Spectral Condition for μP under Width–Depth Scaling

Anonymous authors
Paper under double-blind review

Abstract

Generative foundation models are increasingly scaled in both width and depth, posing significant challenges for stable feature learning and reliable hyperparameter (HP) transfer across model sizes. While maximal update parameterization (μP) has provided a principled solution to both problems for width scaling, existing extensions to the joint width–depth scaling regime remain fragmented, architecture- and optimizer-specific, and often rely on technically involved theories. In this work, we develop a simple and unified spectral framework for μP under joint width–depth scaling. For deep residual networks whose residual blocks contain k transformations, the framework specifies how the norms of weights and their per-step updates should scale with width and depth. It reveals a fundamental transition from $k = 1$ to $k \geq 2$, unifying previously disparate μP formulations and identifying the $k \geq 2$ case as more appropriate for practical architectures with multi-transformation branches such as Transformers. Building on this framework, we derive a general recipe for implementing μP across a broad class of optimizers by mapping spectral constraints to concrete HP parameterizations, recovering existing results and extending them to additional optimizers. Finally, experiments on GPT-2 style language models show that the μP formulation derived from the $k \geq 2$ case achieves stable feature learning and robust HP transfer under width–depth scaling, whereas standard parameterization and μP in the $k = 1$ case often fail to do so. These results support the practical effectiveness of the proposed spectral framework.

1 Introduction

Generative foundation models have been rapidly scaling in *both width and depth* (Kaplan et al., 2020; Hoffmann et al., 2022; Liu et al., 2024a; Singh et al., 2025; Yang et al., 2025), and this trend is expected to continue in the foreseeable future as datasets grow and task complexity increases. However, when model sizes become sufficiently large (e.g., billions of parameters), feature learning dynamics often become unstable or degenerate (Schoenholz et al., 2017; Jacot et al., 2018), and the hyperparameter (HP) tuning becomes prohibitively expensive (Yang et al., 2022). These issues pose fundamental obstacles to efficient scaling, underscoring the need for principled methods enabling stable feature learning and reliable HP transfer from small models to larger ones.

Maximal update parameterization (μP) (Yang & Hu, 2021; Yang et al., 2022) was originally proposed to address both challenges for width scaling, and has recently been preliminarily extended to settings that jointly scale width and depth (Yang et al., 2024; Bordelon et al., 2024b;a; Dey et al., 2025; Qiu et al., 2025). By appropriately reparameterizing HPs with model size, μP aims to preserve scale-invariant feature learning, while maximizing the feature change induced by parameter updates, leading to stable and efficient training dynamics (Yang & Hu, 2021; Dey et al., 2025). Moreover, μP empirically stabilizes optimal HPs across different scales, enabling direct transfer of HPs tuned on small models to much larger ones (Yang et al., 2022; Zheng et al., 2025).

However, in the joint width–depth scaling regime, existing μP formulations remain preliminary. They are often tightly coupled to specific architectural choices, such as the internal depth of residual blocks (Yang et al., 2024; Bordelon et al., 2024b;a; Dey et al., 2025) and particular optimization algorithms (Yang et al., 2024; Bordelon et al., 2024b; Dey et al., 2025; Qiu et al., 2025). Moreover, their derivations typically rely on technically involved tools such as Tensor Programs (Yang & Littwin, 2023; Yang et al., 2024) or dynamical

mean-field theory (Bordelon et al., 2024a;b). Consequently, it remains difficult for the community to both systematically understand existing results and extend the μP principle to new architectures and optimizers, highlighting the need for a simple and unified theoretical framework.

To address the challenges outlined above, we draw inspiration from the unified spectral perspective developed for width-scaling μP (Yang et al., 2023). We extend this spectral perspective to the joint width–depth scaling regime, yielding a simple, unified framework for realizing the μP principle in deep residual networks and systematically deriving μP formulations across a broad class of optimizers. Our main contributions are summarized as follows.

First, we introduce a unified spectral scaling framework for realizing the μP principle in deep residual networks with fixed k-layer residual blocks under width-depth scaling. It specifies how the RMS operator norms of weights and per-step updates should scale with model size. Across architectures with different fixed residual-block depth ks, the spectral condition changes fundamentally from $k = 1$ (Condition B.1) to $k = 2$ (Condition 3.1) due to the emergence of higher-order update terms, but yields no essential further change in the resulting μP formulation for $k \geq 2$. Besides, Condition B.1 recovers Depth-μP-style results (Yang et al., 2024; Bordelon et al., 2024b), while Condition 3.1 recovers CompleteP-style results (Bordelon et al., 2024a; Dey et al., 2025; Qiu et al., 2025), which our theory suggests are more appropriate for practical architectures with multi-layer residual branches such as Transformers. Notably, our analysis uses only elementary linear algebra and probability, making it easier to follow than previous works.

Second, building on the proposed spectral condition, we present a unified recipe for implementing μP across a broad class of optimizers by mapping the spectral constraints to concrete HP parameterizations. Concretely, we systematically derive μP parameterizations for Muon-Kimi (Liu et al., 2025), Muon (Jordan et al., 2024b), Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2024), AdamW (Loshchilov & Hutter, 2019), Sophia (Liu et al., 2024b), Lion (Chen et al., 2023), SGD, and Spectral Sphere Optimizer (SSO) (Xie et al., 2026). These parameterizations are derived from the optimizers' update rules rather than ad hoc tuning heuristics; they also recover important existing width-depth μP formulations (Yang et al., 2024; Bordelon et al., 2024b;a; Dey et al., 2025; Qiu et al., 2025) as special cases.

Finally, through controlled experiments on GPT-2 style Transformer language models (Radford et al., 2019; Karpathy, 2022), we empirically demonstrate that the μP formulation derived from Condition 3.1 achieves stable feature learning and robust HP transfer under joint width-depth scaling. In contrast, standard parameterization (SP) and the μP formulation derived from Condition B.1 are often unstable or fail to transfer HPs reliably. Together, these results support the practical effectiveness of the proposed spectral framework in realistic pretraining settings.

2 Preliminaries

We begin by establishing the necessary background for mathematical techniques and μP. Additional related work is discussed in Appendix A.

2.1 Mathematical Notations and Properties

Scaling notation. Let f and g be positive functions of the scaling variables (e.g., width and depth). We use $f = \mathcal{O}(g)$, $f = \Omega(g)$, and $f = \Theta(g)$ in the standard asymptotic sense. In width-scaling settings, the asymptotics are taken with respect to width; in width-depth scaling settings, they are taken with respect to jointly increasing width and depth. Fixed quantities, such as data dimensions, are hidden in the constants.

Vector and matrix norms. We define $[n] = \{1, 2, \ldots, n\}$. For a vector $\boldsymbol{a} \in \mathbb{R}^n$, we use $\|\boldsymbol{a}\|_2$ and $\|\boldsymbol{a}\|_{\mathrm{R}}$ to denote its ℓ_2 norm and Root Mean Square (RMS) norm, respectively. By definition, we have $\|\boldsymbol{a}\|_{\mathrm{R}} = \|\boldsymbol{a}\|_2/\sqrt{n}$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, we use $\|\boldsymbol{A}\|_2$, and $\|\boldsymbol{A}\|_{\mathrm{R}}$ to denote its spectral norm and RMS operator norm, respectively. The RMS operator norm is defined as $\|\boldsymbol{A}\|_{\mathrm{R}} := \max_{\boldsymbol{v}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{v}\|_{\mathrm{R}}}{\|\boldsymbol{v}\|_{\mathrm{R}}} = \sqrt{\frac{n}{m}}\,\|\boldsymbol{A}\|_2$. Since spectral norm conditions can be equivalently expressed using the RMS operator norm, we adopt the latter to write spectral conditions for notational simplicity throughout this paper. Finally, in the main text, we primarily rely on the following elementary properties of vector and matrix norms.

- **Subadditivity:** $\|\boldsymbol{A}+\boldsymbol{B}\|_{\mathrm{R}} \leq \|\boldsymbol{A}\|_{\mathrm{R}} + \|\boldsymbol{B}\|_{\mathrm{R}}$ and $\|\boldsymbol{a}+\boldsymbol{b}\|_{\mathrm{R}} \leq \|\boldsymbol{a}\|_{\mathrm{R}} + \|\boldsymbol{b}\|_{\mathrm{R}}$.
- **Submultiplicativity:** $\|\boldsymbol{AB}\|_{\mathrm{R}} \leq \|\boldsymbol{A}\|_{\mathrm{R}}\|\boldsymbol{B}\|_{\mathrm{R}}$ and $\|\boldsymbol{Av}\|_{\mathrm{R}} \leq \|\boldsymbol{A}\|_{\mathrm{R}}\|\boldsymbol{v}\|_{\mathrm{R}}$.
- **Spectral norm of random matrices** (Vershynin, 2018): for a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with i.i.d. entries sampled from Gaussian distribution $\mathcal{N}(0,\sigma^2)$, its spectral norm satisfies $\|\boldsymbol{A}\|_2 = \Theta\big(\sigma(\sqrt{m}+\sqrt{n})\big)$ with high probability.

2.2 Spectral Condition for μP under Width Scaling

We briefly review μP and its spectral condition under width scaling (Yang et al., 2023), which serves as the conceptual foundation of our extension to joint width–depth scaling.

Theoretical setup. A canonical setting (Yang et al., 2023) for analyzing μP under width scaling is the deep linear multi-layer perceptron (MLP) trained with one step on a single data point $(\boldsymbol{x},\boldsymbol{y})$. Specifically, we set $\boldsymbol{h}_0(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x}$ and denote by $\boldsymbol{W}_l$ the matrix weight at layer l. The network is then defined as

$$\boldsymbol{h}_l(\boldsymbol{x}) = \boldsymbol{W}_l\boldsymbol{h}_{l-1}(\boldsymbol{x}), \quad l \in [L+1],$$

where the depth $L = \Theta(1)$ is fixed, while the model widths scale to infinity. Although highly simplified, this setup captures the core width-scaling behavior of feature learning (Yang et al., 2023). Moreover, μP formulations motivated by this setup have been successfully used in practical pretraining (Yang et al., 2022; Hu et al., 2024; Zheng et al., 2025), including Transformers trained with AdamW, enabling stable feature learning and reliable HP transfer.

μP principle and its spectral condition under width scaling. As network size increases, standard parameterization (SP) typically leads to either exploding or vanishing feature updates. μP resolves this issue by reparameterizing HPs with size to realize the following stable and efficient principle (Yang & Hu, 2021).

Principle 2.1 (μP principle)**.** *μP aims to realize scale-invariant feature learning while maximizing the feature change induced by parameter updates. Formally, it requires*

$$\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1),\ \|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1),\ l \in [L]. \tag{P1}$$

$$\textit{maximize } \Delta\boldsymbol{W}_l\textit{'s contribution to } \Delta\boldsymbol{h}_L(\boldsymbol{x}),\ l \in [L]. \tag{P2}$$

Under the width-scaling regime, Yang et al. (2023) showed that Principle 2.1 is ensured by the following simple spectral scaling condition on the weights and their per-step updates:

$$\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1),\ \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1),\ l \in [L+1]. \tag{1}$$

This spectral condition provides a concise and unified perspective on μP under width scaling, from which the HP parameterization of a broad class of optimization algorithms can be derived in a unified and transparent manner (Yang et al., 2023; Haas et al., 2024; Ngom et al., 2025).

Limitation of the width-scaling condition. The spectral condition (1), however, applies only when depth is fixed. In contrast, modern foundation models scale both width and depth, and existing μP results in this regime rely on complex analyses, with conclusions that depend on specific architectures and optimizers (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025; Qiu et al., 2025). This motivates our central question: *Can we establish a simple and unified spectral perspective in the joint width–depth scaling regime?*

3 Spectral Condition for μP under Width-Depth Scaling

In this section, we establish the spectral condition for μP under width-depth scaling. We first introduce our problem setup, then derive the corresponding spectral μP condition and discuss its implications.

3.1 Problem Setup

Our setup extends the width-scaling setting in Section 2.2 by introducing residual connections, which are essential for stabilizing training of deep networks (He et al., 2016). Motivated by practical residual branches that often contain multiple transformations (e.g., attention or FFN modules in Transformers), we consider residual networks whose residual branches contain $k = \Theta(1)$ linear transformations. For clarity, the main text focuses on the two-layer residual block ($k = 2$), which is the minimal setting that captures the essential scaling behavior of any fixed-depth branches with $k \geq 2$ while keeping the analysis minimal. The $k = 1$ case and the general $k \geq 2$ case are deferred to Appendices B.1 and B.2, respectively. Formally, the $k = 2$ network studied in the main text is defined as:

$$\begin{aligned} \boldsymbol{h}_0(\boldsymbol{x}) &= \alpha_0 \boldsymbol{W}_0 \boldsymbol{x}, \\ \boldsymbol{h}_l(\boldsymbol{x}) &= \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}), \ l \in [L] \\ \boldsymbol{h}_{L+1}(\boldsymbol{x}) &= \alpha_{L+1} \boldsymbol{W}_{L+1} \boldsymbol{h}_L(\boldsymbol{x}), \end{aligned} \tag{2}$$

where the weights $\boldsymbol{W}_0 \in \mathbb{R}^{n \times d_0}, \boldsymbol{W}_l^{(1)} \in \mathbb{R}^{n_l \times n}, \boldsymbol{W}_l^{(2)} \in \mathbb{R}^{n \times n_l}, \boldsymbol{W}_{L+1} \in \mathbb{R}^{d_{L+1} \times n}$ are all initialized with Gaussian distribution $(\boldsymbol{W}_l)_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_l^2)$[1] and trained with layerwise learning rates η_l. Furthermore, $\{\alpha_l\}_{l=0}^{L+1}$ are block multipliers that control the effective strength of each transformation.

Following existing μP literature (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025), we fix the input and output data dimensions and scale the width and depth to infinity, that is

$$d_0, d_{L+1} = \Theta(1), \quad n_l = \Theta(n), \quad n, L \to \infty. \tag{3}$$

This setting is standard in Transformer-based large models (Vaswani et al., 2017; Singh et al., 2025), where n denotes the model width and is typically of the same order as n_l (e.g., the feed-forward width). Moreover, we assume $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$, which holds for common data modalities such as natural images ($\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$) and one-hot encoded language inputs ($\|\boldsymbol{x}\|_{\mathrm{R}} = \sqrt{1/d_0} = \Theta(1)$).

In the following subsections, we derive a sufficient and unified spectral condition under this $k = 2$ setup for realizing the μP Principle 2.1 under joint width-depth scaling.

3.2 Spectral Scaling Condition

Analogous to the width-scaling condition in Equation (1), the width-depth condition has two components. The initial condition on $\boldsymbol{W}_l$ controls forward feature propagation, yielding $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ across depth. The update condition on $\Delta \boldsymbol{W}_l$ controls the one-step feature change, ensuring $\|\Delta \boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while maximizing the direct contribution of weight updates as required by Principle (P2). We now state the resulting sufficient spectral condition.

Condition 3.1 (Spectral condition for μP under joint width-depth scaling, two-layer residual block)**.** *To realize μP Principle 2.1, the initial weights and their per-step updates should satisfy:*

- ***Initial condition.***

  Input and output weights:

$$\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1), \ \alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1). \tag{C1.1}$$

  Hidden weights:

$$\alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \, \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \ l \in [L]. \tag{C1.2}$$

- ***Update condition.***

[1]For notation simplicity, when quantities associated with $\boldsymbol{W}_l^{(1)}$ and $\boldsymbol{W}_l^{(2)}$ take the same form, we omit the superscript.

Input and output weights:

$$\alpha_0\|\Delta \boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1),\ \alpha_{L+1}\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1). \tag{C2.1}$$

Hidden weights (first-order weight update):

$$\begin{aligned} \alpha_l\|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L),\ l \in [L] \\ \alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L),\ l \in [L]. \end{aligned} \tag{C2.2}$$

Hidden weights (second-order weight update):

$$\alpha_l\|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L),\ l \in [L]. \tag{C2.3}$$

Compared with the width-only condition in Equation (1), Condition 3.1 introduces explicit depth factors for hidden residual blocks. In particular, the products of the residual multiplier and the relevant weight or update norms must scale as $\Theta(1/L)$, reflecting the accumulation of residual contributions over L blocks.

The hidden-layer update constraints in Condition 3.1 arise from expanding the one-step feature update $\Delta \boldsymbol{h}_l(\boldsymbol{x})$. For a two-layer residual block, this expansion contains first-order terms like $\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)}$, where exactly one branch weight is updated, and a second-order term including $\Delta \boldsymbol{W}_l^{(2)} \Delta \boldsymbol{W}_l^{(1)}$, where both branch weights are updated; these give rise to (C2.2) and (C2.3), respectively. By contrast, a one-layer residual block produces only first-order direct update terms, so the second-order constraint is absent.

This update-order viewpoint helps unify prior disparate μP results under joint width-depth scaling (Yang et al., 2024; Bordelon et al., 2024b; Dey et al., 2025; Bordelon et al., 2024a; Qiu et al., 2025) by varying the residual block depth k. When $k = 1$, the update expansion contains no second-order term, and the same analysis gives Condition B.1 in Appendix B.1. The resulting looser constraints naturally lead to residual multipliers $\alpha_l = \Theta(1/\sqrt{L})$ under standard width-scaling μP initialization, which recovers Depth-μP-style results (Yang et al., 2024; Bordelon et al., 2024b). We defer the details to Appendix B.1 and D. In contrast, the $k = 2$ case introduces the second-order constraint (C2.3), which tightens the hidden residual scaling and leads to residual multipliers $\alpha_l = \Theta(1/L)$ under the same initialization convention. This recovers CompleteP-style results (Bordelon et al., 2024a; Dey et al., 2025; Qiu et al., 2025), which are more appropriate for practical architectures with multi-transformation residual branches (e.g., Transformers). Detailed HP parameterizations are given in Section 4 and Appendix C.

Moreover, our analysis naturally extends to residual blocks with any fixed depth k (Condition B.2 in Appendix B.2) and to architectures with biases (Condition B.3 in Appendix B.3). For residual blocks of depth k, the update condition constrains all first- through k-th order update terms to scale as $\Theta(1/L)$. Analogous conditions arise in the presence of biases, accounting for their interactions with weight updates. As shown in Appendix B.2 and B.3, these additional constraints do not lead to a different μP formulation compared to the $k = 2$ case. Therefore, *the residual network with two-layer blocks in Equation (2) is the minimal setting that captures the core scaling behavior of practical architectures with multi-layer residual blocks.*

Although the spectral results are derived for a linear residual MLP with a one-step update, they generalize to more general and practical training regimes. Theoretically, we introduce and verify some natural assumptions from Yang et al. (2023) in Appendix G, under which the spectral results generalize to finite multiple gradient steps, nonlinearities, and finite multiple training examples. Empirically, experiments in Section 5 demonstrate that the resulting μP formulations from Condition 3.1 achieve stable feature learning and reliable HP transfer on GPT-2 style models. Together with prior empirical μP studies (Dey et al., 2025; Qiu et al., 2025), these results suggest that the simplified setup captures the core scaling behavior relevant in practice.

Takeaway 1. Residual block depth k determines the depth μP scaling rule: $k = 1$ gives the Depth-μP-style scaling, whereas $k \geq 2$ introduces high-order update terms and requires the stricter CompleteP-style scaling. The latter is better suited to practical architectures such as Transformers.

3.3 Theoretical Derivation

In this section, we derive Condition 3.1 for the residual network in Equation (2). We first obtain a preliminary initialization condition from forward feature stability, which controls the accumulated residual branch at initialization. We then expand the one-step feature update into zero-, first-, and second-order terms. The first- and second-order terms should satisfy the maximal-update requirement in Principle (P2) while keeping the total feature update stable. In particular, the second-order term, absent in one-layer residual blocks, yields the additional constraint (C2.3). Finally, combining these update constraints refines the preliminary initialization condition to Condition 3.1.

