1 INTRODUCTION

Normalization layers are a fundamental component of modern deep learning and are crucial for the stable and efficient training of large language models (LLMs) (Vaswani et al., 2017). In modern LLM architectures, normalization is typically implemented by RMSNorm (Zhang & Sennrich, 2019), whose core structure consists of a deterministic normalization operation followed by a *learnable scale vector* $\boldsymbol{\gamma}$. This simple normalize-then-scale structure has remained largely unchanged across modern LLMs.

The normalization operation itself has been extensively studied across deep learning. Compared with BatchNorm (BN) in computer vision (Ioffe & Szegedy, 2015) and LayerNorm (LN) in sequence modeling (Ba et al., 2016), RMSNorm (Zhang & Sennrich, 2019) modifies the normalization operation to improve simplicity and training stability in LLMs. However, its accompanying scale vector has largely retained its original form and received substantially less attention.

While the normalization operation is widely understood to stabilize training, the role of scale vectors remains unclear. Since scale vectors constitute only a negligible fraction of the total parameters, they are often treated as minor architectural details. However, do they truly have a negligible effect on LLM training? Answering this question is challenging, as it requires analyzing scale vectors jointly with other Transformer components from both expressivity and optimization perspectives.

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

In this work, we systematically investigate scale vectors in LLMs. We provide theoretical understanding of their mechanisms and propose scalable variants that improve LLM pre-training. **Our contributions** are summarized as follows:

- **Necessity of scale vectors.** (Section 2.1) We empirically show that scale vectors substantially affect LLM pre-training despite their negligible parameter count. Our theoretical analysis further reveals that: (i) counterintuitively, scale vectors *do not increase expressivity* in Pre-Norm architectures; (ii) their benefit arises from a *self-amplifying preconditioning* effect on the subsequent linear mappings, which accelerates the training dynamics.
- **Weight decay of scale vectors.** (Section 2.2) We study an unresolved practical question: whether weight decay (wd) should be applied to scale vectors. We classify normalization layers into *Input-Norm* and *Output-Norm* according to whether their scale vectors are immediately followed by linear maps. Our theoretical analysis shows that these two types have distinct mechanisms: (i) for Input-Norm scale vectors, including those in Pre-Norm layers, wd is *beneficial* as it induces balanced dynamics and controls Hessian sharpness, thereby accelerating and stabilizing training; (ii) for Output-Norm scale vectors, wd is *harmful* because these scale vectors determine expressivity, which wd can undesirably restrict. LLM experiments validate these theoretical insights.
- **Improving scale vectors.** (Section 3) The above understanding motivates three complementary improvements to scale vectors, each with *minimal* computational and parameter overhead. (i) **Heterogeneity.** In Transformer blocks such as self-attention, one shared Pre-Norm feeds multiple projections, including query, key, and value; however, these branches exhibit distinct training dynamics. We therefore introduce branch-specific scale vectors to provide *tailored* self-amplifying preconditioners for different branches. (ii) **Placement.** In Pre-Norm architectures, scale vectors are consistently applied on the input side of the subsequent linear maps, inducing only row-wise self-amplifying preconditioning. We propose new placements that provably provide *both row-wise and column-wise* preconditioning, further accelerate training. (iii) **Reparameterization.** We propose *magnitude-direction reparameterizations* of scale vectors. Our theoretical analysis shows that these reparameterizations induce more anisotropic preconditioner and further accelerate training.
- **Extensive experiments.** (Section 4) (i) We conduct systematic pre-training experiments on 0.12B Llama models to evaluate *each strategy* and its variants, including weight decay, heterogeneity, placement, and reparameterization (Sections 2.2, 3.1, 3.2, 3.3). Each strategy yields clear gains. (ii) We then evaluate a *unified scale-vector strategy* that combines these improvements. We conduct extensive LLM pre-training experiments on both dense and mixture-of-experts (MoE) models, with sizes ranging from 0.12B to 2B parameters, trained on the high-quality FineWeb-Edu dataset under industrial-scale token budgets. We further evaluate different optimizers, including AdamW and Muon, and different learning rate schedules. Across all settings, our unified strategy *consistently achieves lower terminal loss* and *exhibits more favorable scaling behavior* than well-tuned baselines, demonstrating its potential for improved scalability to larger models.

2 Understanding Scale Vectors

Notations. We denote the Hadamard product by $\odot$. For $\boldsymbol{x} \in \mathbb{R}^d$, let $\mathsf{Norm}(\boldsymbol{x}) = \sqrt{d}\boldsymbol{x}/\|\boldsymbol{x}\|$. For vectors, $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ denote the Euclidean inner product and norm, respectively. For matrices, $\lambda_{\min}(\cdot)$, $\lambda_{\max}(\cdot)$, $\mathrm{Tr}(\cdot)$, and $\| \cdot \|_{\mathrm{F}}$ denote the smallest eigenvalue, largest eigenvalue, trace, and Frobenius norm, respectively. For $\boldsymbol{v} \in \mathbb{R}^d$, $\mathrm{diag}(\boldsymbol{v}) \in \mathbb{R}^{d \times d}$ denotes the diagonal matrix with diagonal $\boldsymbol{v}$. We use $\mathcal{O}(\cdot)$ to hide problem-independent constants. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. For a time-dependent quantity $h(t)$, write $\dot{h} := \mathrm{d}h/\mathrm{d}t$.

We focus on RMSNorm, which is widely used in modern LLMs:

$$\mathsf{RMSNorm}(\boldsymbol{x}; \boldsymbol{\gamma}) = \boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

Since BN and LN also contain scale vectors, similar mechanisms may apply beyond RMSNorm.

2.1 Necessity of Scale Vectors

Negligible in model size. Most Transformer parameters are matrix weights in the feedforward network (FFN) and self-attention (Attn) blocks, each of size $\mathcal{O}(D^2)$, where D is the hidden dimension.

In contrast, each scale vector has only D parameters. For example, in the Llama-1B model considered in Section 4, the model has 1,028,065,024 parameters in total, whereas all scale vectors together contain only 80,640 parameters, accounting for merely $\mathbf{7.84 \times 10^{-5}}$ of the model size.

No additional expressivity in Pre-Norm architectures. In Pre-Norm architectures such as Llama (Touvron et al., 2023) (Fig. 11), each RMSNorm layer is immediately followed by a linear transformation. For example, the FFN block $\boldsymbol{X} + \mathsf{FFN}(\mathsf{RMSNorm}(\boldsymbol{X}; \boldsymbol{\gamma}))$ takes the form

$$\boldsymbol{X} + \boldsymbol{W}_{\text{down}}\big(\sigma(\boldsymbol{W}_{\text{gate}}(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{X}))) \odot \boldsymbol{W}_{\text{up}}(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{X}))\big). \tag{2}$$

The same observation applies to Attn blocks and the final output layer. Since $\boldsymbol{\gamma}$ appears before linear maps, it can be absorbed into them: for any $\boldsymbol{\gamma}$ and linear map $\boldsymbol{W}_2$, choosing $\boldsymbol{W}_1 = \boldsymbol{W}_2 \operatorname{diag}(\boldsymbol{\gamma})$ yields $\boldsymbol{W}_1 \mathsf{Norm}(\boldsymbol{x}) \equiv \boldsymbol{W}_2(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}))$. Thus, from the perspective of expressivity, scale vectors in Pre-Norm architectures are redundant. This raises a natural question:

Although negligible in size and expressively redundant, are scale vectors negligible in training?

Indispensable in practice. To answer this question, we conduct a simple ablation study by removing scale vectors. We train 0.12B Llama models on FineWeb-Edu (Lozhkov et al., 2024); experimental details are provided in Appendix C.2.1. As shown in Figure 1, under the same peak learning rate (lr), the model with scale vectors consistently outperforms the model without them throughout training, reducing the terminal loss by approximately 0.028 and yielding a $1.4\times$ token-efficiency gain. Even after retuning the peak lr for the model without scale vectors, its terminal loss remains approximately 0.015 higher.

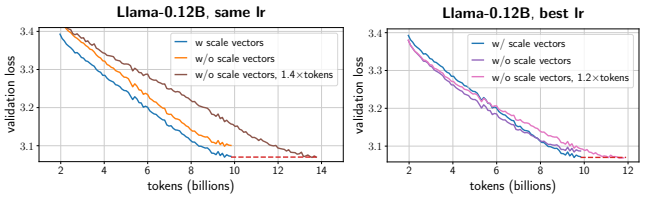


Figure 1: Ablation of scale vectors in 0.12B Llama pre-training. (Left): same peak lr; (Right): separately tuned peak lr.

Theoretical study of training dynamics. These findings suggest that understanding scale vectors requires analyzing their effect on *optimization dynamics*. We therefore study scale vectors in Pre-Norm architectures through the following simplified but representative setting (Saxe et al., 2014).

Setting 2.1. Let $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and $f_\star(\boldsymbol{x}) = \boldsymbol{W}_\star \mathsf{Norm}(\boldsymbol{x})$ with some nonzero $\boldsymbol{W}_\star \in \mathbb{R}^{c \times d}$.

We compare two student models under the squared loss. The model with scale vectors is $f(\boldsymbol{x}; \boldsymbol{W}_f, \boldsymbol{\gamma}) = \boldsymbol{W}_f(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}))$ with $\mathcal{L}_f(\boldsymbol{W}_f, \boldsymbol{\gamma}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}}\|f(\boldsymbol{x}; \boldsymbol{W}_f, \boldsymbol{\gamma}) - f_\star(\boldsymbol{x})\|^2$; the model without scale vectors is $g(\boldsymbol{x}; \boldsymbol{W}_g) = \boldsymbol{W}_g \mathsf{Norm}(\boldsymbol{x})$ with $\mathcal{L}_g(\boldsymbol{W}_g) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}}\|g(\boldsymbol{x}; \boldsymbol{W}_g) - f_\star(\boldsymbol{x})\|^2$. Both objectives are optimized by gradient flow (GF): $(\dot{\boldsymbol{W}}_f(t), \dot{\boldsymbol{\gamma}}(t)) = -\nabla \mathcal{L}_f(\boldsymbol{W}_f(t), \boldsymbol{\gamma}(t))$ and $\dot{\boldsymbol{W}}_g(t) = -\nabla \mathcal{L}_g(\boldsymbol{W}_g(t))$. Following common practice, we initialize the scale vector as $\boldsymbol{\gamma}(0) = \mathbf{1}$, while the remaining parameters have small magnitude. For clarity, we take $\boldsymbol{W}_f(0) = \boldsymbol{W}_g(0) = \mathbf{0}$. Under this initialization, both models have the same initial loss. For brevity, write $\mathcal{L}_f(t) := \mathcal{L}_f(\boldsymbol{W}_f(t), \boldsymbol{\gamma}(t)), \mathcal{L}_g(t) := \mathcal{L}_g(\boldsymbol{W}_g(t))$. We obtain the following result.

Theorem 2.2 (Optimization advantage induced by scale vectors)**.** *Consider Setting 2.1. Then, for every $t > 0$, $\mathcal{L}_f(t) < \mathcal{L}_g(t)$.*

Theoretical insight. The proof of Theorem 2.2 is deferred to Appendix D.1. The key mechanism is that scale vectors transform the original gradient flow into a "*self-amplifying preconditioned*" flow. Specifically, define the effective parameter of f as $\boldsymbol{A}_f(t) := \boldsymbol{W}_f(t) \operatorname{diag}(\boldsymbol{\gamma}(t))$, whose j-th column is $\boldsymbol{a}_{f,j}(t) := \gamma_j(t)\boldsymbol{w}_{f,j}(t) \in \mathbb{R}^c$. Its dynamics satisfy $\dot{\boldsymbol{a}}_{f,j}(t) = (\gamma_j(t)^2 \boldsymbol{I}_c + \boldsymbol{w}_{f,j}(t)\boldsymbol{w}_{f,j}(t)^\top)(\boldsymbol{w}_{\star,j} - \boldsymbol{a}_{f,j}(t))$. By contrast, the model without scale vectors follows $\dot{\boldsymbol{w}}_{g,j}(t) = \boldsymbol{w}_{\star,j} - \boldsymbol{w}_{g,j}(t)$. Thus, the scale vector induces the state-dependent preconditioner $\boldsymbol{P}_{f,j}(t) := \gamma_j(t)^2 \boldsymbol{I}_c + \boldsymbol{w}_{f,j}(t)\boldsymbol{w}_{f,j}(t)^\top$. Under the practical initialization above, gradient flow preserves the invariant $\gamma_j(t)^2 - \|\boldsymbol{w}_{f,j}(t)\|^2 = 1$ and hence $\lambda_{\min}(\boldsymbol{P}_{f,j}(t)) \geqslant \gamma_j(t)^2 \geqslant 1$. Moreover, for every nonzero teacher column $\boldsymbol{w}_{\star,j} \neq \mathbf{0}$, the inequality is strict for all $t > 0$, yielding strictly faster loss descent.

Takeaway. *Although scale vectors are negligible in parameter count and redundant in expressivity within Pre-Norm architectures, they accelerate training through self-amplifying preconditioning.*

2.2 Weight Decay for Scale Vectors

An unresolved weight-decay choice. We next study the training dynamics of scale vectors, with a particular focus on weight decay (wd). WD is a standard component of modern LLM training and is typically applied to matrix parameters in Transformers. For scale vectors, however, the appropriate choice remains unclear. Existing LLM implementations, including OLMo, nanoGPT, and Qwen (OLMo et al., 2024; Karpathy, 2022; Yang et al., 2024; 2025), differ in whether they apply wd to scale vectors. This motivates the question:

Should weight decay be applied to scale vectors?

Distinct types of scale vectors. We primarily focus on scale vectors in Pre-Norm architectures. Since Q/K-Norm (Dehghani et al., 2023) and Output-Norm (Team et al., 2024) are increasingly used in recent LLMs, such as Qwen3 and Gemma3 (Team et al., 2025), we also include them in our study. Specifically, we consider Gemma3, which contains standard Pre-Norm, Q/K-Norm, and Attn/FFN Output-Norm layers. Importantly, RMSNorm layers at different locations play distinct structural roles. We therefore classify the RMSNorm layers in Gemma3 into two types, as illustrated in Fig. 11:

- **Output-Norm layers**: the RMSNorm layer not immediately followed by a linear transformation. This type includes Q/K-Norm and Attn/FFN Output-Norm.
- **Input-Norm layers**: the RMSNorm layer immediately followed by a linear transformation. This type includes standard Pre-norm layers.

Theoretical insight. The scale vectors in these two types of RMSNorm layers play fundamentally different roles in expressivity and optimization, and should therefore be treated differently under wd:

- **Output-Norm scale vectors.** These scale vectors directly parameterize the output of the corresponding submodule and therefore affect its expressivity. Applying wd to them shrinks the output, which can undesirably *restrict expressivity* and weaken the submodule relative to the residual stream. Thus, weight decay should be avoided for Output-Norm scale vectors.
- **Input-Norm scale vectors.** In contrast, these scale vectors add no expressivity (Sec. 2.1); their role is primarily optimization-related. Applying weight decay to them keeps the *parameterization balanced* and *controls Hessian sharpness*, thereby leading to potentially faster and more stable training. Theorem 2.3 provides a detailed analysis.

We now theoretically study whether wd should be applied to Input-Norm scale vectors. Unlike Section 2.1, which analyzes GF, we consider the more realistic setting of stochastic gradient descent (SGD) through its continuous-time SDE approximation. For clarity, we present the scalar-output model $f(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\gamma}) = \langle \boldsymbol{w}, \boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}) \rangle$ with target $f^\star(\boldsymbol{x}) = \langle \boldsymbol{w}^\star, \mathsf{Norm}(\boldsymbol{x}) \rangle$ under Setting 2.1.

Theorem 2.3 (Benefits of wd for Input-Norm scale vectors; informal)**.** *Consider training the above model by continuous-time SGD. Suppose wd $\lambda > 0$ is applied to $\boldsymbol{w}$, and compare two choices for the Input-Norm scale vector $\boldsymbol{\gamma}$: no wd ($\mu = 0$) versus wd ($\mu > 0$).*

- *If $\mu > 0$, then $\mathbb{E}\|\boldsymbol{\gamma}_t\|^2 < \infty$ remains uniformly bounded over time. Consequently, key Hessian sharpness metrics, including $\lambda_{\max}(\nabla^2 \mathcal{L})$, $\mathrm{Tr}(\nabla^2 \mathcal{L})$, and $\|\nabla^2 \mathcal{L}\|_{\mathrm{F}}$, also **remain bounded**.*
- *If $\mu = 0$, then $\mathbb{E}\|\boldsymbol{\gamma}_t\|^2$ becomes unbounded along the trajectory. As a result, the same Hessian sharpness metrics **diverge** along a sequence of times.*

Key mechanism. The formal statement and proof are deferred to Appendix D.2.1. Theorem 2.3 shows that, for Input-Norm scale vectors, wd keeps the parameterization pair $(\boldsymbol{w}, \boldsymbol{\gamma})$ balanced, prevents excessive growth of $\boldsymbol{\gamma}$, and thereby controls Hessian sharpness during training.