Throughout the derivation, we use norm estimates based on subadditivity and submultiplicativity to track the typical scales of $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ and $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$. For example, under standard random initialization, we use estimates of the form $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \alpha_0\|\boldsymbol{W}_0\boldsymbol{x}\|_{\mathrm{R}} = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}\|\boldsymbol{x}\|_{\mathrm{R}})$. These estimates should be understood as scale estimates whose tightness relies on standard non-cancellation and alignment behavior in neural network initialization and training. We discuss the tightness justification under width-depth scaling in Appendix F, following the width-scaling treatment of Yang et al. (2023).

3.3.1 Preliminary Initial Condition

We first derive a preliminary initialization condition that ensures stability of feature magnitudes during forward propagation. We consider each layer sequentially.

Input layer. By submultiplicativity of the RMS operator norm, we can estimate the norm of $\boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0\boldsymbol{W}_0\boldsymbol{x}$ as $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}\|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}})$, where we have assumed $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$. Thus, requiring $\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ ensures $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. To estimate the scale of hidden features, we expand the residual recursion in Equation (2), which yields

$$\boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x}),\ s \in [L]. \tag{4}$$

Applying subadditivity together with the scale-estimation convention described above, we estimate

$$\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\bigg(\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \Big\|\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\Big\|_{\mathrm{R}}\bigg).$$

Since we have $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, it suffices to ensure that $\|\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$ to preserve $\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. Under i.i.d. zero-mean Gaussian initialization, the summands are approximately independent zero-mean random vectors (Yang & Hu, 2021; Yang et al., 2024; Dey et al., 2025), so the typical squared RMS norm of their sum scales as the sum of the squared RMS norms (see Theorem 3.3.1 in Vershynin (2018)), yielding that $\|\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\sqrt{\sum_{l=1}^{s} \|\alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2})$. By submultiplicativity, we can further estimate $\|\alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}})$. Therefore, starting from $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, imposing

$$\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}),\ l \in [L],$$

recursively ensures $\|\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$. This provides a preliminary initial condition on the hidden weights, which will be further refined by incorporating update constraints established in the next subsection.

Output layer. Submultiplicativity gives $\|\boldsymbol{h}_{L+1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}}\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}) = \Theta(\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}})$, where $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ follows from the hidden-layer argument above. Thus choosing $\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$ keeps the output stable.

3.3.2 Update Condition

We next derive the update condition required to ensure stable feature evolution $\|\Delta \boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ in Principle (P1), while maximally updating parameters as prescribed by Principle (P2).

Input layer. Since $\Delta \boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0 \Delta \boldsymbol{W}_0 \boldsymbol{x}$, submultiplicativity of matrix norms yields

$$\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}} \|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}}),$$

and hence we set $\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ to ensure $\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. To analyze the hidden feature updates $\Delta \boldsymbol{h}_s(\boldsymbol{x})$, we expand the residual representation in Equation (4) after a single gradient step: $\boldsymbol{h}_s(\boldsymbol{x}) + \Delta \boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_0(\boldsymbol{x}) + \Delta \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l (\boldsymbol{W}_l^{(2)} + \Delta \boldsymbol{W}_l^{(2)})(\boldsymbol{W}_l^{(1)} + \Delta \boldsymbol{W}_l^{(1)})(\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x}))$, leading to

$$\begin{aligned}
\Delta \boldsymbol{h}_s(\boldsymbol{x}) = \Delta \boldsymbol{h}_0(\boldsymbol{x}) &+ \underbrace{\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})}_{\boldsymbol{\epsilon}_0(s)} + \underbrace{\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \Delta \boldsymbol{W}_l^{(1)} (\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(1)}(s)} \\
&+ \underbrace{\sum_{l=1}^{s} \alpha_l \Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} (\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(2)}(s)} + \underbrace{\sum_{l=1}^{s} \alpha_l \Delta \boldsymbol{W}_l^{(2)} \Delta \boldsymbol{W}_l^{(1)} (\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_2(s)}.
\end{aligned}$$

According to the degree of weight updates in the current residual block, we refer to these contributions as the zero-, first-, and second-order update terms, denoted respectively by $\boldsymbol{\epsilon}_0(s)$, $\boldsymbol{\epsilon}_1^{(1)}(s), \boldsymbol{\epsilon}_1^{(2)}(s)$, and $\boldsymbol{\epsilon}_2(s)$. Using subadditivity together with the scale-estimation convention above, we have

$$\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\big(\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(1)}(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(2)}(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_2(s)\|_{\mathrm{R}}\big). \tag{5}$$

Since $\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the input-layer update, we have $\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Omega(1)$ for all $s \in [L]$. Moreover, by subadditivity estimation, the order of remaining terms do not decay with depth, implying $\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}})$ for any $s \in [L]$. Therefore, to enforce Principle 2.1, it suffices to require $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while satisfying Principle (P2). We next control terms on the right-hand side of Equation (5).

Zero-order term. The term $\boldsymbol{\epsilon}_0(L)$ propagates feature updates from earlier layers and does not depend on the weight update $\Delta \boldsymbol{W}_l$ at the current layer, so it does not need to be maximized from Principle (P2). Therefore, it suffices to verify that $\boldsymbol{\epsilon}_0(L)$ remains $\mathcal{O}(1)$ under the preliminary initial condition. In fact, the same argument used for deriving $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ in Section 3.3.1 directly implies $\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} = \Theta(\sqrt{\sum_{l=1}^{L} \alpha_l^2 \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}^2 \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}^2 \|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2}) = \mathcal{O}(1)$, where we use the self-consistent fact that $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally ensure $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

First-order terms. Using subadditivity and submultiplicativity, we estimate the order of $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}}$ as

$$\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} = \Theta\Big(\sum_{l=1}^{L} \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\Big) + \Theta\Big(\sum_{l=1}^{L} \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\Big).$$

For $l \in [L]$, using $\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the preliminary initial condition and $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ if we finally set $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, we can obtain $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} = \Theta\big(\sum_{l=1}^{L} \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}\big)$. To satisfy Principle (P2), we need to maximize the contribution from each $\Delta \boldsymbol{W}_l^{(1)}$ and ensure $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} = \Theta(1)$ at the same time, which naturally requires

$$\alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \qquad \forall\, l \in [L].$$

To control the scale of $\boldsymbol{\epsilon}_1^{(2)}(L)$, an identical argument gives $\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$ for every $l \in [L]$, which completes the first-order update condition (C2.2).

Second-order term. *This is the key term that distinguishes two-layer residual branches from one-layer residual branches because it is absent when $k = 1$.* As a direct update term, it is also subject to Principle (P2). Using subadditivity and submultiplicativity inequalities as for deriving $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}}$, we can estimate its scale as

$$\|\boldsymbol{\epsilon}_2(L)\|_{\mathrm{R}} = \Theta\Big(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}\Big).$$

Principle (P2) requires maximizing each summand and ensuring $\|\boldsymbol{\epsilon}_2(L)\|_{\mathrm{R}} = \Theta(1)$ in the meanwhile, leading to $\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$ for all $l \in [L]$, which completes the derivation for the second-order update condition on hidden weights (C2.3).

Output layer. For the output layer $\boldsymbol{h}_{L+1}(\boldsymbol{x}) = \alpha_{L+1} \boldsymbol{W}_{L+1} \boldsymbol{h}_L(\boldsymbol{x})$, its one-step feature update is

$$\Delta \boldsymbol{h}_{L+1}(\boldsymbol{x}) = \alpha_{L+1} \boldsymbol{W}_{L+1} \Delta \boldsymbol{h}_L(\boldsymbol{x}) + \alpha_{L+1} \Delta \boldsymbol{W}_{L+1} \big(\boldsymbol{h}_L(\boldsymbol{x}) + \Delta \boldsymbol{h}_L(\boldsymbol{x})\big).$$

By subadditivity and submultiplicativity, we estimate

$$\begin{aligned} \|\Delta \boldsymbol{h}_{L+1}(\boldsymbol{x})\|_{\mathrm{R}} &= \Theta(\alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} \|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}) + \Theta(\alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} \|\boldsymbol{h}_L(\boldsymbol{x}) + \Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}) \\ &= \Theta(1) + \Theta\left(\alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}}\right), \end{aligned}$$

where we used $\alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}}$, $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the preliminary initial condition, and $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the update condition on the hidden weights. Therefore, requiring Principle 2.1 yields the update condition $\alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.

3.3.3 Final Initial Condition

We now derive the final initialization condition for the hidden weights (C1.2) by incorporating the constraints imposed by the update conditions. Multiplying the two first-order update conditions on hidden weights yields

$$\alpha_l^2 \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L^2), \ \forall l \in [L].$$

On the other hand, the second-order update condition is $\alpha_l \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L)$ for all $l \in [L]$. Dividing the product of the first-order conditions by the second-order condition immediately gives $\alpha_l \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L)$ for any $l \in [L]$, which completes the derivation of Condition 3.1.

Given the first-order update condition (C2.2), the refined initialization condition (C1.2) and the second-order update condition (C2.3) are algebraically equivalent up to constant factors. Thus, one of them could be derived from the other under (C2.2). We nevertheless keep both in Condition 3.1 because (C1.2) states the final initial forward-scale requirement, while (C2.3) makes explicit the second-order maximal-update constraint responsible for the $k = 1$ versus $k \geq 2$ distinction.

4 Implementation of Spectral Condition

In this section, we map Condition 3.1 to concrete HP parameterizations. The initialization parameters σ_l and α_l are optimizer-agnostic, while the learning-rate scaling of η_l depends on the optimizer. In the main text, we instantiate this recipe for Muon-Kimi (Liu et al., 2025). Table 2 in Appendix C extends the similar recipe to additional optimizers and optimizer-dependent HPs, including weight decay and stability term ε. The corresponding results for Condition B.1 can be found in Table 8 of Appendix D.

4.1 Initial Condition

Since these HPs interact to satisfy the spectral condition, multiple equivalent parameterization solutions exist (Yang & Hu, 2021; Yang & Littwin, 2023). To facilitate practical adoption, we choose to align the

Table 1: **μP implementation of Condition 3.1 ($k = 2$) for Muon-Kimi (Liu et al., 2025) under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	η_{base}	$\eta_{\text{base}}/\sqrt{r_n}$ (η_{base})	η_{base}

initial variance to the standard width-scaling μP implementation (Yang et al., 2022). Specifically, for any weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, we set:

$$\sigma_l = \begin{cases} \Theta\big(\frac{1}{\sqrt{n_{\text{in}}}} \min\{1, \sqrt{\frac{n_{\text{out}}}{n_{\text{in}}}}\}\big), & 0 \le l \le L, \\ \Theta(1), & l = L+1. \end{cases}$$

Under this variance parameterization, the RMS operator norms of weight matrices at initialization satisfy:

$$\|\boldsymbol{W}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\boldsymbol{W}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \cdot \Theta\left(\sigma_l(\sqrt{n_{\text{in}}} + \sqrt{n_{\text{out}}})\right) = \begin{cases} \Theta(1), & 0 \le l \le L, \\ \Theta(n_{\text{in}}), & l = L+1, \end{cases} \tag{6}$$

where we used the spectral norm property of random matrices reviewed in Section 2. Based on Equation (6), we are ready to determine the parameterization of α_l to satisfy initial conditions in Condition 3.1.

For the input and output layers, given

$$\alpha_l \|\boldsymbol{W}_l\|_{\text{R}} = \begin{cases} \Theta(\alpha_0), & l = 0, \\ \Theta(\alpha_{L+1} n_{\text{in}}), & l = L+1, \end{cases}$$

to satisfy (C1.1), we need to set

$$\alpha_0 = \Theta(1), \quad \alpha_{L+1} = \Theta(1/n_{\text{in}}). \tag{7}$$

For the hidden layers, given

$$\alpha_l \|\boldsymbol{W}_l^{(1)}\|_{\text{R}} \|\boldsymbol{W}_l^{(2)}\|_{\text{R}} = \Theta(\alpha_l), \ l \in [L],$$

to satisfy (C1.2), we need to set

$$\alpha_l = \Theta(1/L), \ l \in [L]. \tag{8}$$

4.2 Update Condition for Muon-Kimi

Since different optimizers take different scales of $\|\Delta \boldsymbol{W}_l\|_{\text{R}}$, the implementation of the update condition depends on the choice of optimizer. In the main text, we focus on Muon-Kimi (Liu et al., 2025). The derivations for Muon (Jordan et al., 2024b), Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2024), AdamW (Loshchilov & Hutter, 2019), Sophia (Liu et al., 2024b), Lion (Chen et al., 2023), SGD, and SSO (Xie et al., 2026) are deferred to Appendix C, where we recover several existing μP results (e.g., for SGD, AdamW, and some matrix-preconditioned optimizers) in the width–depth scaling setting (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025; Qiu et al., 2025).

Muon-Kimi (Liu et al., 2025) is a widely used variant of Muon (Jordan et al., 2024b) designed to align its update scales of matrix parameters with those of AdamW-optimized vector parameters by applying RMS normalization, which facilitates the reuse of HPs well-tuned for AdamW. It has been successfully applied

to pretraining models with up to 1T parameters (Team et al., 2025). Specifically, for a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the update rule (without weight decay) is:

$$\Delta \boldsymbol{W}_l = -\eta_l \cdot 0.2\sqrt{\max\{n_{\text{in}}, n_{\text{out}}\}} \cdot \boldsymbol{U}_l \boldsymbol{V}_l^\top,$$

where $\boldsymbol{U}_l, \boldsymbol{V}_l$ are the left and right singular vector matrices of the gradient that $\nabla_{\boldsymbol{W}_l}\mathcal{L} = \boldsymbol{U}_l \boldsymbol{\Sigma}_l \boldsymbol{V}_l^\top$. The resulting update norm satisfies

$$\|\Delta \boldsymbol{W}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\Delta \boldsymbol{W}_l\|_2 = \Theta\left(\eta_l \sqrt{n_{\text{in}}} \max\left\{1, \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\right\}\right) = \begin{cases} \Theta(\eta_l), & l = 0, \\ \Theta(\eta_l \sqrt{n_{\text{in}}}), & l \in [L], \\ \Theta(\eta_l n_{\text{in}}), & l = L+1. \end{cases} \tag{9}$$

Based on Equation (9), we are now ready to determine the parameterization of η_l to satisfy the update condition.

For the input and output layers, given the dimension magnitude assumption in Equation (3) and the α_l parameterization in Equation (7), we have:

$$\begin{aligned} \alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} &= \begin{cases} \Theta(1)\Theta(\eta_l), & l = 0, \\ \Theta(1/n_{\text{in}})\Theta(\eta_l n_{\text{in}}), & l = L+1, \end{cases} \\ &= \Theta(\eta_l). \end{aligned}$$

Thus, to satisfy (C2.1), we need to set:

$$\eta_0 = \Theta(1), \quad \eta_{L+1} = \Theta(1).$$

For the hidden layers, let us first consider the first-order update condition. Given the dimension magnitude in Equation (3), the weight norm at initialization in Equation (6), and the α_l parameterization in Equation (8), we have:

$$\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\text{R}} \|\boldsymbol{W}_l^{(1)}\|_{\text{R}} = \Theta(1/L) \cdot \|\Delta \boldsymbol{W}_l^{(2)}\|_{\text{R}} = \Theta\left(\frac{1}{L}\eta_l^{(2)}\sqrt{n_{\text{in}}}\right).$$

Thus, to satisfy (C2.2), we need to set:

$$\eta_l^{(2)} = \Theta(\frac{1}{\sqrt{n_{\text{in}}}}),\ l \in [L]. \tag{10}$$

Symmetrically, we have the same choice for $\boldsymbol{W}_l^{(1)}$ that $\eta_l^{(1)} = \Theta(\frac{1}{\sqrt{n_{\text{in}}}})$ to ensure the first-order condition.

By the algebraic equivalence discussed in Section 3.3.3, the second-order condition (C2.3) is then automatically satisfied by the initial condition (C1.2) and the first-order condition (C2.2), so no further constraint is needed for implementing the second-order condition (C2.3).

This completes the μP parameterization of Muon-Kimi, which is summarized in Table 1.

4.3 Simplified Implementation Rule for Modern Optimizers

The Muon-Kimi derivation above, together with derivations for other optimizers in Appendix C, reveals a useful implementation-level simplification. For the modern optimizers considered in this paper, except SGD, implementing Condition 3.1 amounts to taking the corresponding width-scaling μP implementation and adding the hidden residual multiplier $\alpha_l = \Theta(1/L)$. The reason is that normalized and preconditioned updates remove the depth factor α_l inherited by the raw hidden-layer gradients. As a result, their update norms do not depend on α_l, and the optimizer-specific HP rule remains the same as in width-scaling μP. SGD does not share this simplification because its update is proportional to the raw gradient. After setting $\alpha_l = \Theta(1/L)$, the spectral update condition therefore still requires an additional α_l-dependent learning rate rescaling.

Takeaway 2. For most modern optimizers, normalization or preconditioning removes the depth factor α_l of hidden gradients, so the μP formulation from Condition 3.1 is just width-scaling μP plus a hidden residual multiplier $\alpha_l = \Theta(1/L)$.

4.4 Practical HP Parameterization and Transfer

In practice, μP is often implemented using a ratio-based approach (Yang et al., 2022; Dey et al., 2025; Zheng et al., 2025). We define width and depth scaling ratios as $r_n = n/n_{\text{base}}$ and $r_L = L/L_{\text{base}}$, where n_{base} and L_{base} are some fixed base model constants. The target model's HPs are then set by scaling the corresponding base HPs, denoted as α_{base}, σ^2_{base}, and η_{base}, according to these ratios.

For instance, for Muon-Kimi, the hidden layer learning rate is set to $\eta_l = \eta_{\text{base}}/\sqrt{r_n}$, which satisfies the theoretical requirement $\eta_l = \eta_{\text{base}}/\sqrt{n/n_{\text{base}}} = \Theta(\eta_{\text{base}}/\sqrt{n})$ in Equation (10). Table 1 summarizes the complete HP parameterization for Muon-Kimi under width-depth scaling derived above.

As illustrated in Section 1, a critical utility of μP is enabling HP transfer, which effectively reduces the cost of HP search for training large models. In practice, the transfer follows this procedure: optimal base HPs (e.g., η_{base}) are first identified on a small model; these optimal values are then transferred to a larger target model to obtain true HPs (e.g., $\eta_{\text{base}}/\sqrt{r_n}$). Consequently, we only need to search the base HPs on the computationally inexpensive small model.

Note that although the μP parameterization is derived from a simplified setup, we apply it to standard language models pretraining and verify its practical utility in the next section.

5 Experiments

In this section, we empirically evaluate the μP formulations derived from our spectral conditions on GPT-2 style language models. We first show that Condition 3.1 ($k \geq 2$)[2] enables stable feature learning and robust HP transfer under width-depth scaling. We then compare it with Condition B.1 ($k = 1$) to validate the role of residual block depth k on depth scaling. The complete details and results are deferred to Appendix E.

5.1 Experimental Settings

Following standard empirical μP studies (Dey et al., 2025; Qiu et al., 2025), we train GPT-2 style Transformer language models (Radford et al., 2019; Karpathy, 2022) on the OpenWebText dataset (Gokaslan & Cohen, 2019), using the GPT-2 tokenizer with a maximum sequence length of 1024. All models fix the attention head dimension to 64 and use a feedforward (FFN) dimension of $4n$. We vary model width and depth around a base model $(n_{\text{base}}, L_{\text{base}}) = (256, 4)$, with widths up to 4096 and depths up to 256.

Since SGD and AdamW have been extensively studied in prior μP studies under width-depth scaling (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025), we focus on more recent optimizers: Muon-Kimi-AdamW, Muon-AdamW, Shampoo-AdamW, and Sophia. Following common practice (Liu et al., 2025), Muon-Kimi-AdamW uses Muon-Kimi for hidden matrix parameters and AdamW for all other parameters, such as embeddings, the LM head, and biases. Muon-AdamW and Shampoo-AdamW are defined analogously. In the main text, we present Muon-Kimi-AdamW as the primary example; additional optimizer results are reported in Appendix E.

We implement μP for both Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$); the corresponding HP parameterization overviews are given in Table 2 at Appendix C and Table 8 at Appendix D, respectively. For example, for the hybrid Muon-Kimi-AdamW optimizer under Condition 3.1, the Muon-Kimi part follows Table 3, while the AdamW part follows Table 6. Muon-AdamW and Shampoo-AdamW are implemented analogously, with details deferred to Appendix E. The main text focuses on learning rate transfer without weight decay, and weight decay transfer results are deferred to Figure 4 in Appendix E.3.

[2]In fact, Condition 3.1 is derived when $k = 2$. Since Condition B.2 ($k \geq 2$) leads to the same μP formulation, we use $k \geq 2$ to refer to this formulation throughout the experiments.

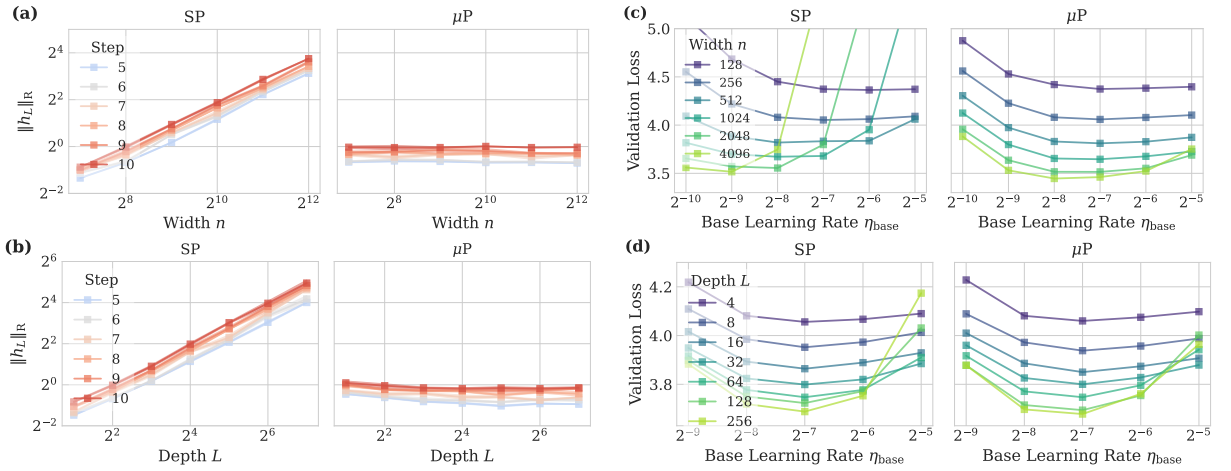

Figure 1: **Feature learning and HP transfer of Muon-Kimi-AdamW under SP and μP.** We train GPT-2 style models with Muon-Kimi-AdamW using SP and μP derived from Condition 3.1 (see Tables 1 and 6). μP maintains stable feature norms and enables robust HP transfer across both width and depth scaling, while generally achieving lower loss than SP as the model size increases. The detailed numerical values are provided in Appendix E.3.2.

5.2 Feature Learning and HP Transfer

In this section, we compare the feature learning stability and HP transferability of SP and the μP formulation derived from Condition 3.1 ($k \geq 2$). The main results are shown in Figure 1, with complete numerical results deferred to Appendix E.3.2.

Feature learning. We first examine feature-scale stability using standard coordinate-check tests (Yang et al., 2022; Dey et al., 2025; Ngom et al., 2025). Models are trained for 10 steps while scaling either width n or depth L, and we measure the RMS norm at the output of the final Transformer block $\|\boldsymbol{h}_L\|_{\mathrm{R}}$. As shown in Figure 1(a,b), under SP, the feature scale grows rapidly with both width and depth. In contrast, μP maintains stable and scale-invariant feature scales, consistent with Principle 2.1. This supports that the μP formulation derived from Condition 3.1 preserves stable feature learning under width-depth scaling.

HP transfer. We next evaluate HP transferability by training all models for 300M tokens with a batch size of 240, using a learning rate schedule with linear warmup followed by cosine decay. As shown in Figure 1(c), SP exhibits substantial shifts in the optimal learning rate when width is scaled, whereas μP keeps the optimal base learning rate nearly invariant. Under depth scaling, μP also preserves HP transferability and achieves lower loss than SP as depth increases (Figure 1(d)). These results support that Condition 3.1 provides a practical HP-transfer rule, which can significantly reduce tuning cost when scaling model size, particularly for pretraining large models (Singh et al., 2025; Team et al., 2025).

Figures 5, 6, and 8 in Appendix E report analogous feature learning and HP transfer results for Muon-AdamW, Shampoo-AdamW, and Sophia, respectively. They show the similar advantage of the μP formulation from Condition 3.1 over SP.

5.3 Role of Residual Block Depth

We further examine the central prediction of our theory: residual-branch depth k determines the appropriate depth μP formulation. Condition B.1 gives the Depth-μP-style scaling for $k = 1$, whereas Condition 3.1 gives the stricter CompleteP-style scaling for $k \geq 2$. Since the GPT-2 style Transformer contains multiple transformations in each residual branch, our theory predicts that Condition 3.1 should provide a better parameterization.

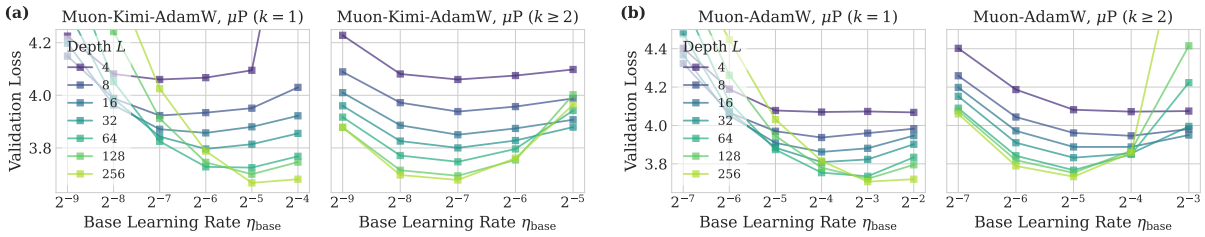

Figure 2: **Validating the role of residual block depth k.** We compare two μP implementations for GPT-2-style models trained with Muon-Kimi-AdamW and Muon-AdamW: Depth-μP-style formulation from Condition B.1 ($k = 1$) and CompleteP-style formulation from Condition 3.1 ($k \geq 2$). Condition 3.1 yields more stable HP transfer, empirically supporting it as the appropriate μP condition for architectures with multi-transformation residual branches. The numerical values are provided in Appendix E.3.

To test this prediction, we compare the two μP implementations under depth scaling. As shown in Figure 2, Condition 3.1 yields stable learning-rate transfer for both Muon-Kimi-AdamW and Muon-AdamW. In contrast, Condition B.1 shifts the optimal learning rate as depth increases, indicating a failure of HP transfer in practical multi-transformation architectures. These results support the claim in Section 3: the high-order update term appearing at $k \geq 2$ imposes the stricter scaling needed for Transformer-like residual blocks.

The same comparison for Shampoo-AdamW and Sophia in Figures 7 and 9 at Appendix E.3 shows similar trends, further supporting the effectiveness of Condition 3.1 ($k \geq 2$) beyond Muon-style optimizers.

5.4 Additional Diagnostics

For Muon-Kimi-AdamW, SP can appear to transfer the optimal learning rate reasonably well across depths in Figure 1(d). We attribute this to two factors. First, the tested depths are still moderate; as depth increases further, Figure 1(b) suggests that hidden features eventually diverge, making stable depth scaling under SP infeasible. Second, modern architectural components such as LayerNorm (Ba et al., 2016) and QKNorm (Henry et al., 2020) substantially enhance training stability, partially masking the underlying scaling pathology of SP at practical depths. To isolate this effect, we remove LayerNorm layers and repeat the depth-scaling experiments in Appendix E.3.2. The results in Figure 3 show that SP training becomes unstable and depth-wise HP transfer breaks down, while μP remains stable even at large depths ($L = 256$) and continues to exhibit robust HP transfer.

We emphasize that this apparent SP depth transfer is not universal. Figure 6 and 8 in Appendix E show that for Shampoo-AdamW and Sophia, the μP parameterization gives substantially more reliable depth-wise HP transfer than SP. Thus, the Muon-Kimi-AdamW behavior in Figure 1(d) appears to be a partially masked case, likely aided by normalization and the tested depth range.

Takeaway 3. CompleteP-style scaling from Condition 3.1 ($k \geq 2$) achieves stable feature learning and robust HP transfer under width-depth scaling, while SP and Depth-μP-style scaling from Condition B.1 ($k = 1$) often fail to do so.

6 Conclusion

In this paper, we present a simple and unified spectral framework for μP under joint width-depth scaling. The framework gives an operational condition on the norm of weights and their per-step updates, and explains why residual blocks with one transformation and those with multiple transformations lead to different depth-scaling rules. By mapping this condition to concrete HP choices, we obtain a general recipe for implementing μP across a broad class of optimizers. Experiments on GPT-2 style language models show that the resulting $k \geq 2$ formulation preserves scale-invariant feature learning and supports robust HP transfer, while the $k = 1$

formulation and SP often fail to do so. These results suggest that the proposed spectral perspective provides a practical and interpretable route to width-depth μP for modern architectures.

Broader Impact Statement

This is mainly a theoretical work, and the proposed μP formulations have the potential to accelerate progress in scaling generative foundation models, including language modeling, text-to-image and video generation. However, improvements in scaling foundation models could also facilitate the creation of deepfakes for disinformation.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Laura Balzano, Tianjiao Ding, Benjamin D. Haeffele, Soo Min Kwon, Qing Qu, Peng Wang, Zhangyang Wang, and Can Yaras. An overview of low-rank structures in the training and adaptation of large models. *CoRR*, abs/2503.19859, 2025.

Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Yuri Prince, Björn Deiseroth, Andrés Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μp: The unit-scaled maximal update parametrization. In *ICLR*, 2025.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *NeurIPS*, 2022.

Blake Bordelon, Hamza Tahir Chaudhry, and Cengiz Pehlevan. Infinite limits of multi-head transformer dynamics. In *NeurIPS*, 2024a.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *ICLR*, 2024b.

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

Nolan Dey, Gurpreet Gosal, Zhiming Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *CoRR*, abs/2304.03208, 2023.

Nolan Dey, Shane Bergsma, and Joel Hestness. Sparse maximal update parameterization: A holistic approach to sparse training dynamics. In *NeurIPS*, 2024.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Bill Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *CoRR*, abs/2505.01618, 2025.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Moritz Haas, Jin Xu, Volkan Cevher, and Leena Chennuru Vankadara. μp^2: Effective sharpness aware minimization requires layerwise perturbation scaling. In *NeurIPS*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, volume EMNLP 2020, pp. 4246–4253, 2020.

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. *CoRR*, abs/2203.15556, 2022.

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.

Satoki Ishikawa and Ryo Karakida. On the parameterization of second-order optimization effective towards the infinite width. In *ICLR*, 2024.

Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, pp. 8580–8589, 2018.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL `https://github.com/KellerJordan/modded-nanogpt`.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. *URL https://kellerjordan. github. io/posts/muon*, 6, 2024b.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

Andrej Karpathy. nanogpt. `https://github.com/karpathy/nanoGPT`, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *ICLR*, 2024b.

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate o(1/k2). In *Dokl akad nauk Sssr*, volume 269, pp. 543, 1983.

Marieme Ngom, Sam Foreman, Venkatram Vishwanath, et al. Extending μp: Spectral conditions for feature learning across optimizers. In *OPT 2025: Optimization for Machine Learning*, 2025.

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. *arXiv preprint arXiv:2512.05620*, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *ICLR*, 2017.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher. On feature learning in structured state space models. In *NeurIPS*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, et al. Controlled llm training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

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.

Greg Yang. Tensor programs III: neural matrix laws. *CoRR*, abs/2009.10685, 2020.

Greg Yang and Edward J. Hu. Tensor programs IV: feature learning in infinite-width neural networks. In *ICML*, volume 139, pp. 11727–11737. PMLR, 2021.

Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit. *CoRR*, abs/2308.01814, 2023.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: tuning large neural networks via zero-shot hyperparameter transfer. *CoRR*, abs/2203.03466, 2022.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *CoRR*, abs/2310.17813, 2023.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: feature learning in infinite depth neural networks. In *ICLR*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *ICML*, 2024.

Chenyu Zheng, Xinyu Zhang, Rongzhen Wang, Wei Huang, Zhi Tian, Weilin Huang, Jun Zhu, and Chongxuan Li. Scaling diffusion transformers efficiently via μp. *CoRR*, abs/2505.15270, 2025.

Contents of Appendix

A Additional Related Work

A.1 μP under Width Scaling

μP was originally introduced to characterize and control training dynamics in the infinite-width limit of neural networks, to enable stable feature learning through appropriate HPs adjustment (Yang & Hu, 2021). Early theoretical work formalized μP for MLP trained with SGD using Tensor Programs (Yang, 2020; Yang & Hu, 2021) and dynamical mean-field theory (Bordelon & Pehlevan, 2022). Empirically, Yang et al. (2022) showed that μP stabilizes optimal HPs across model widths, thereby substantially reducing the tuning cost when scaling up model size.

Motivated by these advantages, the μP principle has been successfully extended to a wide range of modern architectures, including convolutional neural networks (Yang & Littwin, 2023), Transformers (Yang & Littwin, 2023), diffusion Transformers (Zheng et al., 2025), and state-space models (Vankadara et al., 2024). In parallel, μP has been developed for a broad class of optimization algorithms, such as AdamW (Loshchilov & Hutter, 2019), Muon (Ngom et al., 2025), sharpness-aware optimizer (Haas et al., 2024), second-order optimizers (Ishikawa & Karakida, 2024), low-precision training (Blake et al., 2025), and sparse training (Dey et al., 2024). These μP-based methods have also been successfully applied to the pretraining of large-scale foundation models in industrial settings (Yang et al., 2022; Dey et al., 2023; Hu et al., 2024; Zheng et al., 2025).

Despite substantial progress, μP formulations are often tightly coupled to specific architectures (Yang & Littwin, 2023; Zheng et al., 2025; Vankadara et al., 2024) or particular optimization algorithms (Yang & Littwin, 2023; Haas et al., 2024; Ishikawa & Karakida, 2024; Ngom et al., 2025), and their derivations typically rely on technically involved tools such as Tensor Programs or dynamical mean-field theory (Yang, 2020; Yang & Hu, 2021; Yang & Littwin, 2023; Bordelon & Pehlevan, 2022). As a result, it remains difficult to systematically analyze new architectures or optimizers and derive the corresponding μP formulations. To alleviate this limitation, Yang et al. (2023) proposed a simple and general spectral condition for realizing μP in the width-scaling regime, enabling transparent derivations for a broad class of optimization algorithms (Yang et al., 2023; Ngom et al., 2025; Haas et al., 2024). However, this spectral perspective focuses solely on width scaling and does not account for depth scaling, which is crucial for modern deep architectures.

A.2 μP under Width-Depth Scaling

Recent work has begun to extend the μP principle beyond pure width scaling to regimes where network depth grows jointly with model size. Early theoretical analyses (Yang et al., 2024; Bordelon et al., 2024b) of residual networks with one-layer residual blocks trained by SGD or Adam showed that a hidden residual multiplier α_l of order $\Theta(1/\sqrt{L})$ suffices to preserve stable feature learning, but observed that this scaling fails to maintain HP transferability in practical architectures such as Transformers (Yang et al., 2024; Dey et al., 2025).

Subsequent studies (Bordelon et al., 2024a) of Transformers with two-layer residual blocks trained by SGD using dynamical mean-field theory argued that a stronger hidden residual scaling of $\Theta(1/L)$ is preferable, as it enables both nontrivial feature learning and non-negligible updates in attention layers. More recently, Dey et al. (2025) shows that for residual networks with two-layer blocks trained by AdamW, the residual multiplier α_l of $\Theta(1/L)$ is in fact necessary to simultaneously maintain stable feature learning and maximize parameter updates. Moreover, Dey et al. (2025) empirically find that this parameterization enables HP transfer in GPT-2-style Transformer. This hidden residual multiplier (Bordelon et al., 2024a; Dey et al., 2025) is further applied to some matrix-preconditioned optimizers (Qiu et al., 2025).

Overall, existing μP extensions to the joint width-depth scaling regime remain fragmented, architecture- and optimizer-specific, and often rely on technically involved analyses, motivating the need for a simple and unified framework.

B Spectral Condition for General Residual Networks

In this section, we provide derivations that complement the main-text analysis of the two-layer residual block ($k = 2$) in Section 3. The goal is to clarify how the spectral condition changes with the internal depth k of the residual branch and to justify why the two-layer case is the minimal representative of fixed-depth branches with $k \geq 2$. We first analyze one-layer residual blocks ($k = 1$), where the absence of second-order update terms leads to the looser Depth-μP-style scaling in Condition B.1. We then extend the analysis to residual blocks with an arbitrary fixed number $k \geq 2$ of transformations, showing that the additional higher-order update terms do not change the resulting μP implementation beyond the two-layer case. Finally, we discuss bias parameters and show that they can be incorporated as a lightweight extension without changing the main HP parameterization for the matrix weights.

As in the main text, we assume $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$ for simplicity, which holds for natural image data and one-hot language data ($\Theta(1/\sqrt{d_0}) = \Theta(1)$). We also assume the network dimensions satisfy Equation (3). Furthermore, we also use norm estimates based on subadditivity and submultiplicativity to track the typical scales of $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ and $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$. The tightness justification under width-depth scaling is in Appendix F, following the width-scaling treatment of Yang et al. (2023).

B.1 One-layer Residual Block

B.1.1 Problem Setup

We consider a residual network with one-layer residual blocks ($k = 1$), defined as

$$\begin{aligned} \boldsymbol{h}_0(\boldsymbol{x}) &= \alpha_0 \boldsymbol{W}_0 \boldsymbol{x}, \\ \boldsymbol{h}_l(\boldsymbol{x}) &= \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}), \quad \forall\, l \in [L], \\ \boldsymbol{h}_{L+1}(\boldsymbol{x}) &= \alpha_{L+1} \boldsymbol{W}_{L+1} \boldsymbol{h}_L(\boldsymbol{x}), \end{aligned}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{n \times d_0}$, $\boldsymbol{W}_l \in \mathbb{R}^{n \times n}$ for $l \in [L]$, and $\boldsymbol{W}_{L+1} \in \mathbb{R}^{d_{L+1} \times n}$. The network output $\boldsymbol{h}_{L+1}(\boldsymbol{x}) \in \mathbb{R}^{d_{L+1}}$ is used to compute the loss $\mathcal{L}(\boldsymbol{h}_{L+1}(\boldsymbol{x}), \boldsymbol{y})$. This case serves as a reference point for understanding the transition from $k = 1$ to $k \geq 2$: because each residual branch contains only one transformation, its update expansion has no second-order direct update term.

B.1.2 Spectral Scaling Condition

We now state the spectral scaling condition for the above residual network with one-layer blocks for realizing the μP Principle 2.1 under joint width–depth scaling.

Condition B.1 (Spectral condition for μP under joint width-depth scaling, one-layer residual block)**.** *To realize μP Principle 2.1, the initial weights and their per-step updates should satisfy:*

- ***Initial condition.***
  - *Input and output weights:* $\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}},\ \alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
  - *Hidden weights:* $\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L})$, $\forall l \in [L]$.
- ***Update condition.***
  - *Input and output weights:* $\alpha_0 \|\Delta\boldsymbol{W}_0\|_{\mathrm{R}},\ \alpha_{L+1} \|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
  - *Hidden weights (first-order):* $\alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L),\ \forall l \in [L]$.

The essential distinction between one-layer and two-layer residual blocks lies in the *order of the weight-update terms* that directly affect feature evolution. For a one-layer residual block, the feature update expansion contains only zero-order ($\boldsymbol{\epsilon}_0(L)$) and first-order ($\boldsymbol{\epsilon}_1(L)$) terms in the weight updates (see details in Appendix B.1.4). As a result, only the first-order direct update term needs to be maximized under Principle P2, leading to the condition $\alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L)$, while leaving the initialization scale unconstrained beyond the preliminary condition $\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L})$.