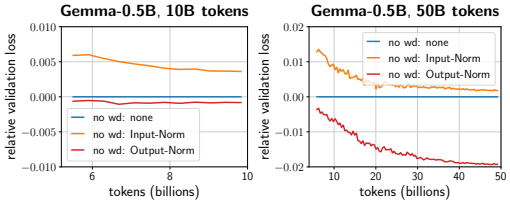


Figure 2: WD is beneficial for Input-Norm scale vectors, but harmful for Output-Norm scale vectors.

Why sharpness matters. In stochastic optimization, the loss descent of SGD *depends directly* on Hessian sharpness metrics, such as $\lambda_{\max}(\nabla^2 \mathcal{L})$, $\mathrm{Tr}(\nabla^2 \mathcal{L})$, and $\|\nabla^2 \mathcal{L}\|_{\mathrm{F}}^2$; see Appendix D.2.2 for details. Therefore, by preventing these quantities from diverging, wd on Input-Norm scale vectors leads to faster training and potentially more stable optimization, including the use of larger lr.

Experimental Justification. To validate our theoretical insights, we independently control wd for Input-Norm and Output-Norm scale vectors. We train 0.5B Gemma models on FineWeb-Edu

for 10B/50B tokens using Muon optimizer (Keller et al., 2024); experimental details are provided in Appendix C.2.2. As shown in Figure 2, applying wd to Input-Norm scale vectors consistently improves performance; whereas removing wd from Output-Norm scale vectors performs better than applying it. These results support the following principle.

Individual weight decay (IWD). *Apply wd to Input-Norm scale vectors, but not to Output-Norm ones.*

3 Improving Scale Vectors

In this section, we propose three methods to further exploit scale vectors. For each method, we first present the motivation and theoretical insight. Then in Section 4, we empirically verify that each design consistently improves LLM pre-training.

3.1 Heterogeneity of Scale Vectors

Shared scale vectors in Attn and FFN. We consider Pre-Norm architecture. In the Attn block, $\boldsymbol{X} + \mathsf{Attn}(\mathsf{RMSNorm}(\boldsymbol{X};\boldsymbol{\gamma}))$, a single Pre-Norm layer $\mathsf{RMSNorm}(\boldsymbol{X};\boldsymbol{\gamma})$ feeds all three projections:

$$\boldsymbol{Q} = \boldsymbol{W}_Q(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{X})), \quad \boldsymbol{K} = \boldsymbol{W}_K(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{X})), \quad \boldsymbol{V} = \boldsymbol{W}_V(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{X})) \tag{3}$$

Thus, the scale vector $\boldsymbol{\gamma}$ is shared across the query, key, and value branches. The FFN block follows the same pattern: a single Pre-Norm output feeds both the gate and up projections.

Heterogeneous scale vectors (HG). Figure 3 shows that the query, key, and value matrices exhibit different growth and decay rates during training. As Section 2.1 shows, in Pre-Norm scale vectors primarily accelerate optimization through self-amplifying preconditioning. From this perspective, different branches should be equipped with separate scale vectors, allowing the induced preconditioners to adapt to branch-specific optimization dynamics. This motivates replacing the shared scale vector in (3) with branch-specific scale vectors $\boldsymbol{\gamma}_Q$, $\boldsymbol{\gamma}_K$, and $\boldsymbol{\gamma}_V$:

$$\begin{aligned} &\boldsymbol{Q} = \boldsymbol{W}_Q(\boldsymbol{\gamma}_Q \odot \mathsf{Norm}(\boldsymbol{X})), \ \boldsymbol{K} = \boldsymbol{W}_K(\boldsymbol{\gamma}_K \odot \mathsf{Norm}(\boldsymbol{X})), \\ &\boldsymbol{V} = \boldsymbol{W}_V(\boldsymbol{\gamma}_V \odot \mathsf{Norm}(\boldsymbol{X})). \end{aligned} \tag{4}$$

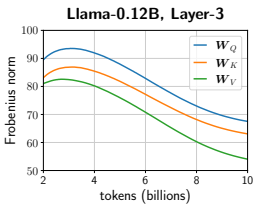


Figure 3: Dynamics of query, key, and value projections.

An analogous modification applies to the FFN block. Specifically, we introduce separate scale vectors $\boldsymbol{\gamma}_{\text{gate}}$ and $\boldsymbol{\gamma}_{\text{up}}$ into the standard FFN in (2), yielding:

$$\boldsymbol{X} + \boldsymbol{W}_{\text{down}}\big(\sigma(\boldsymbol{W}_{\text{gate}}(\boldsymbol{\gamma}_{\text{gate}} \odot \mathsf{Norm}(\boldsymbol{X}))) \odot \boldsymbol{W}_{\text{up}}(\boldsymbol{\gamma}_{\text{up}} \odot \mathsf{Norm}(\boldsymbol{X}))\big).$$

This heterogeneity does not increase expressivity; its role is purely optimization-related. Moreover, it introduces only $\mathcal{O}(d)$ additional parameters, which is negligible relative to the overall model size.

3.2 Placement of Scale Vectors

Standard input-side placement. We consider Pre-Norm architectures. As shown in (2) and (3), scale vectors in Attn or FFN blocks are placed before linear maps. The same holds for the RMSNorm layer before the output projection. Thus, Pre-Norm architectures consistently apply scale vectors on the input side of a linear map. This convention, however, is not obviously optimal for optimization.

Alternative placement for faster optimization. Section 2.1 shows that standard input-side scale vectors already accelerate optimization relative to removing them. A natural question is whether alternative placements can yield an even stronger optimization effects. The key observation is that the placement of a scale vector determines which coordinates are directly modulated. Input-side placement rescales coordinates before they are mixed by $\boldsymbol{W}$, but the resulting output coordinates may still exhibit heterogeneous scales or anisotropic optimization geometry. Hence, the standard placement may not fully exploit the optimization benefits of scale vectors. Motivated by this observation, we consider three increasingly flexible designs.

- AP: **after-placement.** The simplest modification moves the scale vector from the input side to the output side of the linear map:

$$\text{AP:} \qquad \boldsymbol{W}(\boldsymbol{\gamma}^{\text{b}} \odot \mathsf{Norm}(\boldsymbol{x})) \mapsto \boldsymbol{\gamma}^{\text{a}} \odot (\boldsymbol{W}\mathsf{Norm}(\boldsymbol{x})). \tag{5}$$

Compared with the standard design, AP directly modulates output coordinates after linear mixing, and may therefore better match output-side anisotropy.

- DP: **dual-placement.** AP moves the scale vector to the output side but removes input-side modulation. A more flexible alternative is to place scale vectors on both sides of the linear map:

$$\mathsf{DP}: \qquad \boldsymbol{W}(\boldsymbol{\gamma}^{\mathrm{b}} \odot \mathsf{Norm}(\boldsymbol{x})) \mapsto \boldsymbol{\gamma}^{\mathrm{a}} \odot (\boldsymbol{W}(\boldsymbol{\gamma}^{\mathrm{b}} \odot \mathsf{Norm}(\boldsymbol{x}))). \tag{6}$$

  Since scale vectors are negligible in size, this two-sided design incurs only a tiny parameter increase while controlling both input-side and output-side coordinates.

- DNP: **dual normalized placement.** Although DP enables more flexible modulation around the linear map, the extra multiplicative interactions may reduce training stability. We therefore consider a normalized variant that inserts normalization between $\boldsymbol{W}$ and the output-side scale vector:

$$\mathsf{DNP}: \qquad \boldsymbol{W}(\boldsymbol{\gamma}^{\mathrm{b}} \odot \mathsf{Norm}(\boldsymbol{x})) \mapsto \boldsymbol{\gamma}^{\mathrm{a}} \odot \mathsf{Norm}(\boldsymbol{W}(\boldsymbol{\gamma}^{\mathrm{b}} \odot \mathsf{Norm}(\boldsymbol{x}))). \tag{7}$$

  Compared with DP, DNP preserves two-sided modulation while explicitly normalizing the intermediate representation, which may stabilize optimization.

When applied to the query and key projections, DNP is equivalent to adding Q/K-Norm. For other matrices, it introduces additional normalization layers. Moreover, when using heterogeneous input-side scale vectors from Sec. 3.1, it is natural to also use heterogeneous output-side scale vectors.

The following theorem demonstrates that DP can further accelerate the optimization dynamics of scale vectors. Its proof is deferred to Appendix E.1.

Theorem 3.1 (Instantaneous acceleration of DP)**.** *Consider the setting of Theorem 2.2. Let* $\phi(\boldsymbol{x}; \boldsymbol{W}_\phi, \boldsymbol{\gamma}^{\mathrm{a}}, \boldsymbol{\gamma}^{\mathrm{b}}) = \boldsymbol{\gamma}^{\mathrm{a}} \odot (\boldsymbol{W}_\phi(\boldsymbol{\gamma}^{\mathrm{b}} \odot \mathsf{Norm}(\boldsymbol{x})))$ *be the DP student model, and let* $\mathcal{L}_\phi$ *be its squared loss. We compare the standard model* f *and the DP model* ϕ *under gradient flow. Then, at the same effective state, DP is instantaneously* ***at least as fast as*** *the standard model: for any* $(\boldsymbol{W}_f, \boldsymbol{\gamma})$ *and* $(\boldsymbol{W}_\phi, \boldsymbol{\gamma}^{\mathrm{a}}, \boldsymbol{\gamma}^{\mathrm{b}})$ *satisfying* $\boldsymbol{\gamma} = \boldsymbol{\gamma}^{\mathrm{b}}$ *and* $\boldsymbol{W}_f = \mathrm{diag}(\boldsymbol{\gamma}^{\mathrm{a}})\boldsymbol{W}_\phi$*, we have* $\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_\phi \leqslant \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_f \leqslant 0$.

Key mechanism. Theorem 3.1 establishes the instantaneous acceleration of DP at the same effective state as the baseline. The key mechanism is that DP induces *both row-wise and column-wise self-amplifying preconditioning*, leading to faster loss descent than the standard single-sided modulation. A finer trajectory-level analysis, including strict acceleration in both early and global training phases, is deferred to Theorem E.1 in Appendix E.1.

3.3 Reparameterization of Scale Vectors

Why parameterization matters. Another key question is how scale vectors should be parameterized. Even when different parameterizations represent the same functional class, they can induce substantially different optimization dynamics. This issue is relevant for scale vectors, which naturally contain two distinct components: a global magnitude and a directional unit vector. These components play different optimization roles. The magnitude controls the overall strength of modulation, whereas the direction determines how this modulation is distributed across channels. The standard parameterization entangles these two effects in a single vector, potentially leading to suboptimal conditioning. This motivates reparameterizing scale vectors to control magnitude and direction more explicitly.

Magnitude-direction Reparameterization. For nonzero scale vector $\boldsymbol{\gamma} \in \mathbb{R}^d$, we decompose it into its magnitude and direction, and optimize the two components separately. We consider two variants.

- OR: **reparameterization in original space.** A natural approach is to separate magnitude and direction in the standard Euclidean geometry:

$$\mathsf{OR}: \qquad \boldsymbol{\gamma} \mapsto \beta \cdot \mathsf{Norm}(\boldsymbol{\alpha}), \tag{8}$$

  where $\beta \in \mathbb{R}$ controls the global magnitude, and $\mathsf{Norm}(\boldsymbol{\alpha}) \in \mathbb{R}^d$ determines the direction.

- ER: **reparameterization in exponential space.** While OR separates magnitude and direction in Euclidean coordinates, scale vectors act multiplicatively in the network. This suggests that a multiplicative parameterization may be more natural. We therefore consider

$$\mathsf{ER}: \qquad \boldsymbol{\gamma} \mapsto e^{\beta} \cdot e^{\boldsymbol{\alpha} - \mathrm{mean}(\boldsymbol{\alpha})} \tag{9}$$

  where $e^{\beta} \in \mathbb{R}$ controls the overall magnitude, and $e^{\boldsymbol{\alpha} - \mathrm{mean}(\boldsymbol{\alpha})} \in \mathbb{R}^d$ determines the direction.

Although ER produces only positive entries and therefore appears to remove sign information, it does not reduce expressivity in our setting: since scale vectors always appear together with linear maps, the missing signs can be absorbed into the corresponding linear weights.

The following theorem demonstrates that OR can further accelerate optimization dynamics induced by scale vectors. Its proof is deferred to Appendix E.2.

Theorem 3.2 (Optimization advantage of OR)**.** *Consider the setting of Theorem 2.2. Let* $\psi(\boldsymbol{x}; \boldsymbol{W}_\psi, \boldsymbol{\alpha}, \beta) = \boldsymbol{W}_\psi(\beta\mathsf{Norm}(\boldsymbol{\alpha}) \odot \mathsf{Norm}(\boldsymbol{x}))$ *be the reparameterized student model, and let* $\mathcal{L}_\psi$ *be its squared loss. We compare the standard model* f *and the reparameterized model* ψ *under gradient flow, initialized as* $\boldsymbol{\alpha}(0) = \mathbf{1}, \beta(0) = 1, \boldsymbol{W}_\psi(0) = \mathbf{0}$*. Then, at the same effective state, OR is instantaneously* ***at least as fast as*** *the direct parameterization: for any* $(\boldsymbol{W}_f, \boldsymbol{\gamma})$ *and* $(\boldsymbol{W}_\psi, \boldsymbol{\alpha}, \beta)$ *satisfying* $\boldsymbol{W}_f = \boldsymbol{W}_\psi$ *and* $\boldsymbol{\gamma} = \beta\mathsf{Norm}(\boldsymbol{\alpha})$*, we have* $\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_\psi \leqslant \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_f \leqslant 0$.

Key mechanism. Theorem 3.2 establishes the instantaneous acceleration of OR at the same effective state as the baseline. The key mechanism is that OR induces an *anisotropic preconditioner* that treats the magnitude and direction components of the scale vector differently. Under gradient flow, the effective dynamics of $\boldsymbol{\gamma} := \beta\mathsf{Norm}(\boldsymbol{\alpha})$ in ψ satisfy $\dot{\boldsymbol{\gamma}} = -\boldsymbol{P}\nabla_{\boldsymbol{\gamma}}\mathcal{L}_f$, where the implicit preconditioner is $\boldsymbol{P} = d \cdot (\hat{\boldsymbol{\gamma}}\hat{\boldsymbol{\gamma}}^\top) + \rho \cdot (\boldsymbol{I} - \hat{\boldsymbol{\gamma}}\hat{\boldsymbol{\gamma}}^\top)$, where $\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}/\|\boldsymbol{\gamma}\|$ and $\rho = \beta^2 d/\|\boldsymbol{\alpha}\|^2 \approx \mathcal{O}(1)$. Unlike the standard parameterization ($\boldsymbol{P} = \boldsymbol{I}$), this structure amplifies motion along the magnitude direction by a factor of d, aggressively accelerating scale adjustments, while preserving a stable $\mathcal{O}(1)$ descent scale for the orthogonal direction. A finer trajectory-level analysis, including strict acceleration in early and global training phases, is deferred to Theorem E.2 in Appendix E.2 and further supported by numerical simulations.

4 Experiments

We evaluate our methods on LLM pre-training across architectures and model scales. The main configurations are summarized below; additional implementation details are provided in Appendix C.1.

Models and Dataset. We conduct experiments on two widely used LLM architectures: dense **Llama** models (Touvron et al., 2023) and **LlamaMoE** models, with sizes **from 0.12B to 2B** parameters. All models are trained on the high-quality FineWeb-Edu dataset (Lozhkov et al., 2024).

Token Budget. Unless otherwise specified, the training budget is approximately *100* tokens per parameter for dense models and *100* tokens per activated parameter for MoE models. This budget substantially exceeds the Chinchilla-optimal regime (Hoffmann et al., 2022) and is closer to industrial pre-training practice. For small-scale models, we use a total batch size of approximately 0.5M tokens; for large-scale models, we use a large total batch size of approximately 2M tokens.

Optimization. In most experiments, we adopt AdamW as the default optimizer (Loshchilov & Hutter, 2017). For each setting, we tune `lr_max` for the baseline model and use the same value for both the baseline and the corresponding improved model. See more details in Appendix C.1.

To validate our methods beyond this default setting, we additionally report results with the Muon optimizer (Keller et al., 2024) and the warmup-stable-decay (wsd) scheduler (Hu et al., 2024).

4.1 Validation of Each Strategy

We first evaluate the strategies introduced in Sections 2 and 3: HG for scale-vector heterogeneity (Sec. 3.1); AP, DP, and DNP for scale-vector placement (Sec. 3.2); OR and ER for scale-vector reparameterization (Sec. 3.3); and IWD for individual weight decay of scale vectors (Sec. 2.2).

Step-by-step verification. All experiments in this subsection are conducted on the 0.12B Llama model, and the strategies are validated incrementally. The results are shown in Figure 4. (i) The *Heterogeneity* panel shows that heterogeneous scale vectors outperform shared scale vectors. (ii) The *Placement* panel shows that dual placement (DP) consistently improves performance, whereas dual normalized placement (DNP) does not. After placement (AP) improves early training but is later overtaken. (iii) The *Reparameterization* panel shows that applying magnitude-direction reparameterization, either OR or ER, improves performance. Among these variants, DNP+OR achieves the largest gain, outperforming DP+OR and DP+ER, and slightly outperforming DNP+ER. (iv) Although vanilla Llama contains only Pre-Norm layers, which are Input-Norm layers, applying

DNP introduces both Input-Norm and Output-Norm scale vectors. This allows us to re-examine individual weight decay in this setting. As shown in the *Weight Decay* panel, applying weight decay to Input-Norm scale vectors while excluding Output-Norm scale vectors achieves the lowest terminal loss. In addition, we validate the extra Norm and scale vector components of DNP under this optimized weight decay setting, and the results demonstrate that each component still provides improvements (Figure 12). Overall, these results show that all four classes of scale-vector strategies provide gains, further supporting our theoretical understanding.