From an algorithmic (HP parameterization) perspective, when the initialization variance σ_l^2 is aligned with the standard width-scaling μP framework (Yang et al., 2022) as in Section 4, the condition $\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L})$ naturally induces an $\mathcal{O}(1/\sqrt{L})$ residual multiplier. Moreover, Depth-μP-style formulation (Bordelon et al., 2024b; Yang et al., 2024) adopts the $\Theta(1/\sqrt{L})$ residual multiplier, which they interpret as further promoting *feature diversity*. Within our spectral framework, this choice is unified as a natural case corresponding to further maximizing the magnitude of the zero-order feature update $\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}}$ (see derivations in Appendix B.1.4). The parameterization of other HPs (e.g., learning rate) in Depth-μP-style formulation (Bordelon et al., 2024b; Yang et al., 2024) can also be recovered from Condition B.1, with the details deferred to Appendix D.

In contrast, two-layer residual blocks introduce *second-order* update terms arising from products of weight updates across the two sublayers. To satisfy the μP principle (P2), these second-order contributions should be maximized to $\Theta(1)$ as in (C2.3). This requirement imposes an additional constraint on the scaling of weight updates, which in turn tightens the initialization condition to $\alpha_l \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L)$ in (C1.2) and thus the residual multiplier to $\alpha_l = \Theta(1/L)$. This important difference explains why the Depth-μP-style scaling ($k = 1$) does not directly capture residual branches with two or more transformations, and helps account for its poor depth-wise HP transfer behavior in Transformer experiments in Section 5 and Yang et al. (2024); Dey et al. (2025).

B.1.3 Derivation for Initial Condition

We first derive the initialization condition that ensures stability of feature magnitudes during forward propagation for single-layer residual blocks. We consider each layer sequentially.

Input layer. The argument is identical to the two-layer case. By the submultiplicativity of the RMS operator norm, we have

$$\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \alpha_0 \|\boldsymbol{W}_0 \boldsymbol{x}\|_{\mathrm{R}} = \Theta(\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}} \|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}}),$$

where we assume $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$. Thus, choosing $\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ ensures $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. For a single-layer residual block, the forward recursion is

$$\boldsymbol{h}_l(\boldsymbol{x}) = \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}).$$

Expanding the recursion yields

$$\boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}). \tag{11}$$

Applying subadditivity, we can estimate their order as

$$\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}) \|_{\mathrm{R}} \right).$$

Since we have $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, it suffices to ensure that $\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$ to preserve $\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. Under i.i.d. zero-mean Gaussian initialization, the summands are approximately independent zero-mean random vectors (Yang & Hu, 2021; Yang et al., 2024; Dey et al., 2025), so the typical squared RMS norm of their sum scales as the sum of the squared RMS norms (see Theorem 3.3.1 in Vershynin (2018)), yielding that

$$\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}) \|_{\mathrm{R}} = \Theta\left(\sqrt{ \sum_{l=1}^{s} \|\alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2 } \right).$$

By submultiplicativity, we can further estimate $\|\alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}})$. Therefore, starting from $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, imposing

$$\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}), \quad l \in [L]$$

recursively ensures $\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}) \|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$. This provides the initial condition on the hidden weights.

Output layer. The same argument as for the two-layer block case gives

$$\|\boldsymbol{h}_{L+1}(\boldsymbol{x})\|_{\mathrm{R}} = \|\alpha_{L+1}\boldsymbol{W}_{L+1}\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}}\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}) = \Theta(\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}}),$$

so choosing $\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$ keeps the output scale stable. This completes the initialization analysis.

B.1.4 Derivation for Update Condition

We next derive the update condition required to ensure stable feature evolution, i.e., $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, while maximally updating parameters as prescribed by μP Principle (P2).

Input layer. Since $\Delta\boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0\Delta\boldsymbol{W}_0\boldsymbol{x}$, submultiplicativity yields

$$\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}}\|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}}),$$

and thus we set $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. Expanding Equation (11) after a single gradient step gives

$$\Delta\boldsymbol{h}_s(\boldsymbol{x}) = \Delta\boldsymbol{h}_0(\boldsymbol{x}) + \underbrace{\sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l \Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})}_{\boldsymbol{\epsilon}_0(s)} + \underbrace{\sum_{l=1}^{s} \alpha_l \Delta\boldsymbol{W}_l (\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1(s)}.$$

Unlike the two-layer case, there is no second-order update term, since each residual block contains only a single weight matrix. By the subadditivity of vector norms, we have

$$\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1(s)\|_{\mathrm{R}}).$$

Since $\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the input-layer update, we have $\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Omega(1)$ for all $s \in [L]$. Moreover, by subadditivity, the remaining terms do not decay with depth, implying $\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}})$ for any $s \in [L]$. Therefore, to enforce Principle 2.1, it suffices to require $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while satisfying Principle (P2).

Zero-order term. The term $\boldsymbol{\epsilon}_0(L)$ propagates feature updates from earlier layers and does not depend on the weight update $\Delta\boldsymbol{W}_l$ at the current layer, so it does not need to be maximized from Principle (P2). Therefore, it suffices to verify that $\boldsymbol{\epsilon}_0(L)$ remains $\mathcal{O}(1)$ under the initial condition. In fact, the same argument used for deriving $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ directly implies

$$\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} = \Theta\left(\sqrt{\sum_{l=1}^{L} \alpha_l^2 \|\boldsymbol{W}_l\|_{\mathrm{R}}^2 \|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2}\right) = \mathcal{O}(1),$$

where we use the self-consistent fact that $\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally set $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. *We note that if we further set $\alpha_l\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/\sqrt{L})$ for all $l \in [L]$ in the initial condition, then $\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} = \Theta(1)$ and is maximized, which leads to the Depth-μP formulations* (Yang et al., 2024; Bordelon et al., 2024b).

First-order terms. The first-order update terms reflect the direct effect of weight updates $\Delta\boldsymbol{W}_l$ on features and must be maximized ($\Theta(1)$) to satisfy the μP Principle (P2). Using subadditivity and submultiplicativity, we estimate the order of $\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}}$ as

$$\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}} = \Theta\left(\sum_{l=1}^{L} \alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right) + \Theta\left(\sum_{l=1}^{L} \alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} \|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right).$$

For $l \in [L]$, using $\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the preliminary initial condition and the self-consistent fact that $\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ if we finally set $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, we can obtain $\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}} = \Theta\big(\sum_{l=1}^{L} \alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}}\big)$. To satisfy Principle (P2), we need to maximize the contribution from each $\Delta\boldsymbol{W}_l$ and ensure $\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}} = \Theta(1)$ at the same time, which naturally requires

$$\alpha_l \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L), \qquad \forall\, l \in [L],$$

which completes the first-order update condition on hidden weights.

Output layer. The output layer has the same form as in the two-layer block case ($k = 2$), since it depends only on $\boldsymbol{h}_L(\boldsymbol{x})$ and not on the internal structure of the residual blocks. Given $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ and $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ from the hidden-layer analysis, the same argument as in Section 3.3 yields

$$\alpha_{L+1} \|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1).$$

B.2 Multi-layer Residual Block

This subsection justifies the main-text choice of analyzing the two-layer block ($k = 2$) as the representative case for all fixed-depth residual branches with $k \geq 2$.

B.2.1 Problem Setup

We now extend the spectral analysis from one- and two-layer residual blocks to the general case of k-layer residual blocks, where $k \geq 2$ is a fixed $\Theta(1)$ constant. The fixed-k assumption is important: we do not address regimes where the internal block depth itself scales with width or network depth. Specifically, we consider a residual network of depth L whose forward propagation is given by

$$\begin{aligned}
\boldsymbol{h}_0(\boldsymbol{x}) &= \alpha_0 \boldsymbol{W}_0 \boldsymbol{x}, \\
\boldsymbol{h}_l(\boldsymbol{x}) &= \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \boldsymbol{W}_l^{(k)} \boldsymbol{W}_l^{(k-1)} \cdots \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) = \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x}), \quad \forall\, l \in [L], \\
\boldsymbol{h}_{L+1}(\boldsymbol{x}) &= \alpha_{L+1} \boldsymbol{W}_{L+1} \boldsymbol{h}_L(\boldsymbol{x}).
\end{aligned}$$

Here, each residual block consists of a depth-k linear transformation, with $\{\boldsymbol{W}_l^{(i)}\}_{i=1}^{k}$ denoting the weight matrices within the l-th block. As in the previous sections, $\boldsymbol{h}_{L+1}(\boldsymbol{x})$ denotes the network output used to compute the loss. As in the two-layer block case, our goal is to derive a spectral condition for realizing μP Principle 2.1 in this setting.

In the following, we show that although increasing the internal block depth k introduces higher-order interactions between weight updates, the resulting spectral conditions admit a simple and systematic form, and do not fundamentally alter the algorithmic implementation of μP.

B.2.2 Spectral Scaling Condition

We now state the spectral scaling condition for the above residual network with k-layer residual blocks that is sufficient for the μP principle under joint width–depth scaling.

Condition B.2 (Spectral condition for μP under joint width-depth scaling, k-layer residual block)**.** *To realize μP Principle 2.1, the initial weights and their per-step updates should satisfy:*

- ***Initial condition.***
  - *Input and output weights:* $\alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}},\ \alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
  - *Hidden weights:* $\alpha_l \prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L)$, $\forall l \in [L]$.
- ***Update condition.***

- *Input and output weights:* $\alpha_0\|\Delta \boldsymbol{W}_0\|_{\mathrm{R}},\ \alpha_{L+1}\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
- *Hidden weights (first-order):* $\alpha_l\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} = \Theta(1/L),\ \forall l \in [L], i \in [k]$.
- *Hidden weights (j-order, $j \geq 2$), automatically satisfied by combining the initial condition and the first-order update condition:* $\alpha_l \prod_{i\in S}\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{i\notin S}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L),\ \forall S \subseteq [k],\ |S| = j,\ j \in [k],\ l \in [L]$.

Condition B.2 reveals that extending residual blocks from two layers to a general fixed depth $k \geq 2$ does not change the algorithmic realization of the μP principle. Compared to the two-layer case, the new elements introduced by a deeper block are higher-order interaction terms among weight updates within the same block. However, we show these higher-order terms do not impose additional constraints beyond those already enforced by the initial condition and the first-order update condition.

Concretely, once the product of spectral norms at initialization satisfies $\alpha_l \prod_{i=1}^{k}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L)$ and each update obeys the first-order scaling $\alpha_l\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} = \Theta(1/L)$, all higher-order update contributions of order $j \geq 2$ are automatically controlled as $\Theta(1/L)$. As a result, increasing the internal block depth k only increases the number of such higher-order contributions, but does not alter their scaling behavior.

Following the same steps as derivations for implementations in Section 4 and Appendix C, we can find that *implementing μP for a k-layer residual block requires no additional parameterization beyond those already needed for the two-layer case*. In particular, when the initialization variance is aligned with the standard width-scaling μP formulation (Yang & Hu, 2021; Yang et al., 2022) as in Section 4 ($\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1),\ \forall l \in [L]$), the initial condition still induces the residual multiplier $\alpha_l = \Theta(1/L)$ for $l \in [L]$, which is the same as the two-layer case. Built upon the initial condition, the first-order update condition is reduced to

$$\alpha_l\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} = \Theta\left(\frac{1}{L}\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\right).$$

Therefore, requiring the first-order update condition yields $\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ for $\forall l \in [L], i \in [k]$. This is also in the same way as the two-layer case ($\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ for $\forall l \in [L], i \in [2]$), thus leading to the same optimizer-related HPs adjustment. The multi-layer analysis, therefore, serves to justify the robustness and generality of the two-layer μP prescription, rather than to introduce a distinct algorithm dependent on block depth.

B.2.3 Derivation for Preliminary Initial Condition

We first derive a preliminary initialization condition that guarantees stability of feature magnitudes during forward propagation for k-layer ($k \geq 2$) residual blocks. As in the two-layer case, we analyze each layer sequentially.

Input layer. By the submultiplicativity of the RMS operator norm, we have

$$\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \alpha_0\|\boldsymbol{W}_0\boldsymbol{x}\|_{\mathrm{R}} = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}\,\|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}),$$

where we have assumed $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$. Thus, choosing $\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ ensures $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. Expanding the residual recursion yields

$$\boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_{s-1}(\boldsymbol{x}) + \alpha_s \prod_{i=1}^{k}\boldsymbol{W}_s^{(i)}\boldsymbol{h}_{s-1}(\boldsymbol{x}) = \cdots = \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s}\alpha_l \prod_{i=1}^{k}\boldsymbol{W}_l^{(i)}\boldsymbol{h}_{l-1}(\boldsymbol{x}). \tag{12}$$

Applying subadditivity, we can estimate their order as

$$\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \Big\|\sum_{l=1}^{s}\alpha_l \prod_{i=1}^{k}\boldsymbol{W}_l^{(i)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\Big\|_{\mathrm{R}}\right).$$

Since we have $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, it suffices to ensure that $\|\sum_{l=1}^{s} \alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$ to preserve $\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. Under i.i.d. zero-mean Gaussian initialization, the summands are approximately independent zero-mean random vectors (Yang & Hu, 2021; Yang et al., 2024; Dey et al., 2025), so the typical squared RMS norm of their sum scales as the sum of the squared RMS norms (see Theorem 3.3.1 in Vershynin (2018)), yielding that

$$\left\| \sum_{l=1}^{s} \alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) \right\|_{\mathrm{R}} = \Theta\left(\sqrt{\sum_{l=1}^{s} \|\alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2} \right).$$

By submultiplicativity, we can further estimate $\|\alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_l \prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}})$. Therefore, starting from $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, imposing

$$\alpha_l \prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}), \quad l \in [L]$$

recursively ensures $\|\sum_{l=1}^{s} \alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(1)$ for any $s \in [L]$. This provides a preliminary initial condition on the hidden weights, which will be further refined once update constraints are incorporated.

Output layer. The output layer has the same form as in the two-layer case ($k = 2$), yielding $\alpha_{L+1} \|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$, which keeps the output stable. This completes the preliminary initialization analysis.

B.2.4 Derivation for Update Condition

We next derive the update conditions required to ensure stable feature evolution by Principle (P1), i.e., $\|\Delta \boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, while maximally updating parameters as prescribed by Principle (P2).

Input layer. Since $\Delta \boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0 \Delta \boldsymbol{W}_0 \boldsymbol{x}$, the submultiplicativity yields

$$\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}} \|\boldsymbol{x}\|_{\mathrm{R}}) = \Theta(\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}}),$$

and thus we set $\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. Expanding the residual recursion in Equation (12) after one update step gives

$$\Delta \boldsymbol{h}_s(\boldsymbol{x}) = \Delta \boldsymbol{h}_0(\boldsymbol{x}) + \underbrace{\sum_{l=1}^{s} \alpha_l \prod_{i=1}^{k} \boldsymbol{W}_l^{(i)} \Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})}_{\boldsymbol{\epsilon}_0(s)} + \sum_{j=1}^{k} \boldsymbol{\epsilon}_j(s),$$

where $\boldsymbol{\epsilon}_j(s)$ collects all terms that are *j-th order* in $\{\Delta \boldsymbol{W}_l^{(i)}\}_{i=1}^{k}$. By the subadditivity of vector norms, we have

$$\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(s)\|_{\mathrm{R}} + \sum_{j=1}^{k} \|\boldsymbol{\epsilon}_j(s)\|_{\mathrm{R}} \right).$$

Since $\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the input-layer update, we have $\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Omega(1)$ for all $s \in [L]$. Moreover, by subadditivity, the remaining terms do not decay with depth, implying $\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}})$ for any $s \in [L]$. Therefore, to enforce Principle 2.1, it suffices to require $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while satisfying Principle (P2).

Zero-order term. The term $\boldsymbol{\epsilon}_0(L)$ propagates feature updates from earlier layers and does not depend on the weight update $\Delta \boldsymbol{W}_l$ at the current layer, so it does not need to be maximized from Principle (P2).

Therefore, it suffices to verify that $\boldsymbol{\epsilon}_0(L)$ remains $\mathcal{O}(1)$ under the preliminary initial condition. In fact, the same argument used for deriving $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ directly implies

$$\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} = \Theta\left(\sqrt{\sum_{l=1}^{L}\alpha_l^2\prod_{i=1}^{k}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}^2\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2}\right) = \mathcal{O}(1),$$

where we use the self-consistent fact that $\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally enforce $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

First-order terms. The first-order contributions take the form

$$\boldsymbol{\epsilon}_1(L) = \sum_{l=1}^{L}\alpha_l\sum_{i=1}^{k}\left(\boldsymbol{W}_l^{(k)}\cdots\Delta\boldsymbol{W}_l^{(i)}\cdots\boldsymbol{W}_l^{(1)}\right)(\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})).$$

Using subadditivity and submultiplicativity,

$$\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}} = \Theta\left(\sum_{l=1}^{L}\alpha_l\sum_{i=1}^{k}\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}}\right) = \sum_{i=1}^{k}\Theta\left(\sum_{l=1}^{L}\alpha_l\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}}\right),$$

where we used $\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ by the preliminary initial condition and the self-consistent fact that $\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally enforce $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. To satisfy Principle (P2), we need to maximize the contribution from each $\Delta\boldsymbol{W}_l$ and ensure $\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}} = \Theta(1)$ at the same time, which naturally requires

$$\alpha_l\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\prod_{m\neq i}\|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall\, l \in [L],\ i \in [k].$$

Any j-order terms. Similar to the first-order term, for $j \in [k]$, the j-th order feature update term $\boldsymbol{\epsilon}_j(L)$ admits the explicit form

$$\boldsymbol{\epsilon}_j(L) = \sum_{l=1}^{L}\alpha_l\sum_{\substack{S\subseteq[k]\\|S|=j}}\left(\prod_{i\in S}\Delta\boldsymbol{W}_l^{(i)}\right)\left(\prod_{i\notin S}\boldsymbol{W}_l^{(i)}\right)(\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})),$$

where S indexes the subset of sublayers whose weights are replaced by their per-step updates, and the products are ordered consistently with the forward computation within each residual block. Therefore, by the same subadditivity and submultiplicativity arguments for $\|\boldsymbol{\epsilon}_1(L)\|_{\mathrm{R}}$, the j-th order update terms satisfy

$$\|\boldsymbol{\epsilon}_j(L)\|_{\mathrm{R}} = \sum_{\substack{S\subseteq[k]\\|S|=j}}\Theta\left(\sum_{l=1}^{L}\alpha_l\prod_{i\in S}\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\prod_{i\notin S}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\right).$$

Principle (P2) requires maximizing each summand while ensuring $\|\boldsymbol{\epsilon}_j(L)\|_{\mathrm{R}} = \Theta(1)$. It is therefore sufficient to impose

$$\alpha_l\prod_{i\in S}\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}\prod_{i\notin S}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall\, S\subseteq[k],\ |S| = j,\ j \in [k],\ l \in [L].$$

Output layer. The same argument as in the two-layer case in Section 3 yields $\alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.

B.2.5 Derivation for Final Initial Condition

Multiplying the first-order update conditions for each hidden weight yields

$$\alpha_l^k\prod_{i=1}^{k}\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}}^{k-1}\prod_{i=1}^{k}\|\Delta\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L^k), \quad \forall l \in [L].$$

On the other hand, the highest k-order update condition is $\alpha_l \prod_{i=1}^{k} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L)$ for all $l \in [L]$. Combining the two relations immediately gives

$$\alpha_l \prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall\, l \in [L],$$

which refines the preliminary initialization condition.

Finally, as in the two-layer case, we prove that *the refined initial condition and the first-order update condition can derive any j-order ($j \geq 2$) update condition on hidden weights*. Thus, retaining the refined initial condition and the first-order update condition in Condition B.2 is sufficient.