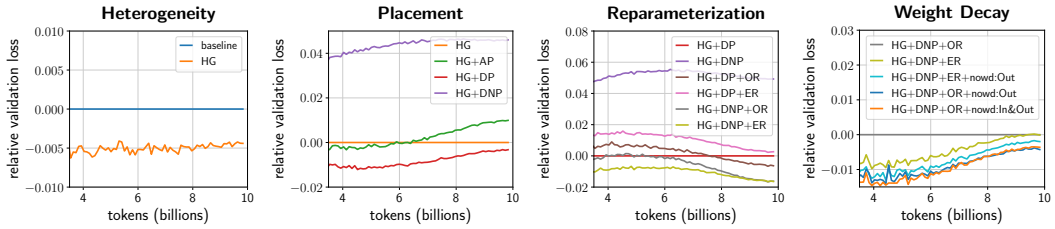


Figure 4: Step-by-step validation of scale-vector strategies on Llama-0.12B. Each strategy class yields gains.

Parameter overhead. Our strategies add only $\mathcal{O}(d)$ scale-vector parameters, whereas Transformer matrix parameters scale as $\mathcal{O}(d^2)$; thus, the overhead is *negligible*. Nevertheless, we conduct a controlled ablation to isolate the effect of parameter count. Specifically, we compare against a widened Llama-0.12B baseline, denoted Llama-large, whose parameter count matches our largest variant, HG+DNP+OR+IWD (0.11985B vs. 0.11974B for the Llama-0.12B baseline). As shown in Figure 5, our method reduces terminal loss by 0.028, whereas LLaMA-large yields only a marginal gain. The parameter overhead further decreases at larger scales; for example, on Llama-1B, our method increases the parameter count by only 7.85×10^{-5}.

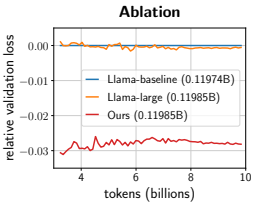


Figure 5: Ablation on parameter count.

The **computational overhead** is also negligible: our strategies introduce only $\mathcal{O}(d)$ elementwise operations, compared with the $\mathcal{O}(d^2)$ cost of Transformer linear projections. Practical measurements on 1B dense models further show minimal wall-clock time and memory overhead (Table 5).

4.2 Main Results of Unified Strategy

Building on the step-by-step validation in Section 4.1, we combine HG (Sec. 3.1), DNP (Sec. 3.2), OR (Sec. 3.3), and IWD (Sec. 2.2) into a *unified scale-vector strategy*. In this subsection, we systematically compare this strategy with tuned baselines across dense and MoE models ranging from 0.12B to 2B parameters. Across all settings, our method consistently improves pre-training loss.

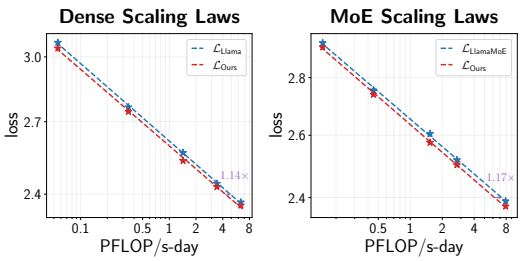


Figure 6: Scaling behavior of our unified scale-vector strategy. (Left): dense models; (right): MoE models. Our strategy scales more favorably than baselines.

Results for Dense Models. Figure 7 compares our method with tuned Llama baselines on FineWeb-Edu for dense Llama models with 0.12B, 0.25B, 0.5B, 0.75B, and 1B parameters; 0.75B results are deferred to Appendix C.1.1. Our method **consistently achieves lower terminal loss** than the tuned Llama baseline and maintains its advantage throughout training, with the gap widening over time. The fitted scaling laws in Figure 6 (left) suggest that these gains may persist at larger scales.

Results for MoE Models. Figure 8 compares our unified scale-vector strategy with tuned Llama-MoE baselines on FineWeb-Edu across MoE models with 0.25B (A0.11B), 0.5B (A0.18B), 0.8B (A0.28B), 1B (A0.38B), and 2B (A0.62B) parameters, where "A" denotes the number of activated parameters; 0.8B results are deferred to Appendix C.1.1. Across all settings, our method **consistently achieves more than 0.015 lower terminal loss** than the well-tuned MoE baseline. Moreover, the advantage is maintained throughout training, with the performance gap widening over time. As shown in Figure 6 (right), our method also exhibits a more favorable **scaling law** than the baseline. In particular, the fitted loss-descent slope of our strategy is **steeper** than that of the baseline.

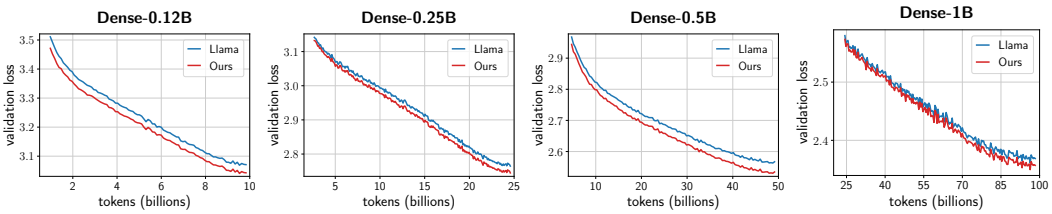


Figure 7: Comparison between our unified scale-vector strategy and the Llama baseline on dense models of different sizes. Our strategy consistently achieves lower terminal loss across model scales.

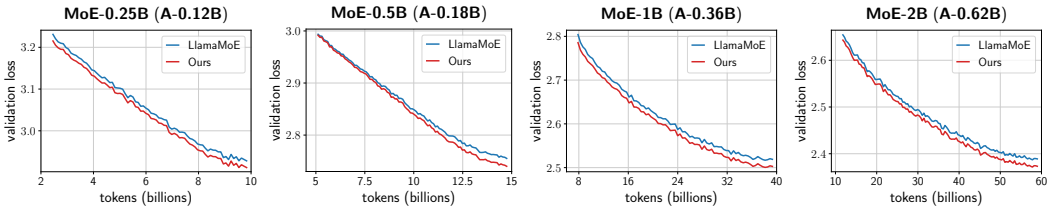


Figure 8: Comparison between our unified scale-vector strategy and the Llama-MoE baseline on MoE models of different sizes. Our strategy consistently achieves lower terminal loss across model scales.

4.3 Compatibility with Muon and wsd

We further examine whether our strategy is compatible with other training configurations, including the Muon optimizer and the wsd lr schedule. Experimental details are provided in Appendix C.1.3.

Muon (Keller et al., 2024; Liu et al., 2025) has recently demonstrated strong efficiency and scalability in LLM pre-training. For the Muon baseline, we adopt the implementation of Liu et al. (2025). We conduct experiments on Dense-0.12B and MoE-0.5B models. The results are shown in the two left panels of Figure 9. Similar to the AdamW setting, our strategy reduces the terminal loss by more than 0.01 compared with the strong Muon baseline and maintains its advantage throughout training. We conjecture that this behavior is consistent with the AdamW results because Muon still optimizes scale vectors using AdamW, while our strategy primarily modifies scale vectors.

The **wsd** schedule has recently become a popular choice of LLM pre-training. On Dense-0.12B and MoE-0.5B models, the advantage of our strategy gradually increases throughout the LR stable phase and does not diminish in the decay phase (the two right panels of Figure 9). This suggests that our strategy may be particularly suitable for modern over-training pipelines that adopt wsd schedules.

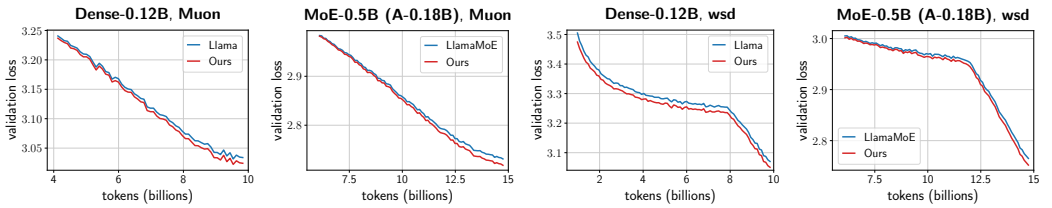


Figure 9: (Left two panels) results with the Muon optimizer; (Right two panels) results with the wsd lr schedule. Our unified scale-vector strategy consistently improves over the corresponding baselines.

5 Conclusion

We present a systematic study of scale vectors in LLMs. Although scale vectors account for only a negligible fraction of model parameters, we show that they have a substantial impact on LLM pre-training. Our theory attributes this effect to optimization rather than expressivity, clarifies when weight decay should be applied, and motivates improved designs based on heterogeneity, placement, and reparameterization. Extensive experiments confirm that these designs consistently improve pre-training performance. Future work includes extending our analysis to broader architectures and studying how scale-vector design should evolve with model scale.

AI USE STATEMENT

In this work, we used generative AI tools to polish parts of the writing and to assist with parts of the code. Generative AI tools were not used to develop or propose the research methodology or to evaluate or interpret the experimental results. All AI-assisted text and code were manually reviewed and revised by the authors. We take full responsibility for the final content of this work, including any content generated with the aid of generative AI.

ETHICS STATEMENT

The authors do not foresee any ethical concerns raised by this work.

REPRODUCIBILITY STATEMENT

All experiments are conducted on publicly available datasets. We provide detailed descriptions of the proposed method, including the full procedure, hyperparameter settings, and implementation details.

REFERENCES

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International conference on machine learning*, pp. 244–253. PMLR, 2018. 22

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1, 14

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018. 26

Wenrui Cai, Defa Zhu, Qingjie Liu, and Qiyang Min. Seednorm: Self-rescaled dynamic normalization. *arXiv preprint arXiv:2510.22777*, 2025. 15

Chen Chen and Lai Wei. Post-layernorm is back: Stable, expressive, and deep. *arXiv preprint arXiv:2601.19895*, 2026. 14

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise SGD provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021. 26

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pp. 7480–7512. PMLR, 2023. 4

Yu Feng and Yuhai Tu. The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9):e2015617118, 2021. 26

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850. PMLR, 10–15 Jul 2018. 16

Jeff Z HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In *Conference on Learning Theory*, pp. 2315–2357. PMLR, 2021. 26

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016. 26

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 35:30016–30030, 2022. 7, 17

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024. 7, 18

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1, 14

Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022. 4

Jordan Keller et al. Muon optimizer. *URL https://github.com/KellerJordan/Muon?tab=readme-ov-file*, 2024. 5, 7, 9, 18, 19

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 16, 17

Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110. PMLR, 2017. 23

Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after SGD reaches zero loss?–a mathematical framework. *International Conference on Learning Representations*, 2022. 26

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*, 2025. 9, 18

Kangqiao Liu, Liu Ziyin, and Masahito Ueda. Noise and fluctuation of finite learning rate stochastic gradient descent. In *International Conference on Machine Learning*, pp. 7045–7056. PMLR, 2021. 26

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7, 17

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content. *URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu*, 2024. 3, 7

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 07–09 Jul 2015. 16

Takashi Mori, Liu Ziyin, Kangqiao Liu, and Masahito Ueda. Logarithmic landscape and power-law escape rate of SGD. *arXiv preprint arXiv:2105.09557*, pp. 15959–15975, 2021. 26

Takashi Mori, Liu Ziyin, Kangqiao Liu, and Masahito Ueda. Power-law escape rate of sgd. In *International Conference on Machine Learning*, pp. 15959–15975. PMLR, 2022. 26

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th international conference on spoken language translation*, 2019. 14

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024. 4

Zihan Qiu, Zeyu Huang, Kaiyue Wen, Peng Jin, Bo Zheng, Yuxin Zhou, Haofeng Huang, Zekun Wang, Xiao Li, Huaqing Zhang, et al. A unified view of attention and residual sinks: Outlier-driven rescaling is essential for transformer training. *arXiv preprint arXiv:2601.22966*, 2026. 15

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pp. 1674–1703. PMLR, 2017. 26

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*, 2014. 3

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 16

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. 4

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, et al. Gemma 3 technical report, 2025. 4, 19

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 16

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3, 7, 16, 17

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 93423–93444, 2025. 16

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024a. 14

Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, Weinan E, and Lei Wu. The sharpness disparity principle in transformers for accelerating language model pre-training. *International Conference on Machine Learning*, pp. 64859–64879, 2025. 16

Jinbo Wang, Mingze Wang, Jiaqi Zhang, Wei Wang, Peng Pei, Xunliang Cai, Weinan E, and Lei Wu. Gradpower: Powering gradients for faster language model pre-training. *International Conference on Machine Learning*, 2026. 16

Mingze Wang and Lei Wu. The noise geometry of stochastic gradient descent: A quantitative and analytical characterization. *arXiv preprint arXiv:2310.00692*, 2023. 26

Mingze Wang, Haotian He, Jinbo Wang, Zilin Wang, Guanhua Huang, Feiyu Xiong, Zhiyu Li, Weinan E, and Lei Wu. Improving generalization and convergence by enhancing implicit regularization. *Advances in Neural Information Processing Systems*, 2024b. 16

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011. 26

Stephan Wojtowytsch. Stochastic gradient descent with noise of machine learning type. part II: Continuous time analysis. *arXiv preprint arXiv:2106.02588*, 2021. 26

Lei Wu, Mingze Wang, and Weijie Su. The alignment property of sgd noise and how it helps select flat minima: A stability analysis. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4680–4693, 2022. 26

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020. 14

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. 4, 17

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 4

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022. 18

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 14

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025a. 19

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. In *Proceedings of the computer vision and pattern recognition conference*, pp. 14901–14911, 2025b. 14

Shuchen Zhu, Rizhen Hu, Mingze Wang, Mou Sun, Xue Wang, Kun Yuan, and Zaiwen Wen. Accelerating llm pre-training through flat-direction dynamics enhancement. *arXiv preprint arXiv:2602.22681*, 2026. 16

Zhijian Zhuo, Yutao Zeng, Ya Wang, Sijun Zhang, Jian Yang, Xiaoqing Li, Xun Zhou, and Jinwen Ma. Hybridnorm: Towards stable and efficient transformer training via hybrid normalization. *arXiv preprint arXiv:2503.04598*, 2025. 14

Liu Ziyin, Mingze Wang, Hongchao Li, and Lei Wu. Parameter symmetry and noise equilibrium of stochastic gradient descent. *Advances in Neural Information Processing Systems*, 37:93874–93906, 2024. 25

Appendix

A Related Works **14**

B A Unified Preconditioning View of Scale-Vector Designs **15**

C Experimental Details **16**

C.1 Experimental details for Section 4 16

C.2 Experimental details for Section 2 and 3 18

C.3 Additional Experiments 19

D Proofs in Section 2 **20**

D.1 Proofs in Section 2.1 20

D.2 Proofs in Section 2.2 23

E Proofs in Section 3 **27**

E.1 Proofs in Section 3.2 27

E.2 Proofs in Section 3.3 31

A Related Works

Normalization operations. The Normalization layer has long been recognized as a key ingredient for stable and efficient deep network training. Early work introduced Batch Normalization (BN) (Ioffe & Szegedy, 2015), which uses batch statistics and achieved great success in computer vision. Layer Normalization (LN) (Ba et al., 2016) later removed the dependence on batch statistics and became the standard choice for sequence modeling and Transformers. More recently, Root Mean Square Layer Normalization (RMSNorm) (Zhang & Sennrich, 2019) further simplified LN by discarding mean centering, and has become the dominant normalization operation in modern LLMs due to its simplicity and efficiency. Beyond these standard designs, recent work has also explored alternatives to explicit normalization, such as Dynamic Tanh (DyT) (Zhu et al., 2025b), which replaces normalization layers with element-wise nonlinear transformations. Overall, prior work has focused primarily on the normalization operation itself, seeking better statistics, lower overhead, or normalization-free replacements. *In contrast*, our work focuses on the scale vector, a component retained across these designs but receiving much less theoretical and empirical attention.

Location of normalization layers. Another line of work studies *where* normalization should be placed within Transformer blocks. The original Transformer adopted Post-Norm, applying normalization after each residual addition. Subsequent work showed Pre-Norm improves training stability, which led it to become the default design in many modern LLMs (Xiong et al., 2020). However, Pre-Norm also introduces structural limitations, including weaker gradient propagation and reduced contribution from deeper layers (Nguyen & Salazar, 2019). Later studies explored combining the advantages of Pre-Norm and Post-Norm, aiming to stable optimization with loss gradient propagation. These works include (Wang et al., 2024a; Zhuo et al., 2025; Chen & Wei, 2026) *In contrast* to this line, which focuses on the location of normalization layers, our work studies the design and mechanism of the scale vectors within these layers.

Normalization layers with enhanced nonlinearity and learnable scaling. More recent work has enriched normalization layers by introducing stronger nonlinear or adaptive modulation mechanisms, for example through gating modules or shallow feedforward networks (Cai et al., 2025; Qiu et al., 2026). These methods increase the flexibility of normalization layers by making them more adaptive to different inputs.