Formally, for any $S \subseteq [k],\ |S| = j,\ j \in [k],\ l \in [L]$, we need to prove that

$$\alpha_l \prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{i \notin S} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L)$$

based on the refined initial condition and the first-order update condition. By multiplying the first-order update conditions, we have

$$\begin{aligned}
\frac{1}{L^j} &= \prod_{i \in S} \left(\alpha_l \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{m \neq i} \|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} \right) \\
&= \alpha_l^j \left(\prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right) \left(\prod_{i \in S} \prod_{m \neq i} \|\boldsymbol{W}_l^{(m)}\|_{\mathrm{R}} \right) \\
&= \alpha_l^j \left(\prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right) \left(\prod_{i \notin S} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right) \left(\prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right)^{j-1} \\
&= \left(\alpha_l \prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{i \notin S} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right) \left(\alpha_l \prod_{i=1}^{k} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right)^{j-1} \\
&= \left(\alpha_l \prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{i \notin S} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \right) \cdot \frac{1}{L^{j-1}},
\end{aligned}$$

which implies that $\alpha_l \prod_{i \in S} \|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} \prod_{i \notin S} \|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1/L)$, which finishes the derivation. Therefore, for any fixed $k \geq 2$, the spectral constraints reduce to the same implementation as in the $k = 2$ case: hidden residual multiplier $\alpha_l = \Theta(1/L)$ and hidden update norms $\|\Delta \boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ under standard width-scaling μP initialization.

B.3 Bias Parameters

The purpose of this subsection is not to introduce a new depth-scaling rule, but to show that bias parameters can be assigned order-one initialization and update scales once the matrix-weight parameterization from Condition 3.1 is in place.

B.3.1 Problem Setup

As shown in Appendix B.2, residual blocks with an arbitrary fixed internal depth $k \geq 2$ admit spectral scaling conditions that are algorithmically equivalent to the $k = 2$ case. Therefore, to illustrate that bias parameters do not change the main μP prescription, we analyze the representative two-layer residual block

with additive biases. Specifically, we consider a residual network whose forward propagation is given by

$$\begin{aligned}\boldsymbol{h}_0(\boldsymbol{x}) &= \alpha_0\big(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0\big),\\ \boldsymbol{h}_l(\boldsymbol{x}) &= \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l\Big(\boldsymbol{W}_l^{(2)}\big(\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \boldsymbol{b}_l^{(1)}\big) + \boldsymbol{b}_l^{(2)}\Big), \quad \forall\, l \in [L],\\ \boldsymbol{h}_{L+1}(\boldsymbol{x}) &= \alpha_{L+1}\boldsymbol{W}_{L+1}\boldsymbol{h}_L(\boldsymbol{x}).\end{aligned}$$

Here, each residual block consists of a two-layer linear transformation with additive biases, where $\boldsymbol{W}_l^{(1)}, \boldsymbol{W}_l^{(2)}$ denote the weight matrices and $\boldsymbol{b}_l^{(1)}, \boldsymbol{b}_l^{(2)}$ denote the corresponding bias vectors within the l-th block. The scalars $\{\alpha_l\}_{l=0}^{L+1}$ represent block multipliers that control the effective strength of each transformation. As in the previous sections, $\boldsymbol{h}_{L+1}(\boldsymbol{x})$ denotes the network output used to compute the loss. Our goal is to derive a spectral condition that realizes μP Principle 2.1 in this setting.

B.3.2 Spectral Scaling Condition

We now state the spectral scaling condition for the residual network with biases for realizing the μP principle under joint width–depth scaling.

Condition B.3 (Spectral condition for μP under joint width-depth scaling, two-layer residual block with biases)**.** *To realize μP Principle 2.1, the initial parameters and their per-step updates should satisfy:*

- ***Initial condition.***
  - *Input parameters:* $\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$, $\alpha_0\|\boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$.
  - *Hidden parameters:*
    - $\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}), \quad \forall l \in [L]$.
  - *Output parameters:* $\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
- ***Update condition.***
  - *Input parameters:* $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$, $\alpha_0\|\Delta\boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$.
  - *Hidden parameters (first-order):*
    - $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\Delta\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\Delta\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
  - *Hidden parameters (second-order), automatically satisfied given initial condition and first-order update conditions:*
    - $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
    - $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\Delta\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L]$.
  - *Output parameters:* $\alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.
- ***Efficient implementation.*** *Under the μP parameterization ($k = 2$) of block multipliers $\{\alpha_l\}$ and matrix weights $\{\boldsymbol{W}_l\}$ described in Section 4 and Appendix C, all bias-related spectral conditions can be satisfied simultaneously by initializing and training with biases of order $\Theta(1)$. Concretely, it is sufficient to enforce*

$$\|\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1), \quad \|\Delta\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1), \quad \forall\, 0 \le l \le L. \tag{13}$$

The initial condition $\|\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ can be satisfied by setting $\sigma_{\boldsymbol{b}_l} = \Theta(1)$, and the implementation for update condition $\|\Delta\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ will be derived in Appendix C.

Condition B.3 shows that, under joint width-depth scaling, bias parameters can be incorporated without modifying the existing HP parameterization of the weight matrices. Specifically, once the block multipliers $\{\alpha_l\}$ and weights $\{\boldsymbol{W}_l\}$ are implemented as in Section 4, biases admit additional, simple order-one spectral conditions that guarantee their initialization and updates scale properly. Thus, biases can be handled by lightweight extensions of our framework, while the μP formulation for bias-free residual blocks remains unchanged.

B.3.3 Derivation for Preliminary Initialization Condition

Input layer. By subadditivity and submultiplicativity,

$$\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0(\|\boldsymbol{W}_0\|_{\mathrm{R}}\|\boldsymbol{x}\|_{\mathrm{R}} + \|\boldsymbol{b}_0\|_{\mathrm{R}})) = \Theta\left(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}\right) + \Theta\left(\alpha_0\|\boldsymbol{b}_0\|_{\mathrm{R}}\right),$$

where we assumed $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$. Thus, choosing $\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}, \alpha_0\|\boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$ ensures $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. Expanding the residual recursion yields

$$\boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \Big(\boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} + \boldsymbol{b}_l^{(2)} \Big). \tag{14}$$

Applying subadditivity, we can estimate their order as

$$\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} + \sum_{l=1}^{s} \alpha_l \boldsymbol{b}_l^{(2)} \Big\|_{\mathrm{R}} \right).$$

Since we have $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, it suffices to ensure other terms are $\mathcal{O}(1)$ for any $s \in [L]$ to preserve $\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. Under i.i.d. zero-mean Gaussian initialization, the summands are approximately independent zero-mean random vectors (Yang & Hu, 2021; Yang et al., 2024; Dey et al., 2025), so the typical squared RMS norm of their sum scales as the sum of the squared RMS norms (see Theorem 3.3.1 in Vershynin (2018)). Therefore, we can obtain

$$\begin{aligned}
&\Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} + \sum_{l=1}^{s} \alpha_l \boldsymbol{b}_l^{(2)} \Big\|_{\mathrm{R}} \\
&= \sqrt{ \Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) \Big\|_{\mathrm{R}}^2 + \Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} \Big\|_{\mathrm{R}}^2 + \Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{b}_l^{(2)} \Big\|_{\mathrm{R}}^2 }
\end{aligned}$$

Furthermore, using the same probability argument and submultiplicativity inequality as in the derivation without biases (e.g., see Section 3), we have

$$\begin{aligned}
\Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) \Big\|_{\mathrm{R}}^2 &= \Theta\left(\sum_{l=1}^{s} \alpha_l^2 \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}^2 \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}^2 \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}^2 \right), \\
\Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} \Big\|_{\mathrm{R}}^2 &= \Theta\left(\sum_{l=1}^{s} \alpha_l^2 \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}^2 \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}}^2 \right), \\
\Big\| \sum_{l=1}^{s} \alpha_l \boldsymbol{b}_l^{(2)} \Big\|_{\mathrm{R}}^2 &= \Theta\left(\sum_{l=1}^{s} \alpha_l^2 \|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}}^2 \right).
\end{aligned}$$

Therefore, starting from $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, imposing

$$\alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}), \quad \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L}), \quad \alpha_l \|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \mathcal{O}(1/\sqrt{L})$$

recursively ensures $\|\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $s \in [L]$. This yields a preliminary initialization condition, which will be refined after incorporating update constraints.

Output layer. The same argument as in the two-layer residual block case yields $\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.

B.3.4 Derivation for Update Condition

Input layer. Recall that

$$\boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0\big(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0\big).$$

After one gradient step, the feature update satisfies

$$\Delta\boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0\big(\Delta\boldsymbol{W}_0\,\boldsymbol{x} + \Delta\boldsymbol{b}_0\big).$$

By subadditivity, submultiplicativity, and using $\|\boldsymbol{x}\|_{\mathrm{R}} = \Theta(1)$ by data assumption, we obtain

$$\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}} + \alpha_0\|\Delta\boldsymbol{b}_0\|_{\mathrm{R}}).$$

Therefore, we choose

$$\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1), \quad \alpha_0\|\Delta\boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$$

to realize $\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

Hidden layers. We next analyze the feature updates $\Delta\boldsymbol{h}_s(\boldsymbol{x})$ after one gradient step. Expanding Equation (14) yields

$$\begin{aligned}\Delta\boldsymbol{h}_s(\boldsymbol{x}) &= \Delta\boldsymbol{h}_0(\boldsymbol{x}) + \Delta\sum_{l=1}^{s}\alpha_l\Big(\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \boldsymbol{W}_l^{(2)}\boldsymbol{b}_l^{(1)} + \boldsymbol{b}_l^{(2)}\Big)\\ &= \Delta\boldsymbol{h}_0(\boldsymbol{x}) + \Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x}) + \Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{b}_l^{(1)} + \Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{b}_l^{(2)}.\end{aligned}$$

By the subadditivity of vector norms, we have

$$\begin{aligned}&\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}}\\ &= \Theta\left(\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \left\|\Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})\right\|_{\mathrm{R}} + \left\|\Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{b}_l^{(1)}\right\|_{\mathrm{R}} + \left\|\Delta\sum_{l=1}^{s}\alpha_l\boldsymbol{b}_l^{(2)}\right\|_{\mathrm{R}}\right).\end{aligned}$$

Since $\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by the input-layer update, we have $\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Omega(1)$ for all $s \in [L]$. Moreover, by subadditivity, the remaining terms do not decay with depth, implying $\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \mathcal{O}(\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}})$ for any $s \in [L]$. Therefore, to enforce Principle 2.1, it suffices to require $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while satisfying Principle (P2). We discuss the components of $\Delta\boldsymbol{h}_L(\boldsymbol{x})$ in sequence.

Matrix-weight terms. The contributions from

$$\Delta\boldsymbol{h}_0(\boldsymbol{x}) + \Delta\sum_{l=1}^{L}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x})$$

have been fully analyzed in the bias-free two-layer case (see Section 3). Applying the same reasoning yields the first- and second-order update conditions on hidden matrix weights:

$$\begin{aligned}\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L),\\ \alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L),\\ \alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\Delta\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \qquad \forall\, l \in [L].\end{aligned}$$

We then need to control the newly introduced bias-related terms.

First-layer bias-related term. Consider $\Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)}$. Expanding the update yields

$$\Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} = \sum_{l=1}^{L} \alpha_l \Big(\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} + \boldsymbol{W}_l^{(2)} \Delta \boldsymbol{b}_l^{(1)} + \Delta \boldsymbol{W}_l^{(2)} \Delta \boldsymbol{b}_l^{(1)} \Big).$$

By subadditivity and submultiplicativity of the RMS norm, we have

$$\begin{aligned} &\left\| \Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} \right\|_{\mathrm{R}} \\ &= \Theta\Big(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} + \sum_{l=1}^{L} \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} + \sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} \Big). \end{aligned}$$

According to Principle (P2), we require $\left\| \Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{b}_l^{(1)} \right\|_{\mathrm{R}} = \Theta(1)$ and maximize the contribution from each summand, leading to

$$\begin{aligned} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \\ \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \\ \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \qquad \forall\, l \in [L]. \end{aligned}$$

Second-layer bias-related term. Finally, for $\Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{b}_l^{(2)}$, we have

$$\left\| \Delta \sum_{l=1}^{L} \alpha_l \boldsymbol{b}_l^{(2)} \right\|_{\mathrm{R}} = \left\| \sum_{l=1}^{L} \alpha_l \Delta \boldsymbol{b}_l^{(2)} \right\|_{\mathrm{R}} = \Theta\Big(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} \Big).$$

To maximally update parameters according to Principle (P2), we require this term to remain $\Theta(1)$ and maximize each summand, which yields

$$\alpha_l \|\Delta \boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L), \qquad \forall\, l \in [L].$$

Output layer. The same argument as in the two-layer case in Section 3 yields $\alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.

B.3.5 Derivation for Final Initial Condition

We now derive the final initialization conditions by incorporating the update constraints obtained in the previous subsection.

Hidden matrix weights. As already shown in the bias-free setting (Section 3), combining the first-order and second-order update conditions immediately yields the initialization constraint

$$\alpha_l \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \, \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall\, l \in [L].$$

Therefore, the presence of biases does not alter the initialization scaling of hidden matrix weights.

Bias parameters. We now derive the initialization conditions for bias terms by combining the first- and second-order update constraints. For the first-layer bias $\boldsymbol{b}_l^{(1)}$, the first-order update conditions give

$$\begin{aligned} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \\ \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} &= \Theta(1/L), \end{aligned}$$

while the second-order update condition yields

$$\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L).$$

Multiplying the two first-order conditions and dividing by the second-order one, we obtain

$$\alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L), \quad \forall l \in [L].$$

Similar to the hidden matrix weights, the second-order bias-related condition is automatically satisfied by combining the refined initial condition and the corresponding first-order update condition.

We note that the second-layer bias $\boldsymbol{b}_l^{(2)}$ has no multiplicative interaction with another parameter in the forward block, so its initialization condition remains the preliminary upper bound $\alpha_l \|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = O(1/\sqrt{L})$.

B.3.6 Derivation for Efficient Implementation

Recall that, based on the μP parameterization ($k = 2$) introduced for matrix weights in Section 4 and Appendix C, we have $\alpha_0 = \Theta(1)$ in Equation (7), $\alpha_l = \Theta(1/L)$ for $l \in [L]$ in Equation (8), $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ for $0 \le l \le L$ in Equation (6) and $\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ for $0 \le l \le L$. Based on these conditions, Condition B.3 reduces to

- **Initial condition.**
  - Input parameters: $\|\boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$.
  - Hidden parameters:
    - $\|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.
    - $\|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \mathcal{O}(\sqrt{L}), \quad \forall l \in [L]$.
- **Update condition.**
  - Input parameters: $\|\Delta \boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$.
  - Hidden parameters (first-order):
    - $\|\Delta \boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.
    - $\|\Delta \boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.

Therefore, it is sufficient to enforce the order-one spectral condition for biases:

$$\|\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1), \quad \|\Delta \boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1), \quad \forall\, 0 \le l \le L.$$

Although the preliminary condition permits $\|\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}}$ as large as $\mathcal{O}(\sqrt{L})$, choosing it to be $\Theta(1)$ is a simpler sufficient choice and keeps all biases on the same scale.

C Implementing Condition 3.1 for Optimizers with Weight Decay

Recall that in Section 4.1 of the main text, we implemented Condition 3.1 for initialization and specified the parameterization of the block multipliers α_l and the initialization variances σ_l, which is optimizer-agnostic. In Section 4.2, we further implemented Condition 3.1 for updates and derived the parameterization of the learning rates η_l for the Muon-Kimi (Liu et al., 2025). We now extend this update-condition analysis to a broader class of optimizers with weight decay. For bias parameters, we also derive the corresponding bias HPs when the optimizer is commonly applied to biases.

C.1 Preparation

Unified update form with weight decay. To provide a unified derivation across different optimizers, we begin by expressing their one-step update rules in a common form. When weight decay is included, a single update step of the weight matrix can be written as

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\boldsymbol{A}_l + \lambda_l \boldsymbol{W}_l\big),$$

where $\boldsymbol{A}_l$ denotes the optimizer-specific update direction before applying the learning rate and excluding weight decay. For example, $\boldsymbol{A}_l = \nabla_{\boldsymbol{W}_l}\mathcal{L}$ for SGD, while for Muon-Kimi $\boldsymbol{A}_l = 0.2\sqrt{\max\{n_{\text{in}}, n_{\text{out}}\}}\,\boldsymbol{U}_l\boldsymbol{V}_l^\top$. The scalar λ_l is the weight-decay coefficient.

The update magnitude $\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \eta_l\|\boldsymbol{A}_l + \lambda_l\boldsymbol{W}_l\|_{\text{R}}$ is required to satisfy the update conditions in Condition 3.1. We analyze this requirement under two complementary regimes.

Without weight decay. When weight decay is disabled ($\lambda_l = 0$), the update reduces to $\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \eta_l\|\boldsymbol{A}_l\|_{\text{R}}$. In this case, the learning rate is chosen so that

$$\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \eta_l\|\boldsymbol{A}_l\|_{\text{R}} \text{ satisfies Condition 3.1.} \tag{Δ1}$$

With weight decay. When weight decay is enabled ($\lambda_l \neq 0$), we choose the weight decay term to be comparable in scale to the optimizer-driven term.

$$\|\lambda_l\boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right). \tag{Δ2}$$

This is a non-degenerate implementation convention: it keeps weight decay active in the update dynamics without letting it dominate the optimizer-driven term. Under the usual scale estimate, $\|\boldsymbol{A}_l + \lambda_l\boldsymbol{W}_l\|_{\text{R}} = \Theta(\|\boldsymbol{A}_l\|_{\text{R}})$, so the learning-rate scaling rule derived from Equation (Δ1) is preserved.

Weights and biases. In the following, we derive parameterizations of the learning rate and weight decay coefficient for a range of optimizers. Since matrix-based optimizers such as Muon and Shampoo are typically not used for bias parameters, we restrict our analysis of bias parameterization to vector-based optimizers such as SGD and AdamW. The same two rules, Equations (Δ1) and (Δ2), are then applied to the biases with corresponding spectral condition in Equation 13 on vector RMS norms.

Momentum. We note that momentum is typically omitted in standard μP analyses (Yang & Hu, 2021; Yang & Littwin, 2023; Yang et al., 2022) (e.g., by setting $\beta_1 = \beta_2 = 0$ in AdamW), while in practical μP implementations the momentum coefficients are taken to be $\Theta(1)$. The main rationale is that the norm of the momentum term and the norm of the current update are expected to be of the same order, and since the spectral condition aims to control the update scale, omitting the momentum is regarded as an acceptable simplification. Moreover, analyzing the update without momentum can be interpreted as studying the *first update step* after initialization, which has been empirically observed to be reliable for understanding neural network training (Yang et al., 2023; Ngom et al., 2025; Bordelon & Pehlevan, 2022; Bordelon et al., 2024a). In the subsequent derivations, we adopt this simplification as well.

Low-rank structure of updates. Following the common practice used in the μP spectral analysis (Yang et al., 2023; Ngom et al., 2025), we also introduce a useful preliminary result for controlling the norm of the update term $\boldsymbol{A}_l$. The key idea is to exploit the effective *low-rank structure* of neural-network updates, which has been widely observed in neural network training (Yang et al., 2023; Balzano et al., 2025; Zhao et al., 2024): only a small number of dominant singular directions carry most of the update.

Assumption C.1 (Low-rank update structure)**.** For vector-based optimizers such as SGD and AdamW, we assume that the effective update term $\boldsymbol{A}_l$ has constant rank with respect to width and depth, i.e., $r(\boldsymbol{A}_l) = \Theta(1)$.

Under this assumption, the spectral norm and Frobenius norm of the update term are of the same order:

$$\|\boldsymbol{A}_l\|_2 = \Theta(\|\boldsymbol{A}_l\|_{\text{F}}), \tag{15}$$

where the hidden constants are independent of width and depth. Indeed, this is because

$$\|\boldsymbol{A}_l\|_2 \leq \|\boldsymbol{A}_l\|_{\text{F}} \leq \sqrt{r(\boldsymbol{A}_l)}\|\boldsymbol{A}_l\|_2 = \Theta(\|\boldsymbol{A}_l\|_2).$$

We will use this low-rank update property to estimate the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ for vector-based optimizers.

Table 2: **Overview of μP implementation from Condition 3.1 ($k = 2$) for optimizer families with weight decay.** Optimizers in the same row share the same μP scaling rules.