In contrast, rather than modifying normalization layers through additional nonlinear modules, we study the default scale vector itself, provide a theoretical understanding of its necessity and weight decay, and propose several simple but effective improvements. Moreover, several of our designs, including type-dependent weight decay, heterogeneous scale vectors, and dual-sided placements, are largely orthogonal to these methods and can potentially be combined with them.

B A Unified Preconditioning View of Scale-Vector Designs

The preceding subsections introduce three different scale-vector designs: heterogeneity, placement, and reparameterization. We now show that they can be understood under a unified principle. For Pre-Norm scale vectors $\boldsymbol{\gamma}$, the scale vector is immediately followed by a matrix $\boldsymbol{W}$ and therefore does not enlarge the expressivity of the corresponding linear transformation. Instead, it changes the *parameterization* of the effective matrix optimized during training. Consequently, these designs act as *lightweight preconditioners*.

Preconditioning via reparameterization. We first illustrate the general connection between reparameterization and preconditioning. Let $\boldsymbol{p} \in \mathbb{R}^m$ denote an effective parameter that enters the loss $\mathcal{L}(\boldsymbol{p})$. If $\boldsymbol{p}$ is optimized directly, its gradient flow is:

$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p}). \tag{10}$$

Now consider that $\boldsymbol{p}$ is parameterized by another variable $\boldsymbol{q} \in \mathbb{R}^n$ with $n \geq m$ through a differentiable map

$$\boldsymbol{p} = \Phi(\boldsymbol{q}), \qquad \Phi : \mathbb{R}^n \to \mathbb{R}^m.$$

Under this reparameterization, by the chain rule, $\nabla_{\boldsymbol{q}}\mathcal{L}(\Phi(\boldsymbol{q})) = \frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}^\top \nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p})$. Therefore, gradient flow on the parameter $\boldsymbol{q}$, $\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = -\nabla_{\boldsymbol{q}}\mathcal{L}(\Phi(\boldsymbol{q}))$, induces the following dynamics on the effective parameter $\boldsymbol{p} = \Phi(\boldsymbol{q})$:

$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = \frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = -\frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}\frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}^\top \nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p}). \tag{11}$$

Comparing (11) with (10), we see that reparameterization induces a state-dependent preconditioned gradient flow in the effective parameter space, with preconditioner

$$\boldsymbol{P}_\Phi(\boldsymbol{q}) := \frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}\frac{\partial\Phi(\boldsymbol{q})}{\partial\boldsymbol{q}}^\top. \tag{12}$$

This preconditioner $\boldsymbol{P}_\Phi(\boldsymbol{q})$ is positive semi-definite and state-dependent.

Reparameterization forms of scale-vector designs. We now illustrate the above preconditioning view for our scale-vector designs. Consider a group of linear branches $\{\boldsymbol{W}_c\}_c$ following a Pre-Norm layer, such as $c \in \{Q, K, V\}$ in an Attn block, $c \in \{\mathrm{gate}, \mathrm{up}\}$ in a FFN block, or a single linear branch such as the final output projection. For a branch matrix $\boldsymbol{W}_c \in \mathbb{R}^{d_2 \times d_1}$, a general class of lightweight reparameterizations of the effective matrix can be written as

$$\boldsymbol{W}_c \mapsto \mathrm{diag}(\boldsymbol{u}_c)\boldsymbol{W}_c\,\mathrm{diag}(\boldsymbol{v}_c) = \boldsymbol{W}_c \odot (\boldsymbol{u}_c\boldsymbol{v}_c^\top), \tag{13}$$

where $\boldsymbol{u}_c \in \mathbb{R}^{d_2}, \boldsymbol{v}_c \in \mathbb{R}^{d_1}$. This form introduces a structured multiplicative scale field over the entries of $\boldsymbol{W}_c \in \mathbb{R}^{d_2 \times d_1}$. Notably, $\boldsymbol{u}_c\boldsymbol{v}_c^\top$ at most rank one, so the design uses only $\mathcal{O}(d_1 + d_2)$ additional parameters while modulating an $\mathcal{O}(d_1 d_2)$-dimensional matrix.

Table 1 summarizes how our scale-vector strategies fit into this unified form, including the standard scale vector, **HG**, **AP**, **DP**, **OR**, and **ER**. The only exception is **DNP**

$(\mathrm{diag}(\boldsymbol{u}_c)\,\mathrm{Norm}(\boldsymbol{W}_c\,\mathrm{diag}(\boldsymbol{v}_c)\,\cdot))$, which is not a purely reparameterization because it inserts an intermediate normalization to further stabilize the hidden representation. Although each row in Table 1 presents a single strategy, many strategies are complementary and can be combined to obtain richer scale-vector designs.

Table 1: Scale-vector designs as special cases of the unified reparameterization $\boldsymbol{W}_c \mapsto \boldsymbol{W}_c \odot (\boldsymbol{u}_c \boldsymbol{v}_c^\top)$.

Design	Choice in the unified form (13)	Preconditioning interpretation
Standard	$\boldsymbol{u}_c = \mathbf{1},\ \boldsymbol{v}_c = \boldsymbol{\gamma}$	Input-side channel preconditioning
HG	$\boldsymbol{u}_c = \mathbf{1},\ \boldsymbol{v}_c = \boldsymbol{\gamma}_c$	Branch-specific preconditioning
AP	$\boldsymbol{u}_c = \boldsymbol{\gamma}^{\mathrm{a}},\ \boldsymbol{v}_c = \mathbf{1}$	Output-side channel preconditioning
DP	$\boldsymbol{u}_c = \boldsymbol{\gamma}^{\mathrm{a}},\ \boldsymbol{v}_c = \boldsymbol{\gamma}^{\mathrm{b}}$	Two-sided input/output preconditioning
OR	$\boldsymbol{u}_c = \mathbf{1},\ \boldsymbol{v}_c = \beta \cdot \mathrm{Norm}(\boldsymbol{\alpha})$	Magnitude-direction anisotropic metric
ER	$\boldsymbol{u}_c = \mathbf{1},\ \boldsymbol{v}_c = e^{\beta} \cdot e^{\boldsymbol{\alpha} - \mathrm{mean}(\boldsymbol{\alpha})}$	Multiplicative/log-space scale metric

Comparison with adaptive optimizer preconditioning. Modern adaptive optimizers, such as RMSProp (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2014), KFAC (Martens & Grosse, 2015), Shampoo (Gupta et al., 2018), and Soap (Vyas et al., 2025), introduce an *optimizer-induced preconditioner* $\boldsymbol{F}_t^{-1}$ in the raw parameter space. Recent variants further improve optimization by enhancing dynamics along flat directions of the preconditioned geometry (Wang et al., 2024b; 2025; 2026; Zhu et al., 2026). Under adaptive optimization, the gradient flow $\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p})$ is modified to $\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\boldsymbol{F}_t^{-1}(\boldsymbol{p})\nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p})$, where $\boldsymbol{F}_t$ is estimated from gradient statistics accumulated during training. For example, RMSProp approximately uses $\boldsymbol{F}_t = \mathrm{diag}(\sqrt{\mathrm{EMA}_t[(\nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p}_t))^{\odot 2}]})$.

In contrast, a reparameterization Φ induces a *reparameterization-induced preconditioner* $\boldsymbol{P}_\Phi$, as in (12). This preconditioner depends on the current model state, but not explicitly on historical gradients. Table 2 summarizes the differences. Thus, the two forms of preconditioning arise from *fundamentally different mechanisms*: reparameterization reshapes the local optimization geometry through the current model state, whereas adaptive optimizers adjust updates according to accumulated gradient statistics.

Table 2: Comparison between optimizer-induced and reparameterization-induced preconditioning.

	Adaptive optimizer	Reparameterization Φ
Preconditioner	$\boldsymbol{F}_t^{-1}(\boldsymbol{p}_t)$	$\boldsymbol{P}_\Phi(\boldsymbol{q}_t) = \frac{\partial \Phi(\boldsymbol{q}_t)}{\partial \boldsymbol{q}} \frac{\partial \Phi(\boldsymbol{q}_t)}{\partial \boldsymbol{q}}^\top$
Determined by	gradient history $\{\nabla_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p}_s)\}_{s \le t}$	current model state $\boldsymbol{q}_t$

Combining these two mechanisms further alters the effective preconditioning geometry and may yield additional optimization gains. Intuitively, reparameterization provides a more suitable coordinate system, while adaptive optimizers exploit gradient statistics within that coordinate system. The experiments in Section 4 support this complementary effect.

C Experimental Details

All experiments are conducted on H100 80G GPUs.

C.1 Experimental details for Section 4

C.1.1 Section 4.2

Models. We utilize two popular classes of LLM models for our pre-training experiments:

- **Llama.** Llama (Touvron et al., 2023) is a dense decoder-only Transformer architecture that uses Rotary Positional Encoding (RoPE) (Su et al., 2024), SwiGLU, RMSNorm, and a Pre-Norm design. We set $d_{\mathrm{FFN}} = \mathrm{int}(8d_{\mathrm{model}}/3)$. We pre-train Llama models ranging from 0.12B to 1B parameters. Detailed configurations are provided in Table 3.

- **LlamaMoE.** LlamaMoE is a decoder-only mixture-of-experts architecture based on Llama. Each model uses 32 sparse experts, activates 4 sparse experts per token, and includes one shared expert. Following QwenMoE (Yang et al., 2024), the hidden dimension of the shared expert is $\mathrm{int}(8d_{\mathrm{model}}/3)$, whereas that of each sparse expert is $\mathrm{int}(2d_{\mathrm{model}}/3)$. We pre-train LlamaMoE models ranging from 0.25B to 2B parameters. Detailed configurations are provided in Table 4.

Token Budget. In this section, the total number of training tokens is set to approximately 100 times the number of model parameters for dense models, or 100 times the number of activated parameters for MoE models. This budget substantially exceeds the Chinchilla-optimal regime (Hoffmann et al., 2022) and is closer to industrial pre-training practice. As shown in Tables 3 and 4, small-scale models use a sequence length of 1,024 and a batch size of 512, corresponding to a total batch size of approximately 0.5M tokens; large-scale models use a sequence length of 4,096 with a batch size of 512, corresponding to approximately 2M tokens per batch.

Optimization. In this section, we use AdamW as the default baseline optimizer (Kingma & Ba, 2014; Loshchilov & Hutter, 2017). Following standard Llama pre-training practice (Touvron et al., 2023), we set $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $\lambda = 0.1$, and the gradient clipping threshold to 1.0. We use the `cos` schedule: linear warm-up to a peak learning rate `lr_max`, followed by cosine decay to `lr_min`=`lr_max`/20. For each setting, we first tune `lr_max` for the baseline model and then use the tuned value for both the baseline and the corresponding improved model. The grid search for `lr_max` is performed over {`6e-4`, `1e-3`, `2e-3`, `3e-3`, `4e-3`, `5e-3`, `6e-3`}. The selected peak lr's are reported in Tables 3 and 4.

Table 3: Model configurations and tuned peak lr's for dense models

Acronym	Size	d_{model}	n_{head}	n_{layer}	total batch size	`lr_max`
Llama-0.12B	0.12B	768	12	6	1024×512	4e-3
Llama-0.25B	0.25B	1024	16	12	1024×512	2e-3
Llama-0.5B	0.48B	1280	20	18	4096×512	2e-3
Llama-0.75B	0.75B	1536	24	21	4096×512	2e-3
Llama-1B	1.03B	1792	28	22	4096×512	1e-3

Table 4: Model configurations and tuned peak lr's for MoE models.

Acronym	Size	Activated Size	d_{model}	n_{head}	n_{layer}	total batch size	`lr_max`
MoE-0.25B	0.25B	0.11B	640	10	6	1024×512	3e-3
MoE-0.5B	0.48B	0.18B	748	12	9	1024×512	2e-3
MoE-0.8B	0.80B	0.28B	960	15	10	4096×512	2e-3
MoE-1B	1.06B	0.38B	1024	16	12	4096×512	1e-3
MoE-2B	2.00B	0.62B	1280	20	15	4096×512	6e-4

Due to space limitations in the main text, the results for Dense-0.75B and MoE-0.8B models are reported in Figure 10.

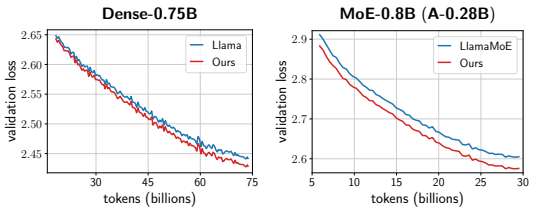


Figure 10: Results for Dense-0.75B; MoE-0.8B, complementary to Figures 7 and 8.

C.1.2 SECTION 4.1

The baseline model in Figure 4 uses the same training configuration as Llama-0.12B in Appendix C.1.1.

For the model-size ablation (Figure 5), we construct a larger Llama variant based on Llama-0.12B. Directly increasing the hidden dimension would require the dimension to remain compatible with both RoPE and the attention heads, leading to a parameter count much larger than the target. Therefore, we instead increase the FFN hidden dimension from d_{FFN} to $d_{\text{FFN}} + 8$, which closely matches the target parameter count. We retune the learning rate for this larger model, and the selected value remains `4e-3`.

To empirically measure computational overhead, we train a Llama-1B model with standard `torch.compile`, under the settings in Appendix C.1.1. As shown in Table 5, our method incurs only minor increases in wall-clock time and memory usage. Results are averaged over 10 training steps after warmup.

Table 5: Runtime and memory overhead on Dense-1B model. Ratios are relative to the corresponding baseline.

model	wall-clock time (ms)	memory usage (MiB)
baseline	151	34263
our strategy	157 (1.04×)	34729 (1.01×)

C.1.3 SECTION 4.3

Muon optimizer. We follow the Muon setup of Keller et al. (2024): Muon is applied only to 2D matrix blocks in Transformer layers, while AdamW is used for all other parameters, including scale vectors, embedding layer, and the output layer. Additionally, following Liu et al. (2025), we further use: (i) per-parameter update scaling, with learning-rate multiplier $c = 0.2\sqrt{\max\{m, n\}}$ for Muon blocks of shape $\mathbb{R}^{m\times n}$, so that the update RMS norm matches that of AdamW; (ii) Nesterov momentum with coefficient $\theta_{\text{muon}} = 0.95$; (iii) weight decay $\lambda = 0.1$. For Dense-0.12B and MoE-0.5B, we use the batch sizes reported in Tables 3 and 4. To ensure strong baselines, we tune the peak lr for Muon. The selected `lr_max` is `5e-3` for Dense-0.12B and `4e-3` for MoE-0.5B, both of which are larger than the corresponding tuned values for AdamW.

Wsd schedule. For AdamW with the `wsd` schedule (Zhai et al., 2022; Hu et al., 2024), we use a linear warmup to the peak learning rate `lr_max`, followed by a stable phase in which the learning rate remains at `lr_max` until 80% of the total training steps, and finally a linear decay to zero. For fairness, we tune the peak learning rate under the `wsd` schedule. The selected `lr_max` is `3e-3` for Dense-0.12B and `2e-3` for MoE-0.5B.

C.2 EXPERIMENTAL DETAILS FOR SECTION 2 AND 3

C.2.1 SECTION 2.1

The baseline model (w/ scale vectors) in Figure 1 uses the same training configuration as Llama-0.12B in Appendix C.1.1.

The comparison model (w/o scale vectors) in Figure 1 is obtained by removing all scale vectors from Llama-0.12B. For the peak lr `max_lr`, the left subfigure uses the same value as the baseline, `4e-3`, whereas the right subfigure retunes this value and selects `2e-3`. All other settings follow Appendix C.1.1.

The longer-training experiments only change the number of training steps. Specifically, the model w/o scale vectors is trained for 1.4× more steps in the left subfigure and 1.2× more steps in the right subfigure.

C.2.2 SECTION 2.2

For Figure 2, we train Gemma-0.5B models (Team et al., 2025) on FineWeb-Edu for approximately 10B and 50B tokens using the Muon optimizer (Keller et al., 2024).

- **Model.** Gemma-0.5B uses the same d_{model}, d_{FFN}, n_{head}, and n_{layer} as Llama-0.5B in Appendix C.1.1. However, its normalization layers differ from those of Llama. We compare the two architectures in Figure 11.
- **Training.** The Muon baseline follows the setup in Appendix C.1.3. We use the same batch size as Llama-0.5B in Table 3. We train for 5,000 steps, corresponding to approximately 10B tokens, denoted by 10BT, and for 25,000 steps, corresponding to approximately 50B tokens, denoted by 50BT. We first tune the peak lr for the baseline setting, which applies weight decay to all scale vectors, under the 50BT training budget. The selected `max_lr` is `4e-3`, which is then used for all other experiments.

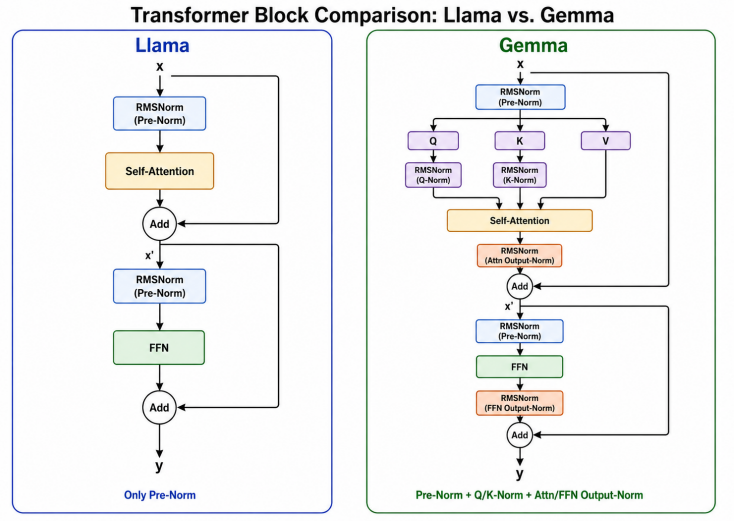


Figure 11: Comparison between Llama and Gemma architectures. Llama contains only standard Pre-Norm layers, whereas Gemma additionally includes Q/K-norm and Attn/FFN Output-Norm layers. **Input-Norm v.s. Output-Norm.** According to our classification in Section 2.2, Pre-Norm layers are Input-Norm layers, whereas Q/K-Norm and Attn/FFN Output-Norm layers are Output-Norm layers.

C.2.3 EXPERIMENTAL DETAILS FOR SECTION 3.1

The model in Figure 3 uses the same training configuration as Llama-0.12B in Appendix C.1.1.

C.3 ADDITIONAL EXPERIMENTS

Results under different random seeds The seed-induced variation in final validation loss is negligible compared to the loss reduction in our experiments. Across five different random seeds in the Dense-0.12B experiment, the baseline achieves final validation losses of 3.07533, 3.07561, 3.07534, 3.07582, 3.07501 yielding a mean of 3.07542 with a standard deviation of 3.08×10^{-4}. The variation among them is consistently below 1e-3, which is much less than the 1e-2 level loss reduction of our strategies.

Downstream tasks Results of downstream evaluation on the MoE-2B model are in Table 6. The results show that our strategy achieves non-trivial improvements on these metrics.

Complementarity of Our Strategy and Hyper-Connections (HC) Our scale-vector strategy is complementary to HC (Zhu et al., 2025a). Our strategy operates on the representation dimension (e.g., d_{model}) , whereas HC operates on the residual-stream dimension (e.g., 4). Applying our HG

Table 6: Zero-shot downstream evaluation of the MoE-2B model.

Model	HellaSwag	PIQA	WinoGrande	ARC-Challenge	MMLU	Avg.
Baseline (AdamW)	53.55	71.87	55.96	35.84	24.75	48.39
ours	**54.42**	**72.69**	**56.99**	**36.26**	**25.49**	**49.17**

scale-vector strategy yields a 0.003 loss reduction on the 0.12B dense model with HC (3.0336 vs. 3.0365). The gain is similar to the improvement observed on the corresponding dense model without HC.

Ablation on DNP Figure 12 illustrates the role of extra Norm and scale-vectors in DNP, demonstrating that both components yield gains.

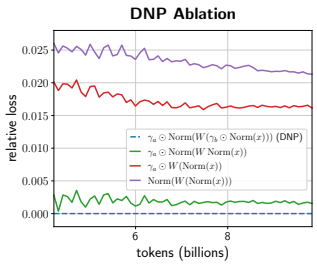


Figure 12: Validation of Norm and scale-vectors in DNP on Llama-0.12B.

D Proofs in Section 2

D.1 Proofs in Section 2.1

Proof of Theorem 2.2. For clarity, we denote

$$\boldsymbol{W} := \boldsymbol{W}_g, \quad \boldsymbol{U} := \boldsymbol{W}_f.$$

Let $\boldsymbol{z} = \mathsf{Norm}(\boldsymbol{x}) = \sqrt{d}\,\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$. Since $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ is isotropic, $\boldsymbol{z}$ is uniformly distributed on the sphere of radius $\sqrt{d}$. Hence $\mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top] = \boldsymbol{I}_d$. Therefore,

$$\mathcal{L}_g(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}^\star\|_F^2.$$

Similarly, writing

$$f(\boldsymbol{x}; \boldsymbol{U}, \boldsymbol{\gamma}) = \boldsymbol{U}\boldsymbol{D}_{\boldsymbol{\gamma}}\boldsymbol{z}, \qquad \boldsymbol{D}_{\boldsymbol{\gamma}} = \mathrm{diag}(\boldsymbol{\gamma}) \in \mathbb{R}^{d\times d},$$

we have

$$\mathcal{L}_f(\boldsymbol{U}, \boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{U}\boldsymbol{D}_{\boldsymbol{\gamma}} - \boldsymbol{W}^\star\|_F^2.$$

Step I: dynamics of the model without scale vectors. Since

$$\mathcal{L}_g(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}^\star\|_F^2,$$

its gradient flow is

$$\frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}t} = \boldsymbol{W}^\star - \boldsymbol{W}.$$

With $\boldsymbol{W}(0) = \boldsymbol{0}$, this gives

$$\boldsymbol{W}(t) = (1 - e^{-t})\boldsymbol{W}^\star.$$

Thus

$$\mathcal{L}_g(\boldsymbol{W}(t)) = \frac{1}{2}e^{-2t}\|\boldsymbol{W}^\star\|_F^2.$$

Step II: dynamics of the model with scale vectors. Write the columns of $\boldsymbol{U}$ and $\boldsymbol{W}^\star$ as

$$\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_d], \qquad \boldsymbol{W}^\star = [\boldsymbol{w}_1^\star, \dots, \boldsymbol{w}_d^\star],$$

where $\boldsymbol{u}_j, \boldsymbol{w}_j^\star \in \mathbb{R}^c$. Then

$$\boldsymbol{U}\boldsymbol{D}_{\boldsymbol{\gamma}} = [\gamma_1\boldsymbol{u}_1, \dots, \gamma_d\boldsymbol{u}_d],$$

and the loss decomposes as

$$\mathcal{L}_f(\boldsymbol{U}, \boldsymbol{\gamma}) = \frac{1}{2}\sum_{j=1}^{d}\|\gamma_j\boldsymbol{u}_j - \boldsymbol{w}_j^\star\|_2^2.$$

For each coordinate j, define

$$\boldsymbol{r}_j = \boldsymbol{w}_j^\star - \gamma_j\boldsymbol{u}_j, \qquad \ell_j = \frac{1}{2}\|\boldsymbol{r}_j\|_2^2.$$

Then

$$\mathcal{L}_f(\boldsymbol{U}, \boldsymbol{\gamma}) = \sum_{j=1}^{d}\ell_j.$$

The gradient-flow equations for the j-th column are

$$\frac{\mathrm{d}\boldsymbol{u}_j}{\mathrm{d}t} = \gamma_j\boldsymbol{r}_j, \qquad \frac{\mathrm{d}\gamma_j}{\mathrm{d}t} = \boldsymbol{u}_j^\top\boldsymbol{r}_j.$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}t}(\gamma_j\boldsymbol{u}_j) = \frac{\mathrm{d}\gamma_j}{\mathrm{d}t}\boldsymbol{u}_j + \gamma_j\frac{\mathrm{d}\boldsymbol{u}_j}{\mathrm{d}t} = \boldsymbol{u}_j\boldsymbol{u}_j^\top\boldsymbol{r}_j + \gamma_j^2\boldsymbol{r}_j.$$

Equivalently,

$$\frac{\mathrm{d}\boldsymbol{r}_j}{\mathrm{d}t} = -(\gamma_j^2\boldsymbol{I}_c + \boldsymbol{u}_j\boldsymbol{u}_j^\top)\boldsymbol{r}_j.$$

Hence

$$\frac{\mathrm{d}\ell_j}{\mathrm{d}t} = \boldsymbol{r}_j^\top\frac{\mathrm{d}\boldsymbol{r}_j}{\mathrm{d}t} = -\gamma_j^2\|\boldsymbol{r}_j\|_2^2 - (\boldsymbol{u}_j^\top\boldsymbol{r}_j)^2 = -2\gamma_j^2\ell_j - (\boldsymbol{u}_j^\top\boldsymbol{r}_j)^2.$$

We also have the conservation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\gamma_j^2 - \|\boldsymbol{u}_j\|_2^2\right) = 2\gamma_j\frac{\mathrm{d}\gamma_j}{\mathrm{d}t} - 2\boldsymbol{u}_j^\top\frac{\mathrm{d}\boldsymbol{u}_j}{\mathrm{d}t} = 2\gamma_j\boldsymbol{u}_j^\top\boldsymbol{r}_j - 2\boldsymbol{u}_j^\top(\gamma_j\boldsymbol{r}_j) = 0.$$

Since $\gamma_j(0) = 1$ and $\boldsymbol{u}_j(0) = \boldsymbol{0}$, it follows that

$$\gamma_j(t)^2 - \|\boldsymbol{u}_j(t)\|_2^2 = 1.$$

Thus

$$\gamma_j(t)^2 = 1 + \|\boldsymbol{u}_j(t)\|_2^2 \geq 1. \tag{14}$$

Consequently,

$$\frac{\mathrm{d}\ell_j}{\mathrm{d}t} = -2\gamma_j^2\ell_j - (\boldsymbol{u}_j^\top\boldsymbol{r}_j)^2 \leq -2\ell_j.$$

Equivalently,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{2t}\ell_j(t)\right) = e^{2t}\left(\frac{\mathrm{d}}{\mathrm{d}t}\ell_j(t) + 2\ell_j(t)\right) = -e^{2t}\left(2(\gamma_j(t)^2 - 1)\ell_j(t) + (\boldsymbol{u}_j(t)^\top\boldsymbol{r}_j(t))^2\right) \leq 0. \tag{15}$$

Therefore,

$$\ell_j(t) \leq e^{-2t}\ell_j(0).$$

Step III: comparison of the errors. We now prove strict inequality whenever $\boldsymbol{w}_j^\star \neq \boldsymbol{0}$. For such a column,

$$\boldsymbol{u}_j(0) = \boldsymbol{0}, \qquad \boldsymbol{r}_j(0) = \boldsymbol{w}_j^\star,$$

and hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_j(0) = \gamma_j(0)\boldsymbol{r}_j(0) = \boldsymbol{w}_j^\star.$$

Thus, for all sufficiently small $s > 0$,

$$\boldsymbol{u}_j(s) = s\boldsymbol{w}_j^\star + \mathcal{O}(s^2), \qquad \boldsymbol{r}_j(s) = \boldsymbol{w}_j^\star + \mathcal{O}(s).$$

Therefore, for all sufficiently small $s > 0$,

$$\boldsymbol{u}_j(s)^\top \boldsymbol{r}_j(s) = s\|\boldsymbol{w}_j^\star\|_2^2 + \mathcal{O}(s^2) > 0.$$

Recalling (14), $\gamma_j(s)^2 \geqslant 1$. Therefore, the integrand

$$2(\gamma_j(s)^2 - 1)\ell_j(s) + (\boldsymbol{u}_j(s)^\top \boldsymbol{r}_j(s))^2$$

in (15) is strictly positive on a nontrivial interval contained in $(0, t)$ for every $t > 0$. Consequently,

$$\ell_j(t) < e^{-2t}\ell_j(0), \qquad \forall t > 0.$$

For the model g, the corresponding column-wise loss is exactly

$$\ell_{g,j}(t) = \frac{1}{2}e^{-2t}\|\boldsymbol{w}_j^\star\|_2^2 = e^{-2t}\ell_j(0).$$

Hence

$$\ell_j(t) \leq \ell_{g,j}(t)$$

for every j, and the inequality is strict for every column satisfying $\boldsymbol{w}_j^\star \neq \boldsymbol{0}$. Since $\boldsymbol{W}^\star \neq \boldsymbol{0}$, at least one such column exists. Summing over all columns gives

$$\mathcal{L}_f(\boldsymbol{U}(t), \boldsymbol{\gamma}(t)) < \mathcal{L}_g(\boldsymbol{W}(t)), \qquad \forall t > 0.$$

This proves the theorem. □

Comparison with Arora et al. (2018). A related line of work studies implicit acceleration by over-parameterization (Arora et al., 2018). It shows that deep linear networks (DLNs), $\boldsymbol{W}_L \cdots \boldsymbol{W}_1 \boldsymbol{z}$, can accelerate optimization by inducing an implicit preconditioner, without increasing the expressivity of standard linear model. Our setting and results differs in three key aspects.

- **Parameter overhead.** Scale vectors introduce substantially smaller overhead. A DLN $f_{\text{DLN}}(\boldsymbol{z}; \boldsymbol{\theta}) = \boldsymbol{W}_L \cdots \boldsymbol{W}_1 \boldsymbol{z}$ replaces a single $d_0 \cdot d_L$ linear map $\boldsymbol{W}\boldsymbol{z}$ with $\sum_{\ell=1}^{L} d_{\ell-1} \cdot d_\ell$ matrix parameters. In contrast, adding a standard input-side scale vector, $\boldsymbol{W}(\boldsymbol{\gamma} \odot \boldsymbol{z})$, introduces only d_0 additional parameters, which is negligible relative to the $d_0 \cdot d_L$ matrix parameters in linear map. It is therefore notable that such a lightweight parameterization can still improve optimization.
- **Preconditioner structure.** The induced preconditioners have different structures. DLNs induce a global spectral preconditioner aligned with the singular directions of the matrix. In contrast, scale vectors induce a channel-wise preconditioner: each input channel has its own preconditioner, which acts directly on the corresponding dynamics. Comparing the regimes in which these two preconditioners are preferable is an interesting direction for future work.
- **Initialization and early-stage dynamics.** The two parameterizations also differ in their initialization behavior. Scale vectors are typically initialized as $\mathbf{1}$, so adding them preserves the initial function. By contrast, DLNs introduce additional matrix factors; under standard small random initialization, the initial end-to-end differs from the baseline unless explicitly matched, and DLNs can be slower than the linear baseline model at early phase, as shown in (Arora et al., 2018). In contrast, scale vectors can accelerate training from the beginning, whereas DLN-induced acceleration may emerge only after the end-to-end matrix has grown sufficiently.

D.2 Proofs in Section 2.2

D.2.1 Proof of Theorem 2.3

Note that

$$\mathbb{E}[\mathsf{Norm}(\boldsymbol{x})\mathsf{Norm}(\boldsymbol{x})^\top] = \boldsymbol{I}_d.$$

For notational convenience and without loss of generality, we consider the problem

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\gamma}) = L(\boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{a}_\star\|_2^2, \quad \boldsymbol{a}_t = \boldsymbol{\gamma}_t \odot \boldsymbol{w}_t \in \mathbb{R}^d, \quad \boldsymbol{a}_\star = \boldsymbol{w}^\star. \tag{16}$$

Here $\boldsymbol{a}_t$ is the effective parameter, while $\boldsymbol{w}_t$ and $\boldsymbol{\gamma}_t$ are the two factors in the scale-vector parameterization.

Stochastic differential equation (SDE). The continuous-time modeling of Stochastic gradient descent (SGD) can be written as follows (Li et al., 2017). Let $q = \eta\sigma^2$. Then the SDE is

$$\mathrm{d}\boldsymbol{w}_t = -\big(\boldsymbol{\gamma}_t \odot \boldsymbol{g}_t + \lambda \boldsymbol{w}_t\big)\mathrm{d}t + \sqrt{q}\,\mathrm{d}\boldsymbol{B}_t^{\boldsymbol{w}}, \tag{17}$$

$$\mathrm{d}\boldsymbol{\gamma}_t = -\big(\boldsymbol{w}_t \odot \boldsymbol{g}_t + \mu \boldsymbol{\gamma}_t\big)\mathrm{d}t + \sqrt{q}\,\mathrm{d}\boldsymbol{B}_t^{\boldsymbol{\gamma}}, \tag{18}$$

where

$$\boldsymbol{g}_t = \nabla_{\boldsymbol{a}} L(\boldsymbol{a}_t) = \boldsymbol{a}_t - \boldsymbol{a}_\star,$$

and $\mathrm{d}\boldsymbol{B}_t^{\boldsymbol{w}}, \mathrm{d}\boldsymbol{B}_t^{\boldsymbol{\gamma}}$ denote independent standard Brownian motions in $\mathbb{R}^d$.