Optimizer family	Detailed parameterization
Muon-Kimi	Table 3
Muon / Shampoo / SOAP	Table 4
SGD	Table 5
AdamW / Lion / Sophia	Table 6
SSO	Table 7

Table 3: **μP implementation of Condition 3.1 ($k = 2$) for Muon-Kimi (Liu et al., 2025) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	η_{base}	$\eta_{\text{base}}/\sqrt{r_n}$ (η_{base})	η_{base}
Weight Decay	λ_{base}	$\lambda_{\text{base}}\sqrt{r_n}$ (λ_{base})	λ_{base}

C.2 Overview

Table 2 summarizes the optimizer families covered in this section and points to the corresponding detailed μP parameterization tables. All rows use the optimizer-agnostic initialization and block-multiplier rules from Section 4.1; the table only organizes the optimizer-dependent update rules derived below.

C.3 Muon-Kimi

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the update rule of Muon-Kimi (Liu et al., 2025) with weight decay is

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{0.2\sqrt{\max\{n_{\text{in}}, n_{\text{out}}\}}\, \boldsymbol{U}_l \boldsymbol{V}_l^\top}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big),$$

where $\boldsymbol{U}_l, \boldsymbol{V}_l$ arise from the compact SVD of the gradient $\nabla_{\boldsymbol{W}_l} \mathcal{L} = \boldsymbol{U}_l \boldsymbol{\Sigma}_l \boldsymbol{V}_l^\top$.

Recalling that in Section 4.2 in the main text, we have derived the learning rate parameterizations to achieve ($\Delta 1$) that

$$\eta_0 = \Theta(1), \ \eta_l^{(1)} = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}}}\right), \ \eta_l^{(2)} = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}}}\right), \ \eta_{L+1} = \Theta(1).$$

According to the update norm in Equation (9), we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \Theta\left(\sqrt{n_{\text{in}}} \max\left\{1, \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\right\}\right) = \begin{cases} \Theta(1), & l = 0, \\ \Theta(\sqrt{n_{\text{in}}}), & l \in [L], \\ \Theta(n_{\text{in}}), & l = L+1. \end{cases}$$

Table 4: **μP implementation of Condition 3.1 ($k = 2$) for Muon (Jordan et al., 2024b), Shampoo (Gupta et al., 2018), and SOAP (Vyas et al., 2024) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	$\eta_{\text{base}}\sqrt{r_n}$ (η_{base})	η_{base}	$\eta_{\text{base}}\sqrt{r_n}$ (η_{base})
Weight Decay	$\lambda_{\text{base}}/\sqrt{r_n}$ (λ_{base})	λ_{base}	$\lambda_{\text{base}}/\sqrt{r_n}$ (λ_{base})
Shampoo ε	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_L^2$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$

Given the magnitude of $\|\boldsymbol{W}_l\|_{\text{R}}$ in Equation (6), as desired $\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right)$ by (Δ2), the parameterizations of λ_l need to be set as follows:

$$\lambda_l = \begin{cases} \Theta(1), & l = 0, \\ \Theta(\sqrt{n_{\text{in}}}), & l \in [L], \\ \Theta(1), & l = L+1. \end{cases}$$

This completes the implementation of the update condition for Muon-Kimi with weight decay, as summarized in Table 3.

C.4 Muon

In this section, we derive the μP implementation from Condition 3.1 for Muon, which recovers and extends the μP scaling rules studied in Qiu et al. (2025).

C.4.1 Update Rule

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the update rule of Muon (Jordan et al., 2024b) is

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{\boldsymbol{U}_l \boldsymbol{V}_l^\top}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big), \tag{16}$$

where $\boldsymbol{U}_l, \boldsymbol{V}_l$ arise from the compact SVD of the gradient $\nabla_{\boldsymbol{W}_l} \mathcal{L} = \boldsymbol{U}_l \boldsymbol{\Sigma}_l \boldsymbol{V}_l^\top$. Compared with Muon-Kimi (Liu et al., 2025), the only difference lies in the absence of the $0.2\sqrt{\max\{n_{\text{in}}, n_{\text{out}}\}}$ prefactor.

Considering the dimension assumption in Equation (3), the resulting norm of $\boldsymbol{A}_l$ satisfies

$$\|\boldsymbol{A}_l\|_{\text{R}} = \|\boldsymbol{U}_l \boldsymbol{V}_l^\top\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\boldsymbol{U}_l \boldsymbol{V}_l^\top\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} = \begin{cases} \Theta(1/\sqrt{n_{\text{out}}}), & l = 0, \\ \Theta(1), & l \in [L], \\ \Theta(\sqrt{n_{\text{in}}}), & l = L+1. \end{cases} \tag{17}$$

C.4.2 Derivation of Parameterization

Input and output layers. When weight decay is disabled ($\lambda_0 = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the multiplier parameterizations $\alpha_0 = \Theta(1), \alpha_{L+1} = \Theta(1/n_{\text{in}})$ in Equation (7), and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ in Equation (17), we have

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \alpha_l \eta_l \|\boldsymbol{A}_l\|_{\text{R}} = \begin{cases} \Theta(\eta_0/\sqrt{n_{\text{out}}}), & l = 0, \\ \Theta(\eta_{L+1}/\sqrt{n_{\text{in}}}), & l = L+1. \end{cases}$$

As desired in (Δ1), to satisfy (C2.1) that $\alpha_0\|\Delta \boldsymbol{W}_0\|_{\mathrm{R}}, \alpha_{L+1}\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$, we need to set

$$\eta_0 = \Theta(\sqrt{n_{\text{out}}}), \quad \eta_{L+1} = \Theta(\sqrt{n_{\text{in}}}).$$

When $\lambda_l \neq 0$, given Equation (6) that $\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\text{in}})$, we have

$$\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \begin{cases} \Theta(\lambda_0), & l = 0, \\ \Theta(\lambda_{L+1} n_{\text{in}}), & l = L+1. \end{cases}$$

To satisfy (Δ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$, we need to set

$$\lambda_0 = \Theta(1/\sqrt{n_{\text{out}}}), \quad \lambda_{L+1} = \Theta(1/\sqrt{n_{\text{in}}}),$$

Hidden layers (first-order). When weight decay is disabled ($\lambda_l = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (8), and the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ in Equation (17), we have

$$\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L) \cdot \eta_l^{(2)} \|\boldsymbol{A}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(\eta_l^{(2)}/L).$$

As desired in (Δ1), to satisfy the first-order update condition on hidden weights (C2.2) that $\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \, \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$, we need to set

$$\eta_l^{(2)} = \Theta(1).$$

When weight decay is enabled ($\lambda_l \neq 0$), given the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l^{(2)} \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(\lambda_l^{(2)}).$$

To satisfy (Δ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$ we need to set

$$\lambda_l^{(2)} = \Theta(1).$$

Symmetrically, we have the same choice for $\boldsymbol{W}_l^{(1)}$:

$$\eta_l^{(1)} = \Theta(1), \quad \lambda_l^{(1)} = \Theta(1).$$

Hidden layers (second-order). As discussed in Section 3.3.3, the second-order update condition is satisfied automatically once the initial condition and the first-order update condition are met. We explain here again for clarity: Multiplying two equations in (C2.2) gives $\alpha_l^2 \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \, \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\Delta \boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L^2)$. Combining this with (C1.2) that $\alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \, \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$ directly implies the second-order condition (C2.3).

This completes the implementation of the update condition for Muon with weight decay, which is summarized in Table 4.

C.5 SGD

In this section, we derive the μP implementation from Condition 3.1 for SGD, which recovers and extends the μP scaling rules studied in Bordelon et al. (2024a).

C.5.1 Update Rule

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the SGD update rule with weight decay can be written as:

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{\nabla_{\boldsymbol{W}_l} \mathcal{L}}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big).$$

Table 5: **μP implementation of Condition 3.1 ($k = 2$) for SGD with weight decay under width–depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image. The initial variance of input bias is σ^2_{base}.

	Input weights & biases	Hidden weights	Output weights	Hidden biases
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})	α_{base}/r_L (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}	σ^2_{base}
Learning Rate	$\eta_{\text{base}} r_n$ (η_{base})	$\eta_{\text{base}} r_L$ (η_{base})	$\eta_{\text{base}} r_n$ (η_{base})	$\eta_{\text{base}} r_L r_n$ (η_{base})
Weight Decay	$\lambda_{\text{base}}/r_n$ (λ_{base})	$\lambda_{\text{base}}/r_L$ (λ_{base})	$\lambda_{\text{base}}/r_n$ (λ_{base})	$\lambda_{\text{base}}/(r_L r_n)$ (λ_{base})

Here, we estimate the scale of raw gradient $\nabla_{\boldsymbol{W}_l}\mathcal{L}$ following the spectral argument of Yang et al. (2023). From the derivation of the update condition in Section 3, we can observe that gradient updates $\Delta\boldsymbol{W}_0, \Delta\boldsymbol{W}_{L+1}$ induce a change $\|\Delta\boldsymbol{h}_{L+1}\|_{\text{R}} = \Theta(1)$ in the output, which induces a change $\Delta\mathcal{L} = \Theta(1)$ for common loss functions $\mathcal{L}$. In contrast, each hidden gradient update $\Delta\boldsymbol{W}_l$ ($l \in [L]$) induces a change $\|\Delta\boldsymbol{h}_{L+1}\|_{\text{R}} = \Theta(1/L)$ in the output, which induces a change $\Delta\mathcal{L} = \Theta(1/L)$. We use these properties to derive the scale of $\nabla_{\boldsymbol{W}_l}\mathcal{L}$ as follows.

For the input weights, we have

$$\Theta(1) = \Delta_{\boldsymbol{W}_0}\mathcal{L} = \Theta(\langle\Delta\boldsymbol{W}_0, \nabla_{\boldsymbol{W}_0}\mathcal{L}\rangle) = \Theta(\|\Delta\boldsymbol{W}_0\|_{\text{F}}\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_{\text{F}}) = \Theta(\|\Delta\boldsymbol{W}_0\|_2\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_2),$$

where $\langle\cdot,\cdot\rangle$ denotes the trace inner product, and we use the facts that the two arguments of the inner product are proportional to each other and the low-rank property of $\boldsymbol{A}_l$ in Equation (15). Since we finally realize the spectral condition (C2.1) that $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\text{R}} = \Theta(1)$ and use $\alpha_0 = \Theta(1)$ by initial implementation in Equation (7), we have $\|\Delta\boldsymbol{W}_0\|_{\text{R}} = \Theta(1)$ so $\|\Delta\boldsymbol{W}_0\|_2 = \Theta(\sqrt{n_{\text{out}}/n_{\text{in}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_2 = \Theta(\sqrt{n_{\text{in}}/n_{\text{out}}})$, which leads to

$$\|\boldsymbol{A}_0\|_{\text{R}} = \|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_2 = \Theta(\frac{n_{\text{in}}}{n_{\text{out}}}) = \Theta(\frac{1}{n_{\text{out}}}).$$

Similarly, for the hidden weight $\boldsymbol{W}_l$, we have

$$\Theta(\frac{1}{L}) = \Delta_{\boldsymbol{W}_l}\mathcal{L} = \Theta(\langle\Delta\boldsymbol{W}_l, \nabla_{\boldsymbol{W}_l}\mathcal{L}\rangle) = \Theta(\|\Delta\boldsymbol{W}_l\|_{\text{F}}\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\text{F}}) = \Theta(\|\Delta\boldsymbol{W}_l\|_2\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2),$$

Since we finally set $\alpha_l\|\Delta\boldsymbol{W}_l\|_{\text{R}}\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1/L)$ to satisfy the update condition (C2.2), and use $\alpha_l = \Theta(1/L)$ in Equation (8), $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ in Equation (6) by initial implementation, we have $\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ so $\|\Delta\boldsymbol{W}_l\|_2 = \Theta(\sqrt{n_{\text{out}}/n_{\text{in}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2 = \Theta(L^{-1}\sqrt{n_{\text{in}}/n_{\text{out}}})$, which leads to

$$\|\boldsymbol{A}_l\|_{\text{R}} = \|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2 = \Theta(\frac{n_{\text{in}}}{Ln_{\text{out}}}) = \Theta(\frac{1}{L}).$$

Finally, for the output weight $\boldsymbol{W}_{L+1}$, we have

$$\Theta(1) = \Delta_{\boldsymbol{W}_{L+1}}\mathcal{L} = \Theta(\langle\Delta\boldsymbol{W}_{L+1}, \nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\rangle) = \Theta(\|\Delta\boldsymbol{W}_{L+1}\|_{\text{F}}\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_{\text{F}}) = \Theta(\|\Delta\boldsymbol{W}_{L+1}\|_2\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_2).$$

Since we will set $\alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(1)$ to realize the update condition (C2.1) and use $\alpha_{L+1} = \Theta(1/n_{\text{in}})$ in initial implementation in Equation (7), we have $\|\Delta\boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(n_{\text{in}})$ so $\|\Delta\boldsymbol{W}_{L+1}\|_2 = \Theta(\sqrt{n_{\text{out}}n_{\text{in}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_2 = \Theta(1/\sqrt{n_{\text{in}}n_{\text{out}}})$, which leads to

$$\|\boldsymbol{A}_{L+1}\|_{\text{R}} = \|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_2 = \Theta(\frac{1}{n_{\text{out}}}) = \Theta(1).$$

To sum up, we have

$$\|\boldsymbol{A}_l\|_{\mathrm{R}} = \|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\mathrm{R}} = \begin{cases} \Theta(1/n_{\mathrm{out}}), & l = 0, \\ \Theta(1/L), & l \in [\,L\,], \\ \Theta(1), & l = L+1. \end{cases} \tag{18}$$

C.5.2 Derivation of Parameterization

Input and output layers. When weight decay is disabled ($\lambda_0 = 0$), using the dimension assumptions $d_0, d_{L+1} = \Theta(1)$ and $n_l = \Theta(n)$ in Equation (3), together with the initialization parameterization $\alpha_0 = \Theta(1)$ and $\alpha_{L+1} = \Theta(1/n_{\mathrm{in}})$ in Equation (7), we obtain

$$\alpha_l\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \alpha_l\eta_l\|\boldsymbol{A}_l\|_{\mathrm{R}} = \begin{cases} \Theta\big(\eta_0/n_{\mathrm{out}}\big), & l = 0, \\ \Theta\big(\eta_{L+1}/n_{\mathrm{in}}\big), & l = L+1. \end{cases}$$

To satisfy the input/output update requirement $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}},\ \alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$ in Condition (C2.1), we therefore choose

$$\eta_0 = \Theta(n_{\mathrm{out}}), \quad \eta_{L+1} = \Theta(n_{\mathrm{in}}).$$

When weight decay is active ($\lambda_l \neq 0$), using $\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\mathrm{in}})$ from the initialization implementation in Equation (6), we have

$$\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \begin{cases} \Theta(\lambda_0), & l = 0, \\ \Theta(\lambda_{L+1}n_{\mathrm{in}}), & l = L+1. \end{cases}$$

Matching this to $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ as desired by condition ($\Delta 2$) yields

$$\lambda_0 = \Theta(1/n_{\mathrm{out}}), \quad \lambda_{L+1} = \Theta(1/n_{\mathrm{in}}).$$

Hidden layers (first-order). For a hidden block we have implemented $\alpha_l = \Theta(1/L)$ in Equation (8) and $\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ in Equation (6). When $\lambda_l = 0$ we obtain

$$\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)\cdot\eta_l^{(2)}\|\boldsymbol{A}_l^{(2)}\|_{\mathrm{R}} = \Theta(\eta_l^{(2)}/L^2).$$

Enforcing the first-order hidden update condition (C2.2) that $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$ gives

$$\eta_l^{(2)} = \Theta(L).$$

By the same reasoning, the same choice applies to the other learning rate, $\eta_l^{(1)} = \Theta(L)$.

If weight decay is enabled on hidden matrices, using $\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ by Equation (6), we obtain $\|\lambda_l^{(i)}\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(\lambda_l^{(i)})$, so condition ($\Delta 2$) that $\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right) = \Theta(1/L)$ implies the natural choice

$$\lambda_l^{(i)} = \Theta(1/L), \qquad i = 1, 2.$$

Hidden layers (second-order). As illustrated in Section 3.3.3 and Appendix C.4, the second-order update condition is satisfied automatically once the initial condition and the first-order update condition are met.

Biases. For biases, we use the spectral bias condition in Equation (13), which sets $\|\Delta \boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ for the input and hidden biases under Condition 3.1. We estimate the corresponding raw-gradient scales in the same way as for matrix weights. For the input biases $\boldsymbol{b}_0 \in \mathbb{R}^{n_{\mathrm{out}} \times 1}$, we have

$$\Theta(1) = \Delta_{\boldsymbol{b}_0}\mathcal{L} = \Theta(\langle \Delta \boldsymbol{b}_0, \nabla_{\boldsymbol{b}_0}\mathcal{L}\rangle) = \Theta(\|\Delta \boldsymbol{b}_0\|_2 \|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_2),$$

Since we finally set $\|\Delta \boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(1)$ to satisfy the update condition in Equation (13), we have $\|\Delta \boldsymbol{b}_0\|_2 = \Theta(\sqrt{n_{\mathrm{out}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_2 = \Theta(1/\sqrt{n_{\mathrm{out}}})$, which leads to

$$\|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_{\mathrm{R}} = \sqrt{\frac{1}{n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_2 = \Theta(\frac{1}{n_{\mathrm{out}}}).$$

Similarly, for the hidden biases $\boldsymbol{b}_l \in \mathbb{R}^{n_{\mathrm{out}} \times 1}$, we have

$$\Theta(1/L) = \Delta_{\boldsymbol{b}_l}\mathcal{L} = \Theta(\langle \Delta \boldsymbol{b}_l, \nabla_{\boldsymbol{b}_l}\mathcal{L}\rangle) = \Theta(\|\Delta \boldsymbol{b}_l\|_2 \|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2),$$

Since we finally set $\|\Delta \boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ to satisfy the update condition in Equation (13), we have $\|\Delta \boldsymbol{b}_l\|_2 = \Theta(\sqrt{n_{\mathrm{out}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2 = \Theta(1/(L\sqrt{n_{\mathrm{out}}}))$, which leads to

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}} = \sqrt{\frac{1}{n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2 = \Theta(\frac{1}{Ln_{\mathrm{out}}}).$$

To sum up, we can estimate the scale of $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}}$ as

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}} = \begin{cases} \Theta(1/n_{\mathrm{out}}), & l = 0, \\ \Theta(1/(Ln_{\mathrm{out}})), & l \in [L]. \end{cases}$$

Requiring $\|\Delta \boldsymbol{b}_l\|_{\mathrm{R}} = \eta_{\boldsymbol{b}_l}\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}} = \Theta(1)$ according to update condition in Equation (13) leads to the learning rate as

$$\eta_{\boldsymbol{b}_l} = \begin{cases} \Theta(n_{\mathrm{out}}), & l = 0, \\ \Theta(Ln_{\mathrm{out}}), & l \in [L]. \end{cases}$$

For the weight decays, we need to satisfy $\lambda_{\boldsymbol{b}_l}\|\boldsymbol{b}_l\|_{\mathrm{R}} = \|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}}$ and given $\|\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ by initialization implementation in Condition B.3, we have

$$\lambda_{\boldsymbol{b}_l} = \begin{cases} \Theta(1/n_{\mathrm{out}}), & l = 0, \\ \Theta\left(1/(Ln_{\mathrm{out}})\right), & l \in [L]. \end{cases}$$

This completes the implementation of the update condition for SGD with weight decay, which is summarized in Table 5.

C.6 AdamW

In this section, we derive the μP implementation from Condition 3.1 for AdamW, which recovers and extends the μP scaling rules studied in Dey et al. (2025).

C.6.1 Update Rule

First, we present the full update rule of AdamW (Loshchilov & Hutter, 2019). To distinguish iteration steps, we append a superscript $t \in [T]$, which might be omitted later when no confusion arises.

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta_l^{(t)}\left(\mathrm{AdamW}\left(\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}\right) + \lambda_l \boldsymbol{W}_l^{(t)}\right),$$

Table 6: **μP implementation of Condition 3.1 ($k = 2$) for AdamW (Loshchilov & Hutter, 2019), Lion (Chen et al., 2023), and Sophia (Liu et al., 2024b) with weight decay under width–depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image. The initial variance of input bias is σ^2_{base}.