By defining the error vector

$$\boldsymbol{\epsilon}_t = \boldsymbol{a}_t - \boldsymbol{a}_\star,$$

the loss becomes

$$L_t = \frac{1}{2}\|\boldsymbol{\epsilon}_t\|_2^2.$$

Hessian sharpness metrics of the loss. Let

$$\boldsymbol{g} = \nabla_{\boldsymbol{a}} L(\boldsymbol{a}) = \boldsymbol{a} - \boldsymbol{a}_\star,$$

and define

$$\boldsymbol{D}_{\boldsymbol{w}} = \mathrm{diag}(\boldsymbol{w}), \qquad \boldsymbol{D}_{\boldsymbol{\gamma}} = \mathrm{diag}(\boldsymbol{\gamma}), \qquad \boldsymbol{D}_{\boldsymbol{g}} = \mathrm{diag}(\boldsymbol{g}).$$

Then the Hessian of $\mathcal{L}$ with respect to the parameters $(\boldsymbol{w}, \boldsymbol{\gamma})$ is

$$\nabla^2_{(\boldsymbol{w},\boldsymbol{\gamma})}\mathcal{L} = \begin{pmatrix} \boldsymbol{D}_{\boldsymbol{\gamma}}^2 & \boldsymbol{D}_{\boldsymbol{\gamma}}\boldsymbol{D}_{\boldsymbol{w}} + \boldsymbol{D}_{\boldsymbol{g}} \\ \boldsymbol{D}_{\boldsymbol{w}}\boldsymbol{D}_{\boldsymbol{\gamma}} + \boldsymbol{D}_{\boldsymbol{g}} & \boldsymbol{D}_{\boldsymbol{w}}^2 \end{pmatrix}. \tag{19}$$

Since

$$\boldsymbol{D}_{\boldsymbol{\gamma}}\boldsymbol{D}_{\boldsymbol{w}} + \boldsymbol{D}_{\boldsymbol{g}} = \mathrm{diag}(2\boldsymbol{\gamma} \odot \boldsymbol{w} - \boldsymbol{a}_*),$$

the Hessian is block diagonal across coordinates. The i-th block is

$$\boldsymbol{H}_i = \begin{pmatrix} \gamma_i^2 & 2\gamma_i w_i - a_{*,i} \\ 2\gamma_i w_i - a_{*,i} & w_i^2 \end{pmatrix}. \tag{20}$$

Therefore,

$$\nabla^2_{(\boldsymbol{w},\boldsymbol{\gamma})}\mathcal{L} = \mathrm{diag}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_d).$$

The two eigenvalues of $\boldsymbol{H}_i$ are

$$\lambda_{i,\pm} = \frac{1}{2}\left[\gamma_i^2 + w_i^2 \pm \sqrt{(\gamma_i^2 - w_i^2)^2 + 4(2\gamma_i w_i - a_{*,i})^2}\right]. \tag{21}$$

Consequently,

$$\lambda_{\max}\left(\nabla^2\mathcal{L}(\boldsymbol{w}, \boldsymbol{\gamma})\right) = \max_{1 \le i \le d} \lambda_{i,+}. \tag{22}$$

The trace is

$$\mathrm{Tr}\left(\nabla^2\mathcal{L}(\boldsymbol{w}, \boldsymbol{\gamma})\right) = \sum_{i=1}^{d}\left(\gamma_i^2 + w_i^2\right) = \|\boldsymbol{\gamma}\|_2^2 + \|\boldsymbol{w}\|_2^2. \tag{23}$$

The squared Frobenius norm is

$$\left\|\nabla^2\mathcal{L}(\boldsymbol{w},\boldsymbol{\gamma})\right\|_F^2 = \sum_{i=1}^{d}\left[\gamma_i^4 + w_i^4 + 2(2\gamma_i w_i - a_{*,i})^2\right]. \tag{24}$$

Then the formal version of Theorem 2.3 is as follows:

Theorem D.1 (Norm and sharpness Comparison)**.** *Consider $\lambda > 0$. The trajectory of (17)(18) satisfies $\sup_{t\geq 0}\mathbb{E}\|\boldsymbol{w}_t\|^2 < \infty$. Moreover,*

- *If $\mu > 0$, then $\sup_{t\geq 0}\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 < \infty$. Consequently, $\sup_{t\geq 0}\mathbb{E}[\lambda_{\max}(\nabla^2\mathcal{L}(t_k))] < \infty$, $\sup_{t\geq 0}\mathbb{E}[\mathrm{Tr}(\nabla^2\mathcal{L}(t_k))] < \infty$, $\sup_{t\geq 0}\mathbb{E}\|\nabla^2\mathcal{L}(t_k)\|_{\mathrm{F}} < \infty$.*
- *In contrast, if $\mu = 0$, then $\sup_{t\geq 0}\mathbb{E}\|\boldsymbol{\gamma}_t\|^2 = \infty$. Therefore, there exists a sequence $t_k \to \infty$ such that $\mathbb{E}\|\boldsymbol{\gamma}_{t_k}\|^2 \to \infty$. Consequently, $\mathbb{E}[\lambda_{\max}(\nabla^2\mathcal{L}(t_k))] \to \infty$, $\mathbb{E}[\mathrm{Tr}(\nabla^2\mathcal{L}(t_k))] \to \infty$, $\mathbb{E}\|\nabla^2\mathcal{L}(t_k)\|_{\mathrm{F}} \to \infty$.*

Proof. Consider the coordinate-wise scale difference

$$s_{t,i} = \gamma_{t,i}^2 - w_{t,i}^2.$$

By Itô's formula, we have

$$\mathrm{d}s_{t,i} = 2\big(\lambda w_{t,i}^2 - \mu\gamma_{t,i}^2\big)\mathrm{d}t + 2\sqrt{q}\big(\gamma_{t,i}\mathrm{d}B_{t,i}^{\boldsymbol{\gamma}} - w_{t,i}\mathrm{d}B_{t,i}^{\boldsymbol{w}}\big). \tag{25}$$

Notice that the Itô correction terms cancel because $w_{t,i}$ and $\gamma_{t,i}$ are both one-dimensional coordinates with the same noise variance q.

We first show that $\mathbb{E}\|\boldsymbol{w}_t\|_2^2$ is bounded over $t \geq 0$. Using Itô's formula,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\|\boldsymbol{w}_t\|_2^2 &= -2\lambda\mathbb{E}\|\boldsymbol{w}_t\|_2^2 - 2\mathbb{E}\left[(\boldsymbol{\gamma}_t \odot \boldsymbol{w}_t)^\top(\boldsymbol{a}_t - \boldsymbol{a}_\star)\right] + dq \\ &= -2\lambda\mathbb{E}\|\boldsymbol{w}_t\|_2^2 - 2\mathbb{E}\left[\boldsymbol{a}_t^\top(\boldsymbol{a}_t - \boldsymbol{a}_\star)\right] + dq. \end{aligned} \tag{26}$$

For any $\boldsymbol{a}$, we have

$$-2\boldsymbol{a}^\top(\boldsymbol{a} - \boldsymbol{a}_\star) = -2\|\boldsymbol{a}\|_2^2 + 2\boldsymbol{a}^\top\boldsymbol{a}_\star \leq \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2.$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\|\boldsymbol{w}_t\|_2^2 \leq -2\lambda\mathbb{E}\|\boldsymbol{w}_t\|_2^2 + \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2 + dq. \tag{27}$$

By Grönwall's inequality,

$$\mathbb{E}\|\boldsymbol{w}_t\|_2^2 \leq e^{-2\lambda t}\mathbb{E}\|\boldsymbol{w}_0\|_2^2 + \frac{dq + \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2}{2\lambda}\left(1 - e^{-2\lambda t}\right). \tag{28}$$

Thus $\mathbb{E}\|\boldsymbol{w}_t\|_2^2$ is bounded uniformly over $t \geq 0$.

- **Without weight decay on $\boldsymbol{\gamma}$ ($\mu = 0$):**

  Taking expectation in (25), we get

  $$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[s_{t,i}] = 2\lambda\mathbb{E}[w_{t,i}^2] \geq 0.$$

  Suppose, for contradiction, that there exists $M > 0$ such that

  $$\sup_{t\geq 0}\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 \leq M.$$

  Combining this with

  $$\sup_{t\geq 0}\mathbb{E}\|\boldsymbol{w}_t\|_2^2 < \infty,$$

  the occupation measures

  $$\nu_T(A) = \frac{1}{T}\int_0^T \mathbb{P}\big((\boldsymbol{w}_t, \boldsymbol{\gamma}_t) \in A\big)\mathrm{d}t$$

are uniformly tight. By the Krylov–Bogoliubov theorem, there exists an invariant probability measure π.

Let $\mathcal{A}_0$ denote the generator of the SDE when $\mu = 0$. From (25),

$$\mathcal{A}_0(\gamma_i^2 - w_i^2) = 2\lambda w_i^2.$$

Since π is invariant,

$$0 = \int \mathcal{A}_0(\gamma_i^2 - w_i^2)\mathrm{d}\pi = 2\lambda \int w_i^2 \mathrm{d}\pi. \tag{29}$$

Since $\lambda > 0$, this implies

$$w_i = 0 \quad \pi\text{-a.s.}$$

for every i. Hence

$$\boldsymbol{w} = \mathbf{0} \quad \pi\text{-a.s.}$$

Now apply the generator to $\|\boldsymbol{w}\|_2^2$. We have

$$\mathcal{A}_0\|\boldsymbol{w}\|_2^2 = dq - 2\boldsymbol{a}^\top(\boldsymbol{a} - \boldsymbol{a}_\star) - 2\lambda\|\boldsymbol{w}\|_2^2.$$

Since $\boldsymbol{w} = \mathbf{0}$ π-a.s., we also have

$$\boldsymbol{a} = \boldsymbol{\gamma} \odot \boldsymbol{w} = \mathbf{0} \quad \pi\text{-a.s.}$$

Therefore,

$$\mathcal{A}_0\|\boldsymbol{w}\|_2^2 = dq \quad \pi\text{-a.s.}$$

Using invariance again,

$$0 = \int \mathcal{A}_0\|\boldsymbol{w}\|_2^2 \mathrm{d}\pi = dq,$$

which contradicts $q > 0$. Thus

$$\sup_{t\geq 0} \mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 = \infty.$$

- **With weight decay on $\boldsymbol{\gamma}$ ($\mu > 0$):**

  By Itô's formula,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 &= -2\mu\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 - 2\mathbb{E}\left[(\boldsymbol{\gamma}_t \odot \boldsymbol{w}_t)^\top(\boldsymbol{a}_t - \boldsymbol{a}_\star)\right] + dq \\ &= -2\mu\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 - 2\mathbb{E}\left[\boldsymbol{a}_t^\top(\boldsymbol{a}_t - \boldsymbol{a}_\star)\right] + dq. \end{aligned} \tag{30}$$

  Using again

$$-2\boldsymbol{a}^\top(\boldsymbol{a} - \boldsymbol{a}_\star) \leq \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2,$$

  we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 \leq -2\mu\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 + \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2 + dq.$$

  Hence, by Grönwall's inequality,

$$\mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 \leq e^{-2\mu t}\mathbb{E}\|\boldsymbol{\gamma}_0\|_2^2 + \frac{dq + \frac{1}{2}\|\boldsymbol{a}_\star\|_2^2}{2\mu}\left(1 - e^{-2\mu t}\right).$$

  Therefore,

$$\sup_{t\geq 0} \mathbb{E}\|\boldsymbol{\gamma}_t\|_2^2 < \infty.$$

Finally, the estimates of Hessian statistics ($\lambda_{\max}(\nabla^2\mathcal{L})$, $\mathrm{Tr}(\nabla^2\mathcal{L})$, $\|\nabla^2\mathcal{L}\|_{\mathrm{F}}$) follow (22) (23) (24) naturally. □

This conclusion is related to that of Ziyin et al. (2024), who show that weight decay can induce balanced solutions in deep linear networks under stochastic dynamics. The key difference is that their analysis applies the same weight decay to all parameters, whereas our setting distinguishes between the two parameters, allowing us to isolate the effect of applying or omitting weight decay on $\boldsymbol{\gamma}$.

D.2.2 How Hessian sharpness influences loss descent

We illustrate how Hessian sharpness metrics influence the loss descent of SGD. We begin by examining the loss dynamics. Let $\mathcal{L}(\cdot)$ be twice differentiable. Consider the SGD update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\left(\nabla\mathcal{L}(\boldsymbol{\theta}_t) + \boldsymbol{\xi}_t\right),$$

where the gradient noise satisfies $\mathbb{E}[\boldsymbol{\xi}_t] = \mathbf{0}$. A second-order Taylor expansion gives:

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})] = \mathbb{E}[\mathcal{Q}(\boldsymbol{\theta}_t)] + \frac{\eta^2}{2}\mathbb{E}[\boldsymbol{\xi}_t^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\boldsymbol{\xi}_t] + o(\eta^2), \tag{31}$$

where

$$\mathcal{Q}(\boldsymbol{\theta}_t) := \mathcal{L}(\boldsymbol{\theta}_t) - \eta\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \frac{\eta^2}{2}\nabla\mathcal{L}(\boldsymbol{\theta}_t)^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\nabla\mathcal{L}(\boldsymbol{\theta}_t),$$
$$\mathcal{P}(\boldsymbol{\theta}_t) := \mathbb{E}[\boldsymbol{\xi}_t^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\boldsymbol{\xi}_t],$$

Here $\mathcal{Q}(\boldsymbol{\theta})$ denotes the deterministic GD contribution, while $\mathcal{P}(\boldsymbol{\theta})$ denotes the contribution from SGD noise.

- **GD term.** By the standard descent lemma, the deterministic GD term is controlled by the maximum eigenvalue of the Hessian:

$$\mathcal{Q}(\boldsymbol{\theta}_t) \leqslant \mathcal{L}(\boldsymbol{\theta}_t) - \eta\left(1 - \frac{\eta\lambda_{\max}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t))}{2}\right)\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2.$$

- **Noise term.** The noise contribution is $\mathcal{P}(\boldsymbol{\theta}_t) = \mathbb{E}[\boldsymbol{\xi}_t^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\boldsymbol{\xi}_t]$. Therefore, different assumptions on the SGD noise covariance $\mathbb{E}[\boldsymbol{\xi}_t\boldsymbol{\xi}_t^\top]$ lead to different Hessian statistics controlling the noise-induced loss change.
  - Classical optimization analyses (Hazan et al., 2016) often assume the *bounded noise variance*: $\mathbb{E}[\|\boldsymbol{\xi}_t\|^2] \leqslant \sigma^2$, with σ being a fixed constant. It follows that
  $$\mathcal{P}(\boldsymbol{\theta}_t) \leqslant \sigma^2\lambda_{\max}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)).$$
  - Another common assumption is the affine variance condition (Bottou et al., 2018): $\mathbb{E}[\|\boldsymbol{\xi}_t\|^2] \leqslant \sigma_0^2 + \sigma_1^2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2$. Then
  $$\mathcal{P}(\boldsymbol{\theta}_t) \leqslant (\sigma_0^2 + \sigma_1^2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2)\lambda_{\max}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)).$$
  - For regression with squared loss, several works show that the magnitude of SGD noise can be bounded by the loss value (Feng & Tu, 2021; Mori et al., 2021; Wojtowytsch, 2021; Liu et al., 2021): $\mathbb{E}[\|\boldsymbol{\xi}_t\|^2] \leqslant \sigma^2\mathcal{L}(\boldsymbol{\theta}_t)$. It follows that:
  $$\mathcal{P}(\boldsymbol{\theta}_t) \leqslant \sigma^2\mathcal{L}(\boldsymbol{\theta}_t)\lambda_{\max}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)).$$
  - A common simplification in SDE analyses of SGD is to approximate the gradient noise by isotropic Gaussian noise (Welling & Teh, 2011; Raginsky et al., 2017): $\mathbb{E}[\boldsymbol{\xi}_t\boldsymbol{\xi}_t^\top] \simeq \sigma^2\boldsymbol{I}_d$. Then
  $$\mathcal{P}(\boldsymbol{\theta}_t) \simeq \sigma^2\mathrm{Tr}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)).$$
  - In near-interpolation regimes, some works model the SGD noise covariance as approximately aligned with the Hessian (HaoChen et al., 2021; Damian et al., 2021; Li et al., 2022): $\mathbb{E}[\boldsymbol{\xi}_t\boldsymbol{\xi}_t^\top] \simeq \sigma^2\nabla^2\mathcal{L}(\boldsymbol{\theta})$. In this case,
  $$\mathcal{P}(\boldsymbol{\theta}_t) \simeq \|\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\|_{\mathrm{F}}^2.$$
  - For squared loss in near-interpolation regimes, a more refined approximation is that the SGD noise covariance aligns with both the loss value and the Hessian (Mori et al., 2022; Wu et al., 2022; Wang & Wu, 2023): $\mathbb{E}[\boldsymbol{\xi}_t\boldsymbol{\xi}_t^\top] \simeq 2\mathcal{L}(\boldsymbol{\theta}_t)\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)$. Then
  $$\mathcal{P}(\boldsymbol{\theta}_t) \simeq 2\mathcal{L}(\boldsymbol{\theta}_t)\|\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\|_{\mathrm{F}}^2.$$

  Overall, across these noise assumptions, the SGD noise contribution to the loss is governed by **Hessian sharpness metrics**, such as
  $$\lambda_{\max}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)), \quad \mathrm{Tr}(\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)), \quad \|\nabla^2\mathcal{L}(\boldsymbol{\theta}_t)\|_{\mathrm{F}}^2.$$

E PROOFS IN SECTION 3

E.1 PROOFS IN SECTION 3.2

Proof Notations. We compare the input-only scale-vector model

$$f(\boldsymbol{x};\boldsymbol{U},\boldsymbol{\gamma}) = \boldsymbol{U}(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}))$$

with the input-output scale-vector model

$$\phi(\boldsymbol{x};\boldsymbol{M},\boldsymbol{\gamma}^a,\boldsymbol{\gamma}^b) = \boldsymbol{\gamma}^a \odot \left(\boldsymbol{M}(\boldsymbol{\gamma}^b \odot \mathsf{Norm}(\boldsymbol{x}))\right).$$

Here $\boldsymbol{\gamma} \in \mathbb{R}^d$ is the input scale vector of the model f, whereas $\boldsymbol{\gamma}^a \in \mathbb{R}^c$ and $\boldsymbol{\gamma}^b \in \mathbb{R}^d$ are the output and input scale vectors of the model ϕ, respectively. Define

$$\boldsymbol{D}_\gamma = \mathrm{diag}(\boldsymbol{\gamma}), \qquad \boldsymbol{D}_a = \mathrm{diag}(\boldsymbol{\gamma}^a), \qquad \boldsymbol{D}_b = \mathrm{diag}(\boldsymbol{\gamma}^b).$$

The effective weights are

$$\boldsymbol{A}_f = \boldsymbol{U}\boldsymbol{D}_\gamma, \qquad \boldsymbol{A}_\phi = \boldsymbol{D}_a\boldsymbol{M}\boldsymbol{D}_b.$$

The corresponding losses are

$$\mathcal{L}_f(\boldsymbol{U},\boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{U}\boldsymbol{D}_\gamma - \boldsymbol{W}^\star\|_F^2,$$

and

$$\mathcal{L}_\phi(\boldsymbol{M},\boldsymbol{\gamma}^a,\boldsymbol{\gamma}^b) = \frac{1}{2}\|\boldsymbol{D}_a\boldsymbol{M}\boldsymbol{D}_b - \boldsymbol{W}^\star\|_F^2.$$

For a generic effective state $\boldsymbol{A}$, define the residual

$$\boldsymbol{R} = \boldsymbol{W}^\star - \boldsymbol{A}.$$

Theorem E.1 (Optimization advantage of input-output scale vectors)**.** *Consider the setting above. Under gradient flow, the following statements hold.*

- ***(i) Instantaneous acceleration at the same effective state.*** *Suppose that f and ϕ are evaluated at parameter values satisfying $\boldsymbol{U}\boldsymbol{D}_\gamma = \boldsymbol{D}_a\boldsymbol{M}\boldsymbol{D}_b, \boldsymbol{\gamma} = \boldsymbol{\gamma}^b$. Then their instantaneous loss derivatives satisfy*

$$\frac{d}{dt}\mathcal{L}_\phi \le \frac{d}{dt}\mathcal{L}_f \le 0.$$

- ***(ii) Local strict acceleration from zero initialization.*** *Let $\boldsymbol{U}(0) = \boldsymbol{0}, \quad \boldsymbol{M}(0) = \boldsymbol{0}, \quad \boldsymbol{\gamma}(0) = \boldsymbol{1}, \quad \boldsymbol{\gamma}^a(0) = \boldsymbol{1}, \quad \boldsymbol{\gamma}^b(0) = \boldsymbol{1}$. If $\boldsymbol{W}^\star \neq \boldsymbol{0}$, then, for all sufficiently small $t > 0$,*

$$\mathcal{L}_\phi(t) < \mathcal{L}_f(t).$$

  More precisely,

$$\mathcal{L}_\phi(t) - \mathcal{L}_f(t) = -\frac{2}{3}t^3\sum_{i=1}^{c}\|\boldsymbol{W}_{i:}^\star\|_2^4 + \mathcal{O}(t^4).$$

- ***(iii) Global strict acceleration under a matching-support teacher.*** *Assume $\boldsymbol{W}^\star \neq \boldsymbol{0}$ and that its support forms a matching, namely*

$$\|\boldsymbol{W}_{i:}^\star\|_0 \le 1 \quad \textit{for every } i, \qquad \|\boldsymbol{W}_{:j}^\star\|_0 \le 1 \quad \textit{for every } j.$$

  Under the standard initialization $\boldsymbol{U}(0) = \boldsymbol{0}, \quad \boldsymbol{M}(0) = \boldsymbol{0}, \quad \boldsymbol{\gamma}(0) = \boldsymbol{1}, \quad \boldsymbol{\gamma}^a(0) = \boldsymbol{1}, \quad \boldsymbol{\gamma}^b(0) = \boldsymbol{1}$, we have

$$\mathcal{L}_\phi(t) < \mathcal{L}_f(t), \qquad \forall t > 0.$$

Proof of Claim (i) in Theorem E.1. For f, since

$$\mathcal{L}_f = \frac{1}{2}\|\boldsymbol{U}\boldsymbol{D}_\gamma - \boldsymbol{W}^\star\|_F^2 = \frac{1}{2}\|\boldsymbol{R}\|_F^2,$$

the gradient-flow equations are

$$\dot{\boldsymbol{U}} = \boldsymbol{R}\boldsymbol{D}_\gamma, \qquad \dot{\gamma}_j = \langle \boldsymbol{U}_{:j}, \boldsymbol{R}_{:j}\rangle.$$

Therefore,

$$-\frac{d\mathcal{L}_f}{dt} = \|\boldsymbol{R}\boldsymbol{D}_\gamma\|_F^2 + \sum_{j=1}^{d}\langle \boldsymbol{U}_{:j}, \boldsymbol{R}_{:j}\rangle^2.$$

Using $\boldsymbol{U}_{:j} = \boldsymbol{A}_{:j}/\gamma_j$, we obtain

$$-\frac{d\mathcal{L}_f}{dt} = \sum_{j=1}^{d}\gamma_j^2\|\boldsymbol{R}_{:j}\|_2^2 + \sum_{j=1}^{d}\frac{\langle \boldsymbol{A}_{:j}, \boldsymbol{R}_{:j}\rangle^2}{\gamma_j^2}.$$

For ϕ, the gradient-flow equations are

$$\dot{\boldsymbol{M}} = \boldsymbol{D}_a\boldsymbol{R}\boldsymbol{D}_b,$$
$$\dot{\gamma}_i^a = \langle \boldsymbol{R}_{i:}, (\boldsymbol{M}\boldsymbol{D}_b)_{i:}\rangle,$$

and

$$\dot{\gamma}_j^b = \langle \boldsymbol{R}_{:j}, (\boldsymbol{D}_a\boldsymbol{M})_{:j}\rangle.$$

Hence

$$-\frac{d\mathcal{L}_\phi}{dt} = \|\boldsymbol{D}_a\boldsymbol{R}\boldsymbol{D}_b\|_F^2 + \sum_{i=1}^{c}\langle \boldsymbol{R}_{i:}, (\boldsymbol{M}\boldsymbol{D}_b)_{i:}\rangle^2 + \sum_{j=1}^{d}\langle \boldsymbol{R}_{:j}, (\boldsymbol{D}_a\boldsymbol{M})_{:j}\rangle^2.$$

Because

$$(\boldsymbol{M}\boldsymbol{D}_b)_{i:} = \frac{\boldsymbol{A}_{i:}}{\gamma_i^a}, \qquad (\boldsymbol{D}_a\boldsymbol{M})_{:j} = \frac{\boldsymbol{A}_{:j}}{\gamma_j^b},$$

we get

$$-\frac{d\mathcal{L}_\phi}{dt} = \sum_{i=1}^{c}\sum_{j=1}^{d}(\gamma_i^a)^2(\gamma_j^b)^2R_{ij}^2 + \sum_{i=1}^{c}\frac{\langle \boldsymbol{A}_{i:}, \boldsymbol{R}_{i:}\rangle^2}{(\gamma_i^a)^2} + \sum_{j=1}^{d}\frac{\langle \boldsymbol{A}_{:j}, \boldsymbol{R}_{:j}\rangle^2}{(\gamma_j^b)^2}.$$

Using the matched-input-scale assumption $\boldsymbol{\gamma} = \boldsymbol{\gamma}^b$, subtracting the expression for $\mathcal{D}_f$ proves

$$\frac{d\mathcal{L}_f}{dt} - \frac{d\mathcal{L}_\phi}{dt} = \sum_{i=1}^{c}\sum_{j=1}^{d}\left((\gamma_i^a)^2 - 1\right)\gamma_j^2R_{ij}^2 + \sum_{i=1}^{c}\frac{\langle \boldsymbol{A}_{i:}, \boldsymbol{R}_{i:}\rangle^2}{(\gamma_i^a)^2}.$$

Consequently, due to the conservation law of gradient flow

$$\frac{d}{dt}\left((\gamma_i^a)^2 - \|\boldsymbol{M}_{i:}\|_2^2\right) = 0,$$

we have

$$(\gamma_i^a(t))^2 = \|\boldsymbol{M}_{i:}(t)\|_2^2 + (\gamma_i^a(0))^2 - \|\boldsymbol{M}_{i:}(0)\|_2^2 \geq 1, \qquad i = 1, \ldots, c.$$

Therefore,

$$\frac{d\mathcal{L}_f}{dt} - \frac{d\mathcal{L}_\phi}{dt} \geqslant 0.$$

□

Proof of Claim (ii) in Theorem E.1. At $t = 0$, both effective weights are zero:

$$\boldsymbol{A}_f(0) = \boldsymbol{A}_\phi(0) = \boldsymbol{0}.$$

Moreover, since all scale vectors are initialized at one,

$$\dot{\boldsymbol{U}}(0) = \boldsymbol{W}^\star, \qquad \dot{\boldsymbol{M}}(0) = \boldsymbol{W}^\star.$$

Thus

$$\boldsymbol{U}(t) = t\boldsymbol{W}^\star + O(t^2), \qquad \boldsymbol{M}(t) = t\boldsymbol{W}^\star + O(t^2).$$

For the input-only model f, its input scale obeys

$$\dot{\gamma}_j = \langle \boldsymbol{U}_{:j}, \boldsymbol{R}_{:j}\rangle.$$

Hence

$$\gamma_j(t) = 1 + \frac{1}{2}\|\boldsymbol{W}^\star_{:j}\|_2^2 t^2 + O(t^3).$$

For the input-output model ϕ, the output scale obeys

$$\dot{\gamma}_i^a = \langle \boldsymbol{R}_{i:}, (\boldsymbol{M}\boldsymbol{D}_b)_{i:} \rangle.$$

Using $\boldsymbol{M}(t) = t\boldsymbol{W}^\star + O(t^2)$ and $\boldsymbol{R}(t) = \boldsymbol{W}^\star + O(t)$, we obtain

$$\gamma_i^a(t) = 1 + \frac{1}{2}\|\boldsymbol{W}^\star_{i:}\|_2^2 t^2 + O(t^3).$$

The input scale in ϕ satisfies the same second-order expansion as the input scale in f:

$$\gamma_j^b(t) = 1 + \frac{1}{2}\|\boldsymbol{W}^\star_{:j}\|_2^2 t^2 + O(t^3).$$

We now compare the instantaneous loss-decrease rates. Since

$$\boldsymbol{A}_f(t) = t\boldsymbol{W}^\star + O(t^2), \qquad \boldsymbol{A}_\phi(t) = t\boldsymbol{W}^\star + O(t^2),$$

and both residuals are

$$\boldsymbol{R}_f(t) = \boldsymbol{W}^\star + O(t), \qquad \boldsymbol{R}_\phi(t) = \boldsymbol{W}^\star + O(t),$$

a direct Taylor expansion of the gradient-flow loss-decrease rates gives

$$\mathcal{D}_\phi(t) - \mathcal{D}_f(t) = 2t^2 \sum_{i=1}^{c} \|\boldsymbol{W}^\star_{i:}\|_2^4 + O(t^3).$$

Indeed, the leading difference is exactly the row-wise output-scale contribution in ϕ, while the input-scale contributions agree up to this order. Since

$$\frac{d}{dt}\left(\mathcal{L}_\phi(t) - \mathcal{L}_f(t)\right) = -\left(\mathcal{D}_\phi(t) - \mathcal{D}_f(t)\right),$$

and $\mathcal{L}_\phi(0) = \mathcal{L}_f(0)$, integration from 0 to t gives

$$\mathcal{L}_\phi(t) - \mathcal{L}_f(t) = -\frac{2}{3}t^3 \sum_{i=1}^{c} \|\boldsymbol{W}^\star_{i:}\|_2^4 + O(t^4).$$

If $\boldsymbol{W}^\star \neq \boldsymbol{0}$, then at least one row has positive norm, so

$$\sum_{i=1}^{c} \|\boldsymbol{W}^\star_{i:}\|_2^4 > 0.$$

Therefore $\mathcal{L}_\phi(t) < \mathcal{L}_f(t)$ for all sufficiently small $t > 0$. □

Proof of Claim (iii) in Theorem E.1. Because $\boldsymbol{W}^\star$ has matching support and the zero entries are initialized at zero, the gradient-flow dynamics preserves the support. Indeed, if $W^\star_{ij} = 0$ and $M_{ij}(0) = U_{ij}(0) = 0$, then initially the effective weight at (i, j) is zero, hence the residual at (i, j) is zero. The gradient equations give

$$\dot{U}_{ij} = \gamma_j R_{ij}, \qquad \dot{M}_{ij} = \gamma_i^a R_{ij} \gamma_j^b,$$

so U_{ij} and M_{ij} remain zero. Thus the system decomposes into independent scalar problems, one for each nonzero entry of $\boldsymbol{W}^\star$.

It remains to prove the scalar claim. Let $w^\star \neq 0$. By symmetry, we assume $w^\star > 0$. The input-only model has effective weight

$$a_f = u\gamma.$$

Its gradient flow satisfies

$$\dot{u} = \gamma(w^\star - a_f), \qquad \dot{\gamma} = u(w^\star - a_f).$$

The conservation law

$$\gamma^2 - u^2 = 1$$

holds. Therefore

$$\gamma^2 + u^2 = \sqrt{1 + 4a_f^2},$$

and hence

$$\dot{a}_f = \sqrt{1 + 4a_f^2}\,(w^\star - a_f).$$

For the input-output model, the effective weight is

$$a_\phi = \gamma^a m \gamma^b.$$

The scalar gradient flow satisfies

$$\dot{m} = \gamma^a \gamma^b (w^\star - a_\phi),$$
$$\dot{\gamma}^a = m\gamma^b (w^\star - a_\phi), \qquad \dot{\gamma}^b = \gamma^a m (w^\star - a_\phi).$$

The conservation laws are

$$(\gamma^a)^2 - m^2 = 1, \qquad (\gamma^b)^2 - m^2 = 1.$$

Since $\gamma^a(0) = \gamma^b(0) = 1$, we have

$$\gamma^a(t) = \gamma^b(t) = \sqrt{1 + m(t)^2}.$$

Thus

$$a_\phi = m(1 + m^2).$$

The effective dynamics is

$$\dot{a}_\phi = (1 + 4m^2 + 3m^4)(w^\star - a_\phi).$$

We compare the two scalar speed factors at the same effective weight $a > 0$. For f,

$$q_f(a) = \sqrt{1 + 4a^2}.$$

For ϕ, let $m = m(a) > 0$ be the unique solution of

$$a = m(1 + m^2).$$

Then

$$q_\phi(a) = 1 + 4m^2 + 3m^4.$$

Let $y = m^2 > 0$. Since $a^2 = y(1 + y)^2$, we have

$$q_f(a)^2 = 1 + 4y(1 + y)^2,$$

while

$$q_\phi(a)^2 = (1 + 4y + 3y^2)^2.$$

A direct calculation gives

$$q_\phi(a)^2 - q_f(a)^2 = y(9y^3 + 20y^2 + 14y + 4) > 0.$$

Hence

$$q_\phi(a) > q_f(a), \qquad \forall a > 0.$$

Both effective weights solve scalar ODEs of the form

$$\dot{a} = q(a)(w^\star - a), \qquad a(0) = 0,$$

where $q_\phi(a) > q_f(a)$ for all $a > 0$. The scalar comparison principle implies

$$a_\phi(t) > a_f(t), \qquad \forall t > 0.$$

Both trajectories remain below $w^\star$, so

$$|w^\star - a_\phi(t)| < |w^\star - a_f(t)|.$$

Therefore

$$\frac{1}{2}(w^\star - a_\phi(t))^2 < \frac{1}{2}(w^\star - a_f(t))^2, \qquad \forall t > 0.$$

Applying this scalar comparison independently to every nonzero entry of $\boldsymbol{W}^\star$, and summing the scalar losses, yields

$$\mathcal{L}_\phi(t) < \mathcal{L}_f(t), \qquad \forall t > 0.$$

□

E.2 Proofs in Section 3.3

Proof Notations. We compare the standard scale-vector model

$$f(\boldsymbol{x};\boldsymbol{U},\boldsymbol{\gamma}) = \boldsymbol{U}(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x}))$$

with the OR-reparameterized scale-vector model

$$\psi(\boldsymbol{x};\boldsymbol{V},\boldsymbol{\alpha},\beta) = \boldsymbol{V}(\beta\mathsf{Norm}(\boldsymbol{\alpha}) \odot \mathsf{Norm}(\boldsymbol{x})).$$

The corresponding effective weights are

$$\boldsymbol{A}_f = \boldsymbol{U}\,\mathrm{diag}(\boldsymbol{\gamma}), \qquad \boldsymbol{A}_\psi = \boldsymbol{V}\,\mathrm{diag}(\beta\mathsf{Norm}(\boldsymbol{\alpha})).$$

We denote their losses by $\mathcal{L}_f$ and $\mathcal{L}_\psi$, respectively.