	Input weights & biases	Hidden weights	Output weights	Hidden biases
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})	α_{base}/r_L (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}	σ^2_{base}
Learning Rate	η_{base}	η_{base}/r_n (η_{base})	η_{base}	η_{base}
Weight Decay	λ_{base}	$\lambda_{\text{base}} r_n$ (λ_{base})	λ_{base}	λ_{base}
AdamW ε	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_L r_n)$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_L r_n)$ $(\varepsilon_{\text{base}})$

where

$$\begin{aligned}
&\text{AdamW}\left(\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}\right) = \frac{\hat{\boldsymbol{m}}_l^{(t)}}{\sqrt{\hat{\boldsymbol{v}}_l^{(t)}} + \varepsilon_l}, \\
&\text{where} \quad \begin{cases} \hat{\boldsymbol{m}}_l^{(t)} = \frac{\boldsymbol{m}_l^{(t)}}{1-\beta_1^t}, & \boldsymbol{m}_l^{(t)} = \beta_1 \boldsymbol{m}_l^{(t-1)} + (1-\beta_1)\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}, \\ \hat{\boldsymbol{v}}_l^{(t)} = \frac{\boldsymbol{v}_l^{(t)}}{1-\beta_2^t}, & \boldsymbol{v}_l^{(t)} = \beta_2 \boldsymbol{v}_l^{(t-1)} + (1-\beta_2)\left(\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}\right)^2. \end{cases}
\end{aligned} \tag{19}$$

We simplify the full update rule by omitting the momentum and the stabilization term, i.e., setting $\beta_1 = 0$, $\beta_2 = 0$, and $\varepsilon_l = 0$. As discussed at the beginning of the Appendix C, omitting momentum does not affect the scaling analysis. The stabilization term ε_l must, in fact, be scaled consistently with $\sqrt{\hat{\boldsymbol{v}}_l^{(t)}}$, and since it does not alter the resulting parameterization of learning rate, we defer its discussion to the end of this section. Now, the AdamW is reduced to sign gradient descent as:

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta_l^{(t)}\left(\text{sign}\left(\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}\right) + \lambda_l \boldsymbol{W}_l^{(t)}\right).$$

Here, the superscript of the iteration step can be left out, and we write this simplified update rule as:

$$\Delta \boldsymbol{W}_l = -\eta_l\big(\underbrace{\text{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l\big). \tag{20}$$

Given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the norm of $\boldsymbol{A}_l$ satisfies

$$\begin{aligned}
\|\boldsymbol{A}_l\|_{\text{R}} &= \|\text{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\,\|\text{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)\|_2 \\
&= \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \cdot \Theta\left(\|\text{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)\|_F\right) \qquad \text{(low-rank property of } \boldsymbol{A}_l \text{ in Equation (15))} \\
&= \Theta\left(\sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\sqrt{n_{\text{in}} n_{\text{out}}}\right) \\
&= \Theta(n_{\text{in}}) = \begin{cases} \Theta(1), & l = 0, \\ \Theta(n_{\text{in}}), & l \in [L], \\ \Theta(n_{\text{in}}), & l = L+1. \end{cases}
\end{aligned} \tag{21}$$

C.6.2 Derivation of Parameterization

Input and output layers. When weight decay is disabled ($\lambda_0 = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the multiplier parameterizations $\alpha_0 = \Theta(1), \alpha_{L+1} = \Theta(1/n_{\text{in}})$

in Equation (7), and the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ in Equation (21), we have

$$\alpha_l\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \alpha_l\eta_l\|\boldsymbol{A}_l\|_{\mathrm{R}} = \begin{cases} \Theta(\eta_0), & l = 0, \\ \Theta(\eta_{L+1}), & l = L+1. \end{cases}$$

As desired in (Δ1), to satisfy (C2.1) that $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\mathrm{R}}, \alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$, we need to set

$$\eta_0 = \Theta(1), \quad \eta_{L+1} = \Theta(1).$$

When $\lambda_l \neq 0$, given Equation (6) that $\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\mathrm{in}})$, we have

$$\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \begin{cases} \Theta(\lambda_0), & l = 0, \\ \Theta(\lambda_{L+1}n_{\mathrm{in}}), & l = L+1. \end{cases}$$

To satisfy (Δ2) that $\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$, we need to set

$$\lambda_0 = \Theta(1), \quad \lambda_{L+1} = \Theta(1).$$

Hidden layers (first-order). When weight decay is disabled ($\lambda_0 = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (8), and the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ in Equation (21), we have

$$\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)\cdot\eta_l^{(2)}\|\boldsymbol{A}_l^{(2)}\|_{\mathrm{R}}\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(\eta_l^{(2)}n_{\mathrm{in}}/L).$$

As desired in (Δ1), to satisfy the first-order update condition on hidden weights (C2.2) that $\alpha_l\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$, we need to set

$$\eta_l^{(2)} = \Theta(1/n_{\mathrm{in}}).$$

When $\lambda_l \neq 0$, given the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l^{(2)}\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(\lambda_l^{(2)}).$$

To satisfy (Δ2) that $\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$ we need to set

$$\lambda_l^{(2)} = \Theta(n_{\mathrm{in}}).$$

Symmetrically, we have the same choice for $\boldsymbol{W}_l^{(1)}$:

$$\eta_l^{(1)} = \Theta(1/n_{\mathrm{in}}), \quad \lambda_l^{(1)} = \Theta(n_{\mathrm{in}}).$$

Hidden layers (second-order). As illustrated in Section 3.3.3 or in Appendix C.4 for Muon, the second-order update condition is satisfied automatically once the initial condition and the first-order update condition are met.

Biases. For bias parameters $\boldsymbol{b}_l \in \mathbb{R}^{n_{\mathrm{out}}\times 1}$, by the definition we have

$$\|\mathrm{sign}\left(\nabla_{\boldsymbol{b}_l}\mathcal{L}\right)\|_{\mathrm{R}} = \Theta(1), \quad 0 \leq l \leq L.$$

To satisfy the condition $\|\Delta\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ for $\forall\ 0 \leq l \leq L$ in Equation (13), we set

$$\eta_{\boldsymbol{b}_l} = \Theta(1), \quad 0 \leq l \leq L,$$

and the corresponding weight decay

$$\lambda_{\boldsymbol{b}_l} = \Theta(1), \quad 0 \leq l \leq L.$$

Parameterization of ε_l. To make the stabilization term ε_l effective and not dominate the gradient, we desire it to be of the same scale as $\sqrt{\hat{\boldsymbol{v}}_l^{(t)}}$. When omitting the momentum, we have $\sqrt{\hat{\boldsymbol{v}}_l} = \nabla_{\boldsymbol{W}_l}\mathcal{L}$. Therefore, we need to ensure $\varepsilon_l = \Theta(\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_l)}\mathcal{L}\|_{\mathrm{R}})$, the latter can be estimated based on the derivation for $\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\mathrm{R}}$ in Equation (18) of Appendix C.5.

For the input weights $\boldsymbol{W}_0$, we have

$$\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_0)}\mathcal{L}\|_{\mathrm{R}} = \frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_{\mathrm{F}} = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{W}_0}\mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\sqrt{\frac{n_{\mathrm{in}}}{n_{\mathrm{out}}}}\right) = \Theta(\frac{1}{n_{\mathrm{out}}}).$$

Therefore, we set

$$\varepsilon_0 = \Theta(\frac{1}{n_{\mathrm{out}}}).$$

For the hidden weights $\boldsymbol{W}_l, l \in [L]$, we have

$$\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_l)}\mathcal{L}\|_{\mathrm{R}} = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\frac{1}{L}\sqrt{\frac{n_{\mathrm{in}}}{n_{\mathrm{out}}}}\right) = \Theta(\frac{1}{Ln_{\mathrm{out}}}).$$

Therefore, we set

$$\varepsilon_l = \Theta(\frac{1}{Ln_{\mathrm{out}}}), \quad l \in [L].$$

For the output weights $\boldsymbol{W}_{L+1}$, we have

$$\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_{L+1})}\mathcal{L}\|_{\mathrm{R}} = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}n_{\mathrm{out}}}}\sqrt{\frac{1}{n_{\mathrm{in}}n_{\mathrm{out}}}}\right) = \Theta(\frac{1}{n_{\mathrm{in}}}).$$

Therefore, we set

$$\varepsilon_{L+1} = \Theta(\frac{1}{n_{\mathrm{in}}}).$$

Similarly, for the biases we have derived in Appendix C.5 that

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}} = \begin{cases}\Theta(1/n_{\mathrm{out}}), & l = 0,\\ \Theta(1/(Ln_{\mathrm{out}})), & l \in [L].\end{cases}$$

Therefore, we set the stabilization term as

$$\varepsilon_{\boldsymbol{b}_l} = \begin{cases}\Theta(1/n_{\mathrm{out}}), & l = 0,\\ \Theta(1/(Ln_{\mathrm{out}})), & l \in [L].\end{cases}$$

This completes the implementation of the update condition for AdamW, which is summarized in Table 6.

C.7 Lion

In this section, we derive the μP implementation from Condition 3.1 for Lion.

The full update rule of Lion (Chen et al., 2023) is

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)}\left(\boldsymbol{u}_l^{(t)} + \lambda_l \boldsymbol{W}_l^{(t)}\right),$$

where

$$\boldsymbol{u}_l^{(t)} = \mathrm{sign}\left(\beta_1 \boldsymbol{m}_l^{(t-1)} + (1-\beta_1)\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}\right),$$
$$\boldsymbol{m}_l^{(t)} = \beta_2 \boldsymbol{m}_l^{(t-1)} + (1-\beta_2)\nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}.$$

If the momentum terms are omitted, i.e., setting $\beta_1 = 0$ and $\beta_2 = 0$, Lion is reduced to sign gradient descent as in the simplified AdamW update rule in Equation (20). Therefore, we reuse AdamW's parameterizations in Table 6 for Lion.

C.8 Sophia

In this section, we derive the μP implementation from Condition 3.1 for Sophia.

The full update rule of Sophia (Liu et al., 2024b) is

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\text{clip}\left(\frac{\boldsymbol{m}_l^{(t)}}{\max\{\rho \boldsymbol{h}_l^{(t)}, \varepsilon\}}, 1 \right) + \lambda_l \boldsymbol{W}_l^{(t)} \right),$$

where

$$\boldsymbol{m}_l^{(t)} = \beta_1 \boldsymbol{m}_l^{(t-1)} + (1-\beta_1)\boldsymbol{G}_l^{(t)},$$

and $\boldsymbol{h}_l^{(t)}$ is updated every k iterations as

$$\boldsymbol{h}_l^{(t)} = \begin{cases} \beta_2 \boldsymbol{h}_l^{(t-1)} + (1-\beta_2)\hat{\boldsymbol{h}}_l^{(t)}, & t \bmod k = 1, \\ \boldsymbol{h}_l^{(t-1)}, & t \bmod k \neq 1. \end{cases}$$

where the elements of $\hat{\boldsymbol{h}}_l^{(t)}$ are the diagonal second-order derivatives with respect to $\boldsymbol{W}_l^{(t)}$, i.e., $\left(\hat{\boldsymbol{h}}_l^{(t)}\right)_{ij} = \frac{\partial^2 \mathcal{L}}{\partial (\boldsymbol{W}_l^{(t)})_{ij}^2}$.

Letting $\boldsymbol{A}_l^{(t)} = \text{clip}\left(\frac{\boldsymbol{m}_l^{(t)}}{\max\{\rho \boldsymbol{h}_l^{(t)}, \varepsilon\}}, 1 \right)$, we use the following upper bound to estimate its norm, following Ngom et al. (2025), which derives a parameterization for Sophia under width scaling:

$$\|\boldsymbol{A}_l^{(t)}\|_{\text{R}} \leq \|\mathbf{1}_{n_{\text{out}} \times n_{\text{in}}}\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\mathbf{1}_{n_{\text{out}} \times n_{\text{in}}}\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \sqrt{n_{\text{in}} n_{\text{out}}} = n_{\text{in}}.$$

The resulting width-scaling parameterization is empirically validated to be effective in Ngom et al. (2025). Since this bound has the same order as the AdamW estimate in Equation (21), Sophia shares AdamW's parameterizations in Table 6.

C.9 Shampoo

In this section, we derive the μP implementation from Condition 3.1 for Shampoo, which recovers and extends the μP scaling rules studied in Qiu et al. (2025).

C.9.1 Update Rule

Denote $\boldsymbol{G}_l^{(t)} = \nabla_{\boldsymbol{W}_l^{(t)}} \mathcal{L}$. Following Shampoo (Gupta et al., 2018), the gradient is preconditioned by Kronecker-factored left and right preconditioners:

$$\begin{aligned}
\boldsymbol{M}_l^{(t)} &= \beta_1 \boldsymbol{M}_l^{(t-1)} + (1-\beta_1)\boldsymbol{G}_l^{(t)}, \\
\boldsymbol{L}_l^{(t)} &= \beta_2 \boldsymbol{L}_l^{(t-1)} + (1-\beta_2)\boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top}, \\
\boldsymbol{R}_l^{(t)} &= \beta_2 \boldsymbol{R}_l^{(t-1)} + (1-\beta_2)\boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)}, \\
\widehat{\boldsymbol{L}}_l^{(t)} &= \frac{\boldsymbol{L}_l^{(t)}}{1-\beta_2^t}, \qquad \widehat{\boldsymbol{R}}_l^{(t)} = \frac{\boldsymbol{R}_l^{(t)}}{1-\beta_2^t}, \\
\boldsymbol{W}_l^{(t)} &= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\left(\widehat{\boldsymbol{L}}_l^{(t)} + \varepsilon_l \boldsymbol{I} \right)^{-\frac{1}{4}} \boldsymbol{M}_l^{(t)} \left(\widehat{\boldsymbol{R}}_l^{(t)} + \varepsilon_l \boldsymbol{I} \right)^{-\frac{1}{4}} + \lambda_l \boldsymbol{W}_l^{(t)} \right),
\end{aligned} \tag{22}$$

We simplify the full update rule by omitting the momentum and the stabilization term, i.e., setting $\beta_1 = 0$, $\beta_2 = 0$, and $\varepsilon_l = 0$. As discussed at the beginning of the Appendix C, omitting momentum does not affect

the scaling analysis. The stabilization term ε_l must, in fact, be scaled consistently with $\widehat{\boldsymbol{L}}_l^{(t)}$ and $\widehat{\boldsymbol{R}}_l^{(t)}$, and since it does not alter the resulting parameterization of learning rate, we defer its discussion to the end of this section.

Now, we have $\widehat{\boldsymbol{L}}_l^{(t)} = \boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top}$, and $\widehat{\boldsymbol{R}}_l^{(t)} = \boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)}$. Therefore, we obtain

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\left(\boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top} \right)^{-\frac{1}{4}} \boldsymbol{G}_l^{(t)} \left(\boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)} \right)^{-\frac{1}{4}} + \lambda_l \boldsymbol{W}_l^{(t)} \right).$$

C.9.2 Derivation of Parameterization

Learning rate and weight decay. Applying compact SVD to $\boldsymbol{G}_l^{(t)}$ as in Muon, and interpreting the inverse powers on the support of the gradient, we have

$$\boldsymbol{G}_l^{(t)} = \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} \boldsymbol{V}_l^{(t)\top},$$

and thus

$$\boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top} = \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)2} \boldsymbol{U}_l^{(t)\top}, \quad \boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)} = \boldsymbol{V}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)2} \boldsymbol{V}_l^{(t)\top}.$$

The preconditioned direction is therefore

$$\begin{aligned} &\left(\boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top} \right)^{-\frac{1}{4}} \boldsymbol{G}_l^{(t)} \left(\boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)} \right)^{-\frac{1}{4}} \\ &= \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)-\frac{1}{2}} \boldsymbol{U}_l^{(t)\top} \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} \boldsymbol{V}_l^{(t)\top} \boldsymbol{V}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)-\frac{1}{2}} \boldsymbol{V}_l^{(t)\top} = \boldsymbol{U}_l^{(t)} \boldsymbol{V}_l^{(t)\top}. \end{aligned}$$

Consequently, under this simplification, Shampoo reduces to

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{U}_l^{(t)} \boldsymbol{V}_l^{(t)\top} + \lambda_l \boldsymbol{W}_l^{(t)} \right),$$

which matches the update rule of Muon in Equation (16). Therefore, the learning rate and weight decay of Shampoo share Muon's parameterizations in Table 4.

Note that the hidden layer learning rate parameterization derived in Qiu et al. (2025, Table 1) is $\frac{(n_{\text{out}}/n_{\text{in}})^{1-(e_L+e_R)}}{L^{2(e_L+e_R)-1} n_{\text{blk}}^{e_L+e_R}}$, where the e_L and e_R are the exponents of $\boldsymbol{L}_l^{(t)}$ and $\boldsymbol{R}_l^{(t)}$ which equal $\frac{1}{4}$ in the standard Shampoo as Equation (22), and n_{blk} is the number of blocks which equals 1 when blocking is not used. In this case, $\frac{(n_{\text{out}}/n_{\text{in}})^{1-(e_L+e_R)}}{L^{2(e_L+e_R)-1} n_{\text{blk}}^{e_L+e_R}} = \sqrt{n_{\text{out}}/n_{\text{in}}} = \Theta(1)$, consistent with our result for Shampoo in Table 4.

Parameterization of ε_l. To make the stabilization term ε_l effective and not dominate the update, we desire it to be of the same scale as the single value of $\widehat{\boldsymbol{L}}_l^{(t)}$ and $\widehat{\boldsymbol{R}}_l^{(t)}$. When omitting the momentum, we have

$$\|\widehat{\boldsymbol{L}}_l^{(t)}\|_2 = \|\widehat{\boldsymbol{R}}_l^{(t)}\|_2 = \|\boldsymbol{G}_l^{(t)}\|_2^2 = \|\nabla_{\boldsymbol{W}_l^{(t)}} \mathcal{L}\|_2^2 = \begin{cases} \Theta(\frac{n_{\text{in}}}{n_{\text{out}}}) = \Theta(\frac{1}{n_{\text{out}}}), & l = 0, \\ \Theta(\frac{1}{L^2} \frac{n_{\text{in}}}{n_{\text{out}}}) = \Theta(\frac{1}{L^2}), & l \in [L], \\ \Theta(\frac{1}{n_{\text{in}} n_{\text{out}}}) = \Theta(\frac{1}{n_{\text{in}}}), & l = L+1, \end{cases}$$

where the estimation of $\|\nabla_{\boldsymbol{W}_l^{(t)}} \mathcal{L}\|_2$ can be found in the derivation for $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_{\text{R}}$ in Equation (18) at Appendix C.5. Therefore, we need to set the stabilization term as

$$\varepsilon_l = \begin{cases} \Theta(\frac{1}{n_{\text{out}}}), & l = 0, \\ \Theta(\frac{1}{L^2}), & l \in [L], \\ \Theta(\frac{1}{n_{\text{in}}}), & l = L+1. \end{cases}$$

Note that the hidden layer ε_l parameterization derived in Qiu et al. (2025, Table 1) is $\frac{n_{\text{in}}}{L^2 n_{\text{out}} n_{\text{blk}}}$, where the n_{blk} is the number of blocks which equals 1 when blocking is not used. In this case, $\frac{n_{\text{in}}}{L^2 n_{\text{out}} n_{\text{blk}}} = \frac{n_{\text{in}}}{L^2 n_{\text{out}}} = \Theta(\frac{1}{L^2})$, consistent with our result for Shampoo in Table 4.

C.10 SOAP

In this section, we derive the μP implementation from Condition 3.1 for SOAP, which recovers and extends the μP scaling rules studied in Qiu et al. (2025).