Theorem E.2 (Optimization advantage of OR reparameterization)**.** *Consider the setting in Theorem 2.2. We compare the standard model f with the reparameterized model ψ under gradient flow. The parameters are initialized as $\boldsymbol{\alpha}(0) = \mathbf{1}, \beta(0) = 1, \boldsymbol{V}(0) = \mathbf{0}$; $\boldsymbol{\gamma}(0) = \mathbf{1}, \boldsymbol{U}(0) = \mathbf{0}$. Then the following statements hold.*

- ***(i) Instantaneous acceleration at the same effective state.*** *For any $(\boldsymbol{U},\boldsymbol{\gamma})$ in f and $(\boldsymbol{V},\boldsymbol{\alpha},\beta)$ in ψ satisfying $\boldsymbol{U} = \boldsymbol{V}$ and $\boldsymbol{\gamma} = \beta\mathsf{Norm}(\boldsymbol{\alpha})$, the OR reparameterization is at least as fast as the direct scale-vector parameterization:*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_\psi \leqslant \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_f \leqslant 0.$$

- ***(ii) Local strict acceleration from initialization.*** *If $d > 1$, then the OR reparameterization is strictly faster in the early phase:*

$$\mathcal{L}_\psi(t) - \mathcal{L}_f(t) = -\frac{2}{3}(1 - 1/d)\|\boldsymbol{W}_\star\|_{\mathrm{F}}^4 t^3 + \mathcal{O}(t^4).$$

- ***(iii) Global strict acceleration under balanced teacher columns.*** *Suppose $d > 1$ and $\|\boldsymbol{w}_{\star,1}\|_2 = \cdots = \|\boldsymbol{w}_{\star,d}\|_2 > 0$, where $\boldsymbol{w}_{\star,j}$ denotes the j-th column of $\boldsymbol{W}_\star$. Then the OR reparameterization is strictly faster throughout training*

$$\mathcal{L}_\psi(t) < \mathcal{L}_f(t), \forall t > 0.$$

In addition, in Figure 13, we provide numerical validation of three claims in Theorem E.2. The simulations support the theoretically predicted acceleration induced by OR.

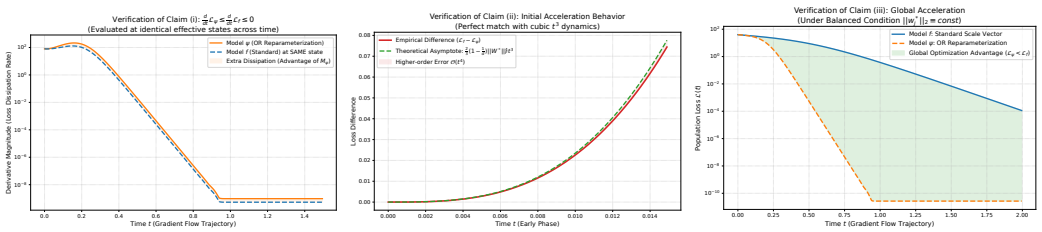


Figure 13: Numerical validation of the claims in Theorem E.2.

Proof of Claim (i) in Theorem E.2. Consider the model

$$\psi(\boldsymbol{x};\boldsymbol{V},\beta,\boldsymbol{\alpha}) = \boldsymbol{V}\big(\beta\mathsf{Norm}(\boldsymbol{\alpha}) \odot \mathsf{Norm}(\boldsymbol{x})\big),$$

where

$$\boldsymbol{V} \in \mathbb{R}^{c\times d}, \qquad \boldsymbol{\alpha} \in \mathbb{R}^d, \qquad \beta \in \mathbb{R}.$$

Let

$$\boldsymbol{q} = \mathsf{Norm}(\boldsymbol{\alpha}) = \sqrt{d}\,\frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|_2}, \qquad \boldsymbol{\gamma} = \beta\boldsymbol{q}.$$

Then

$$\psi(\boldsymbol{x}; \boldsymbol{V}, \beta, \boldsymbol{\alpha}) = \boldsymbol{V}(\boldsymbol{\gamma} \odot \mathsf{Norm}(\boldsymbol{x})).$$

Thus ψ can be viewed as a magnitude-direction reparameterization of the scale vector $\boldsymbol{\gamma}$ in f.

As before, since

$$\mathbb{E}[\mathsf{Norm}(\boldsymbol{x})\mathsf{Norm}(\boldsymbol{x})^\top] = \boldsymbol{I}_d,$$

the loss of ψ is

$$\mathcal{L}_\psi(\boldsymbol{V}, \beta, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{V}\boldsymbol{D}_{\boldsymbol{\gamma}} - \boldsymbol{W}^\star\|_F^2, \qquad \boldsymbol{D}_{\boldsymbol{\gamma}} = \mathrm{diag}(\boldsymbol{\gamma}).$$

Write

$$\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d], \qquad \boldsymbol{W}^\star = [\boldsymbol{w}_1^\star, \ldots, \boldsymbol{w}_d^\star],$$

and define

$$\boldsymbol{r}_j = \boldsymbol{w}_j^\star - \gamma_j \boldsymbol{v}_j, \qquad s_j = \boldsymbol{v}_j^\top \boldsymbol{r}_j, \qquad \boldsymbol{s} = (s_1, \ldots, s_d)^\top.$$

Then

$$\mathcal{L}_\psi = \frac{1}{2}\sum_{j=1}^{d} \|\boldsymbol{r}_j\|_2^2.$$

The gradient-flow equations are

$$\dot{\boldsymbol{v}}_j = \gamma_j \boldsymbol{r}_j,$$
$$\dot{\beta} = \boldsymbol{q}^\top \boldsymbol{s},$$

and

$$\dot{\boldsymbol{\alpha}} = \beta \boldsymbol{J}_{\boldsymbol{q}}(\boldsymbol{\alpha})^\top \boldsymbol{s},$$

where

$$\boldsymbol{J}_{\boldsymbol{q}}(\boldsymbol{\alpha}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\alpha}} = \frac{\sqrt{d}}{\|\boldsymbol{\alpha}\|_2}\left(\boldsymbol{I}_d - \frac{\boldsymbol{\alpha}\boldsymbol{\alpha}^\top}{\|\boldsymbol{\alpha}\|_2^2}\right).$$

Since $\boldsymbol{\alpha}(0) = \mathbf{1}$, we have $\|\boldsymbol{\alpha}(0)\|_2 = \sqrt{d}$. Moreover, $\dot{\boldsymbol{\alpha}}$ is always orthogonal to $\boldsymbol{\alpha}$, so

$$\|\boldsymbol{\alpha}(t)\|_2 = \sqrt{d}, \qquad \forall t \geq 0.$$

Therefore,

$$\boldsymbol{J}_{\boldsymbol{q}}(\boldsymbol{\alpha}(t)) = \boldsymbol{P}_{\boldsymbol{q}(t)} := \boldsymbol{I}_d - \frac{\boldsymbol{q}(t)\boldsymbol{q}(t)^\top}{d}.$$

Consequently,

$$\dot{\boldsymbol{q}} = \beta \boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}.$$

Since $\boldsymbol{\gamma} = \beta \boldsymbol{q}$, we obtain

$$\dot{\boldsymbol{\gamma}} = \dot{\beta}\, \boldsymbol{q} + \beta \dot{\boldsymbol{q}} = \boldsymbol{q}\boldsymbol{q}^\top \boldsymbol{s} + \beta^2 \boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}.$$

Equivalently,

$$\dot{\boldsymbol{\gamma}} = \boldsymbol{M}_\psi \boldsymbol{s}, \qquad \boldsymbol{M}_\psi := \boldsymbol{q}\boldsymbol{q}^\top + \beta^2 \boldsymbol{P}_{\boldsymbol{q}}.$$

For comparison, in the original scale-vector model f, the corresponding gradient-flow equation is simply

$$\dot{\boldsymbol{\gamma}} = \boldsymbol{s}.$$

Thus, ψ replaces the Euclidean scale-vector dynamics by a preconditioned dynamics with preconditioner

$$\boldsymbol{M}_\psi = \boldsymbol{q}\boldsymbol{q}^\top + \beta^2 \boldsymbol{P}_{\boldsymbol{q}}.$$

Moreover, ψ satisfies the conservation law

$$\frac{d}{dt}\left(\beta^2 - \|\boldsymbol{V}\|_F^2\right) = 0.$$

Indeed,

$$\frac{d}{dt}\|\boldsymbol{V}\|_F^2 = 2\sum_{j=1}^{d} \boldsymbol{v}_j^\top \dot{\boldsymbol{v}}_j = 2\sum_{j=1}^{d} \gamma_j \boldsymbol{v}_j^\top \boldsymbol{r}_j = 2\beta \boldsymbol{q}^\top \boldsymbol{s},$$

while

$$\frac{d}{dt}\beta^2 = 2\beta\dot{\beta} = 2\beta \boldsymbol{q}^\top \boldsymbol{s}.$$

Since $\beta(0) = 1$ and $\boldsymbol{V}(0) = \boldsymbol{0}$, we have

$$\beta(t)^2 - \|\boldsymbol{V}(t)\|_F^2 = 1.$$

In particular,

$$\beta(t)^2 \geq 1.$$

Now the loss dissipation of ψ is

$$\frac{d}{dt}\mathcal{L}_\psi = -\sum_{j=1}^{d} \gamma_j^2 \|\boldsymbol{r}_j\|_2^2 - (\boldsymbol{q}^\top \boldsymbol{s})^2 - \beta^2 \|\boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}\|_2^2.$$

By contrast, for f at the same effective state, the loss dissipation is

$$\frac{d}{dt}\mathcal{L}_f = -\sum_{j=1}^{d} \gamma_j^2 \|\boldsymbol{r}_j\|_2^2 - \|\boldsymbol{s}\|_2^2.$$

Decompose

$$\boldsymbol{s} = \boldsymbol{s}_\parallel + \boldsymbol{s}_\perp, \qquad \boldsymbol{s}_\parallel = \frac{\boldsymbol{q}^\top \boldsymbol{s}}{d}\boldsymbol{q}, \qquad \boldsymbol{s}_\perp = \boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}.$$

Since $\|\boldsymbol{q}\|_2^2 = d$, we have

$$(\boldsymbol{q}^\top \boldsymbol{s})^2 = d\|\boldsymbol{s}_\parallel\|_2^2.$$

Therefore,

$$(\boldsymbol{q}^\top \boldsymbol{s})^2 + \beta^2 \|\boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}\|_2^2 = d\|\boldsymbol{s}_\parallel\|_2^2 + \beta^2 \|\boldsymbol{s}_\perp\|_2^2.$$

Since $d \geq 1$ and $\beta^2 \geq 1$, this implies

$$(\boldsymbol{q}^\top \boldsymbol{s})^2 + \beta^2 \|\boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}\|_2^2 \geq \|\boldsymbol{s}_\parallel\|_2^2 + \|\boldsymbol{s}_\perp\|_2^2 = \|\boldsymbol{s}\|_2^2.$$

Hence, at the same effective state, the magnitude-direction parameterization ψ dissipates the loss at least as fast as the direct scale-vector parameterization f. □

Proof of Claim (ii) in Theorem E.2. For both models, at initialization,

$$\boldsymbol{v}_j(0) = \boldsymbol{u}_j(0) = \boldsymbol{0}, \qquad \boldsymbol{r}_j(0) = \boldsymbol{w}_j^\star.$$

Thus

$$\boldsymbol{v}_j(t) = t\boldsymbol{w}_j^\star + O(t^2), \qquad \boldsymbol{u}_j(t) = t\boldsymbol{w}_j^\star + O(t^2),$$

and hence

$$s_j(t) = \boldsymbol{v}_j(t)^\top \boldsymbol{r}_j(t) = t\|\boldsymbol{w}_j^\star\|_2^2 + O(t^2) = t\rho_j + O(t^2).$$

For f, the scale-vector dynamics are

$$\dot{\gamma}_j = s_j.$$

Therefore,

$$\gamma_j^{(f)}(t) = 1 + \frac{1}{2}\rho_j t^2 + O(t^3).$$

For ψ, at initialization we have

$$\boldsymbol{q}(0) = \boldsymbol{1}, \qquad \beta(0) = 1, \qquad \boldsymbol{P}_{\boldsymbol{q}(0)} = \boldsymbol{I}_d - \frac{1}{d}\boldsymbol{1}\boldsymbol{1}^\top.$$

The induced scale-vector dynamics are

$$\dot{\boldsymbol{\gamma}} = \boldsymbol{q}\boldsymbol{q}^\top \boldsymbol{s} + \beta^2 \boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s}.$$

Thus, at leading order,

$$\dot{\gamma}_j^{(\psi)}(t) = t\rho_j + \left(1 - \frac{1}{d}\right) tS + O(t^2).$$

Hence

$$\gamma_j^{(\psi)}(t) = 1 + \frac{1}{2}\left[\rho_j + \left(1 - \frac{1}{d}\right) S\right] t^2 + O(t^3).$$

Let

$$D_f(t) := -\frac{d}{dt}\mathcal{L}_f(t), \qquad D_\psi(t) := -\frac{d}{dt}\mathcal{L}_\psi(t).$$

Using the loss-dissipation identities,

$$D_f(t) = \sum_{j=1}^{d} \left(\gamma_j^{(f)}(t)\right)^2 \|\boldsymbol{r}_j^{(f)}(t)\|_2^2 + \|\boldsymbol{s}^{(f)}(t)\|_2^2,$$

while

$$D_\psi(t) = \sum_{j=1}^{d} \left(\gamma_j^{(\psi)}(t)\right)^2 \|\boldsymbol{r}_j^{(\psi)}(t)\|_2^2 + (\boldsymbol{q}(t)^\top \boldsymbol{s}^{(\psi)}(t))^2 + \beta(t)^2 \|\boldsymbol{P}_{\boldsymbol{q}(t)} \boldsymbol{s}^{(\psi)}(t)\|_2^2.$$

Substituting the expansions above gives

$$D_\psi(t) - D_f(t) = 2\left(1 - \frac{1}{d}\right) S^2 t^2 + O(t^3).$$

Therefore,

$$\begin{aligned}\mathcal{L}_\psi(t) - \mathcal{L}_f(t) &= -\int_0^t \left(D_\psi(s) - D_f(s)\right) ds \\ &= -\frac{2}{3}\left(1 - \frac{1}{d}\right) S^2 t^3 + O(t^4).\end{aligned}$$

This proves the claim. □

Proof of Claim (iii) in Theorem E.2. Under the balanced condition and the symmetric initialization

$$\boldsymbol{q}(0) = \mathbf{1},$$

the symmetry is preserved along the gradient flow of ψ. Indeed, the vector $\boldsymbol{s}$ remains proportional to $\mathbf{1}$, and hence

$$\boldsymbol{P}_{\boldsymbol{q}} \boldsymbol{s} = \mathbf{0}.$$

Therefore,

$$\boldsymbol{q}(t) = \mathbf{1}, \qquad \boldsymbol{\gamma}(t) = \beta(t)\mathbf{1}.$$

For both models, each column remains aligned with its teacher column. Thus we may write

$$\boldsymbol{u}_j(t) = p_f(t)\boldsymbol{w}_j^\star$$

for f, and

$$\boldsymbol{v}_j(t) = p_\psi(t)\boldsymbol{w}_j^\star$$

for ψ.

For f, the scalar dynamics are

$$\dot{p}_f = \gamma_f(1 - \gamma_f p_f), \qquad \dot{\gamma}_f = \rho p_f(1 - \gamma_f p_f),$$

with

$$p_f(0) = 0, \qquad \gamma_f(0) = 1.$$

For ψ, the scalar dynamics are

$$\dot{p}_\psi = \beta(1 - \beta p_\psi), \qquad \dot{\beta} = d\rho p_\psi(1 - \beta p_\psi),$$

with

$$p_\psi(0) = 0, \qquad \beta(0) = 1.$$

Both dynamics have the same form

$$\dot{p} = \eta(1 - \eta p), \qquad \dot{\eta} = k\rho p(1 - \eta p),$$

where $k = 1$ for f and $k = d$ for ψ.

The corresponding invariant is

$$\eta^2 - k\rho p^2 = 1.$$

Let

$$\theta = \eta p.$$

Then

$$\dot{\theta} = (\eta^2 + k\rho p^2)(1 - \theta).$$

Using the invariant, set

$$z = k\rho p^2.$$

Then

$$\eta^2 = 1 + z, \qquad \theta^2 = \eta^2 p^2 = \frac{z(1+z)}{k\rho}.$$

Therefore,

$$z = \frac{\sqrt{1 + 4k\rho\theta^2} - 1}{2}.$$

Hence the effective coefficient θ satisfies the closed scalar ODE

$$\dot{\theta} = \left[1 + 2 \cdot \frac{\sqrt{1 + 4k\rho\theta^2} - 1}{2}\right](1 - \theta).$$

Equivalently,

$$\dot{\theta} = \sqrt{1 + 4k\rho\theta^2}\,(1 - \theta).$$

For ψ, $k = d$, while for f, $k = 1$. Since $d > 1$, we have

$$\sqrt{1 + 4d\rho\theta^2} > \sqrt{1 + 4\rho\theta^2}$$

for every $\theta \in (0, 1)$. By the scalar comparison principle,

$$\theta_\psi(t) > \theta_f(t), \qquad \forall t > 0.$$

The residual coefficient is $1 - \theta$, so

$$1 - \theta_\psi(t) < 1 - \theta_f(t).$$

Therefore,

$$\mathcal{L}_\psi(t) = \frac{d\rho}{2}(1 - \theta_\psi(t))^2 < \frac{d\rho}{2}(1 - \theta_f(t))^2 = \mathcal{L}_f(t), \qquad \forall t > 0.$$

This proves the claim. □