Denote the weight gradient as $\boldsymbol{G}_l^{(t)} = \nabla_{\boldsymbol{W}_l^{(t)}} \mathcal{L} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$ and its rank $r = \text{rank}(\boldsymbol{G}_l^{(t)})$. SOAP adopts a similar precondition of Shampoo's for $\boldsymbol{L}_l^{(t)}$ and $\boldsymbol{R}_l^{(t)}$:

$$\boldsymbol{L}_l^{(t)} = \beta_3 \boldsymbol{L}_l^{(t-1)} + (1-\beta_3)\boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top}, \quad \boldsymbol{R}_l^{(t)} = \beta_3 \boldsymbol{R}_l^{(t-1)} + (1-\beta_3)\boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)}.$$

By applying eigendecomposition to matrices $\boldsymbol{L}_l^{(t)} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{out}}}$ and $\boldsymbol{R}_l^{(t)} \in \mathbb{R}^{n_{\text{in}} \times n_{\text{in}}}$, we get two orthogonal matrices $\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{out}}}$ and $\boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)} \in \mathbb{R}^{n_{\text{in}} \times n_{\text{in}}}$ as:

$$\boldsymbol{L}_l^{(t)} = \boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \boldsymbol{\Lambda}_{\boldsymbol{L}_l}^{(t)} \boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)\top}, \quad \boldsymbol{R}_l^{(t)} = \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)} \boldsymbol{\Lambda}_{\boldsymbol{R}_l}^{(t)} \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)\top}. \tag{23}$$

This induces a rotated gradient:

$$\boldsymbol{G}_l'^{(t)} = \boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)\top} \boldsymbol{G}_l^{(t)} \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)}.$$

The full update rule of SOAP is

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \text{AdamW}\left(\boldsymbol{G}_l'^{(t)} \right) \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)\top} + \lambda_l \boldsymbol{W}_l^{(t)} \right),$$

where $\text{AdamW}(\cdot)$ is defined as in Equation (19).

First, omit the momentum and the stabilization term in the AdamW operator. Then, $\text{AdamW}(\cdot)$ is reduced to $\text{sign}(\cdot)$ as discussed in Appendix C.6.1.

Then, omit the momentum term in $\boldsymbol{L}_l^{(t)}$ and $\boldsymbol{R}_l^{(t)}$, i.e., set $\beta_3 = 0$. Applying compact SVD to the weight matrix $\boldsymbol{G}_l^{(t)} = \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} \boldsymbol{V}_l^{(t)\top}$, where $\boldsymbol{U}_l^{(t)} \in \mathbb{R}^{n_{\text{out}} \times r}$, $\boldsymbol{\Sigma}_l^{(t)} \in \mathbb{R}^{r \times r}$, and $\boldsymbol{V}_l^{(t)} \in \mathbb{R}^{n_{\text{in}} \times r}$, we have

$$\boldsymbol{L}_l^{(t)} = \boldsymbol{G}_l^{(t)} \boldsymbol{G}_l^{(t)\top} = \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)2} \boldsymbol{U}_l^{(t)\top}, \quad \boldsymbol{R}_l^{(t)} = \boldsymbol{G}_l^{(t)\top} \boldsymbol{G}_l^{(t)} = \boldsymbol{V}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)2} \boldsymbol{V}_l^{(t)\top}.$$

According to Equation (23), the eigenvectors corresponding to the non-zero eigenvalues match the singular vectors, i.e.,

$$\boldsymbol{Q}_{\boldsymbol{L}_l[:,:r]}^{(t)} = \boldsymbol{U}_l^{(t)}, \quad \boldsymbol{Q}_{\boldsymbol{R}_l[:,:r]}^{(t)} = \boldsymbol{V}_l^{(t)}.$$

We can partition the orthogonal matrices as $\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} = [\boldsymbol{U}_l^{(t)} \ \boldsymbol{U}_\perp^{(t)}]$ and $\boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)} = [\boldsymbol{V}_l^{(t)} \ \boldsymbol{V}_\perp^{(t)}]$. Substituting the eigendecomposition of $\boldsymbol{G}_l^{(t)}$, the rotated gradient becomes:

$$\boldsymbol{G}_l'^{(t)} = \boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)\top} \boldsymbol{G}_l^{(t)} \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)} = \begin{bmatrix} \boldsymbol{U}_l^{(t)\top} \\ \boldsymbol{U}_\perp^{(t)\top} \end{bmatrix} \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} \boldsymbol{V}_l^{(t)\top} \begin{bmatrix} \boldsymbol{V}_l^{(t)} & \boldsymbol{V}_\perp^{(t)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_l^{(t)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

Then, we find that SOAP is reduced to:

$$\begin{aligned}
\boldsymbol{W}_l^{(t)} &= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \text{sign}\left(\begin{bmatrix} \boldsymbol{\Sigma}_l^{(t)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \right) \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)\top} + \lambda_l \boldsymbol{W}_l^{(t)} \right) \\
&= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \begin{bmatrix} \boldsymbol{I}_{r\times r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)\top} + \lambda_l \boldsymbol{W}_l^{(t)} \right) \\
&= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{U}_l^{(t)} \boldsymbol{V}_l^{(t)\top} + \lambda_l \boldsymbol{W}_l^{(t)} \right),
\end{aligned}$$

Table 7: **μP implementation of Condition 3.1 ($k = 2$) for SSO (Xie et al., 2026) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	α_{base}/r_L (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	η_{base}	η_{base}	$\eta_{\text{base}} r_n$ (η_{base})
Weight Decay	λ_{base}	λ_{base}	$\lambda_{\text{base}}/r_n$ (λ_{base})

which, again, matches exactly the update rule of Muon in Equation (16). Therefore, SOAP shares Muon's parameterizations in Table 4.

Note that the hidden layer learning rate parameterization derived in Qiu et al. (2025, Table 1) is $\frac{n_{\text{out}}^{e_L/2} n_{\text{in}}^{e_R/2}}{n_{\text{in}}}$, where the e_L and e_R are the indicators for left- and right-side preconditioners, which equals 1 for standard SOAP. In this case, $\frac{n_{\text{out}}^{e_L/2} n_{\text{in}}^{e_R/2}}{n_{\text{in}}} = \frac{\sqrt{n_{\text{out}} n_{\text{in}}}}{n_{\text{in}}} = \Theta(1)$, consistent with our result for SOAP in Table 4.

C.11 Spectral Sphere Optimizer (SSO)

In this section, we derive the μP implementation from Condition 3.1 for SSO.

C.11.1 Update Rule

SSO (Xie et al., 2026) aims to perform steepest descent on the spectral sphere (see Section 3.1 in the original paper), where the update follows:

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{R \boldsymbol{\Phi}_l}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big),$$

with

$$R = \Theta\left(\sqrt{\frac{n_{\text{out}}}{n_{\text{in}}}}\right), \quad \text{and} \quad \boldsymbol{\Phi}_l = \arg\max_{\boldsymbol{\Phi}} \langle \nabla_{\boldsymbol{W}_l} \mathcal{L}, \boldsymbol{\Phi} \rangle \text{ s.t. } \|\boldsymbol{\Phi}\|_2 = 1,\ \|\boldsymbol{W}_l - \eta_l \boldsymbol{\Phi}\|_2 = \|\boldsymbol{W}_l\|_2 = R.$$

Thus we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\boldsymbol{A}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} R \|\boldsymbol{\Phi}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \Theta\left(\sqrt{\frac{n_{\text{out}}}{n_{\text{in}}}}\right) = 1. \tag{24}$$

C.11.2 Derivation of Parameterization

Input and output layers. When weight decay is disabled ($\lambda_0 = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the multiplier parameterizations $\alpha_0 = \Theta(1), \alpha_{L+1} = \Theta(1/n_{\text{in}})$ in Equation (7), and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ in Equation (24), we have

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \alpha_l \eta_l \|\boldsymbol{A}_l\|_{\text{R}} = \begin{cases} \Theta(\eta_0), & l = 0, \\ \Theta(\eta_{L+1}/n_{\text{in}}), & l = L+1. \end{cases}$$

As desired in (Δ1), to satisfy (C2.1) that $\alpha_0 \|\Delta \boldsymbol{W}_0\|_{\text{R}}, \alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(1)$, we need to set

$$\eta_0 = \Theta(1), \quad \eta_{L+1} = \Theta(n_{\text{in}}).$$

When $\lambda_l \neq 0$, given Equation (6) that $\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\mathrm{in}})$, we have

$$\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \begin{cases} \Theta(\lambda_0), & l = 0, \\ \Theta(\lambda_{L+1} n_{\mathrm{in}}), & l = L+1. \end{cases}$$

To satisfy ($\Delta 2$) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$, we need to set

$$\lambda_0 = \Theta(1), \quad \lambda_{L+1} = \Theta(1/n_{\mathrm{in}}),$$

Hidden layers (first-order). When weight decay is disabled ($\lambda_l = 0$), given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (8), and the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ in Equation (24), we have

$$\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L) \cdot \eta_l^{(2)} \|\boldsymbol{A}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(\eta_l^{(2)}/L).$$

As desired in ($\Delta 1$), to satisfy the first-order update condition on hidden weights (C2.2) that $\alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \, \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \Theta(1/L)$, we need to set

$$\eta_l^{(2)} = \Theta(1).$$

When weight decay is enabled ($\lambda_l \neq 0$), given the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l^{(2)} \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(\lambda_l^{(2)}).$$

To satisfy ($\Delta 2$) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$ we need to set

$$\lambda_l^{(2)} = \Theta(1).$$

Symmetrically, we have the same choice for $\boldsymbol{W}_l^{(1)}$:

$$\eta_l^{(1)} = \Theta(1), \quad \lambda_l^{(1)} = \Theta(1).$$

Hidden layers (second-order). As illustrated in Section 3.3.3 or in Appendix C.4 for Muon, the second-order update condition is satisfied automatically once the initial condition and the first-order update condition are met.

This completes the implementation of the update condition for SSO with weight decay, which is summarized in Table 7.

D Implementing Condition B.1 for Optimizers with Weight Decay

We implement Condition B.1 using the same width-scaling μP initialization convention as in Section 4.1. Under this convention, hidden matrix weights satisfy $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ at initialization, so the hidden initial condition in Condition B.1 permits $\alpha_l = \mathcal{O}(1/\sqrt{L})$. In this section, we focus on the Depth-μP-style choice $\alpha_l = \Theta(1/\sqrt{L})$, which corresponds to maximizing the zero-order feature-update contribution discussed in Appendix B.1. The input and output layer multipliers remain the same as in Section 4.1, that $\alpha_0 = \Theta(1)$ and $\alpha_{L+1} = \Theta(1/n_{\mathrm{in}})$. We now start update-condition analysis for optimizers with weight decay.

To provide a unified derivation across different optimizers, we begin by expressing their update rules in a general form. When weight decay is included, a single update step of the weight matrix can be written as

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\boldsymbol{A}_l + \lambda_l \boldsymbol{W}_l\big),$$

where $\boldsymbol{A}_l$ denotes an optimizer-specific update for $\boldsymbol{W}_l$, and λ_l is the weight decay coefficient.

The update magnitude $\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \eta_l \|\boldsymbol{A}_l + \lambda_l \boldsymbol{W}_l\|_{\mathrm{R}}$ is required to satisfy the update conditions in Condition B.1. We analyze this requirement under two complementary regimes.

Table 8: **Overview of μP implementation from Condition B.1 ($k = 1$) for optimizer families with weight decay.** Optimizers in the same row share the same μP scaling rules.

Optimizer family	Detailed parameterization
Muon-Kimi	Table 9
Muon / Shampoo / SOAP	Table 10
SGD	Table 11
AdamW / Lion / Sophia	Table 12
SSO	Table 13

Without weight decay. When weight decay is disabled ($\lambda_l = 0$), the update reduces to $\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \eta_l \|\boldsymbol{A}_l\|_{\mathrm{R}}$. In this case, we require:

$$\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \eta_l \|\boldsymbol{A}_l\|_{\mathrm{R}} \text{ satisfies Condition B.1.} \tag{Υ1}$$

With weight decay. When weight decay is enabled ($\lambda_l \neq 0$), we choose the weight decay term to be comparable in scale to the optimizer-driven term:

$$\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right). \tag{Υ2}$$

As discussed in Appendix C, this is a non-degenerate implementation convention: it keeps weight decay active in the update dynamics without letting it dominate the optimizer-driven term. Under the usual scale estimate, $\|\boldsymbol{A}_l + \lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(\|\boldsymbol{A}_l\|_{\mathrm{R}})$, so the learning-rate scaling rule derived from Equation (Υ1) is preserved.

Weights and biases. In the following, we derive parameterizations of the learning rate and weight decay coefficient for a range of optimizers. As in Appendix C, matrix-based optimizers such as Muon and Shampoo are typically not applied to bias parameters, so we restrict the analysis of bias parameterization to vector-based optimizers such as SGD and AdamW. The same two rules, Equations (Υ1) and (Υ2), are then applied to the biases with the corresponding one-layer specialization of spectral Condition B.3.

Momentum. We use the same momentum simplification as in Appendix C: the derivations omit momentum, while practical implementations take momentum coefficients to be $\Theta(1)$. This can also be interpreted as analyzing the first update step after initialization.

D.1 Overview

Table 8 summarizes the optimizer families covered in this section and points to the corresponding detailed Depth-μP-style parameterization tables. All rows use the initialization convention above, with hidden residual multiplier $\alpha_l = \Theta(1/\sqrt{L})$; the table only organizes the optimizer-dependent update rules derived below.

D.2 Muon-Kimi

In this section, we derive the μP implementation from Condition B.1 for Muon-Kimi.

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\mathrm{out}} \times n_{\mathrm{in}}}$, the update rule of Muon-Kimi (Liu et al., 2025) with weight decay is

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{0.2\sqrt{\max\{n_{\mathrm{in}}, n_{\mathrm{out}}\}}\, \boldsymbol{U}_l \boldsymbol{V}_l^\top}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big),$$

Table 9: **μP implementation of Condition B.1 ($k = 1$) for Muon-Kimi (Liu et al., 2025) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	η_{base}	$\eta_{\text{base}}/\sqrt{r_n r_L}$ (η_{base})	η_{base}
Weight Decay	λ_{base}	$\lambda_{\text{base}}\sqrt{r_n}$ (λ_{base})	λ_{base}

where $\boldsymbol{U}_l, \boldsymbol{V}_l$ arise from the compact SVD of the gradient $\nabla_{\boldsymbol{W}_l}\mathcal{L} = \boldsymbol{U}_l\boldsymbol{\Sigma}_l\boldsymbol{V}_l^\top$. Using the norm computation in Equation (9), we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \Theta\left(\sqrt{n_{\text{in}}}\max\left\{1, \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\right\}\right) = \begin{cases} \Theta(1), & l = 0, \\ \Theta(\sqrt{n_{\text{in}}}), & l \in [L], \\ \Theta(n_{\text{in}}), & l = L+1. \end{cases}$$

D.2.1 Derivation of Parameterization

Input and output layers. The input and output layer conditions in Condition B.1 are identical to those in Condition 3.1. Therefore, as in Appendix C.3, their Muon-Kimi learning-rate and weight-decay parameterizations are

$$\eta_0 = \Theta\left(1\right), \quad \eta_{L+1} = \Theta\left(1\right), \quad \lambda_0 = \Theta(1), \quad \lambda_{L+1} = \Theta(1).$$

Hidden layers (first-order). The only change from the $k = 2$ Muon-Kimi implementation is the hidden multiplier: Condition B.1 uses the Depth-μP-style choice $\alpha_l = \Theta(1/\sqrt{L})$ rather than $\Theta(1/L)$. When $\lambda_l = 0$, given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3) and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ above, we have

$$\alpha_l\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \Theta(1/\sqrt{L}) \cdot \eta_l\|\boldsymbol{A}_l\|_{\text{R}} = \Theta(\eta_l\sqrt{n_{\text{in}}}/\sqrt{L}).$$

As desired in Equation (Υ1), to satisfy the first-order update condition on hidden weights, $\alpha_l\|\Delta\boldsymbol{W}_l\|_{\text{R}} = \Theta(1/L)$, we need to set

$$\eta_l = \Theta(\frac{1}{\sqrt{n_{\text{in}}L}}).$$

When $\lambda_l \neq 0$, the weight-decay parameterization is unchanged from the $k = 2$ Muon-Kimi implementation because it matches the scale of $\boldsymbol{A}_l$ and does not depend on the residual multiplier. Given the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l\boldsymbol{W}_l\|_{\text{R}} = \Theta(\lambda_l).$$

To satisfy Equation (Υ2) that $\|\lambda_l\boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right)$, we need to set

$$\lambda_l = \Theta(\sqrt{n_{\text{in}}}).$$

This completes the implementation of Condition B.1 for Muon-Kimi with weight decay, as summarized in Table 9.

Table 10: **μP implementation of Condition B.1 ($k = 1$) for Muon (Jordan et al., 2024b), Shampoo (Gupta et al., 2018), and SOAP (Vyas et al., 2024) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	$\eta_{\text{base}}\sqrt{r_n}$ (η_{base})	$\eta_{\text{base}}/\sqrt{r_L}$ (η_{base})	$\eta_{\text{base}}\sqrt{r_n}$ (η_{base})
Weight Decay	$\lambda_{\text{base}}/\sqrt{r_n}$ (λ_{base})	λ_{base}	$\lambda_{\text{base}}/\sqrt{r_n}$ (λ_{base})
Shampoo ε	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_L$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$

D.3 Muon, Shampoo and SOAP

In this section, we derive the μP implementation from Condition B.1 for Muon, Shampoo, and SOAP.

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the update rule of Muon (Jordan et al., 2024b) with weight decay is

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{\boldsymbol{U}_l \boldsymbol{V}_l^\top}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big),$$

where $\boldsymbol{U}_l, \boldsymbol{V}_l$ arise from the compact SVD of the gradient $\nabla_{\boldsymbol{W}_l} \mathcal{L} = \boldsymbol{U}_l \boldsymbol{\Sigma}_l \boldsymbol{V}_l^\top$. Using the norm computation in Equation (17), we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} = \begin{cases} \Theta(1/\sqrt{n_{\text{out}}}), & l = 0, \\ \Theta(1), & l \in [L], \\ \Theta(\sqrt{n_{\text{in}}}), & l = L+1. \end{cases}$$

As discussed in Appendices C.9 and C.10, Shampoo and SOAP reduce to the same Muon update direction under the simplifications used in this section, so they share the same parameterization.

D.3.1 Derivation of Parameterization

Input and output layers. The input and output layer conditions in Condition B.1 are identical to those in Condition 3.1. Therefore, as in Appendix C.4, their Muon learning-rate and weight-decay parameterizations are

$$\eta_0 = \Theta(\sqrt{n_{\text{out}}}), \quad \eta_{L+1} = \Theta(\sqrt{n_{\text{in}}}), \quad \lambda_0 = \Theta(1/\sqrt{n_{\text{out}}}), \quad \lambda_{L+1} = \Theta(1/\sqrt{n_{\text{in}}}).$$

Hidden layers (first-order). The only change from the $k = 2$ Muon implementation is the hidden multiplier: Condition B.1 uses the Depth-μP-style choice $\alpha_l = \Theta(1/\sqrt{L})$ rather than $\Theta(1/L)$. When $\lambda_l = 0$, given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3) and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ above, we have

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/\sqrt{L}) \cdot \eta_l \|\boldsymbol{A}_l\|_{\text{R}} = \Theta(\eta_l/\sqrt{L}).$$

As desired in Equation (Υ1), to satisfy the first-order update condition on hidden weights, $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/L)$, we need to set

$$\eta_l = \Theta(1/\sqrt{L}).$$

Table 11: **μP implementation of Condition B.1 ($k = 1$) for SGD with weight decay under width–depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image. The initial variance of input bias is σ^2_{base}.

	Input weights & biases	Hidden weights	Output weights	Hidden biases
Block Multiplier	α_{base}	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})	α_{base}/r_n (α_{base})	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}	σ^2_{base}
Learning Rate	$\eta_{\text{base}} r_n$ (η_{base})	η_{base}	$\eta_{\text{base}} r_n$ (η_{base})	$\eta_{\text{base}} r_n$ (η_{base})
Weight Decay	$\lambda_{\text{base}}/r_n$ (λ_{base})	$\lambda_{\text{base}}/\sqrt{r_L}$ (λ_{base})	$\lambda_{\text{base}}/r_n$ (λ_{base})	$\lambda_{\text{base}}/(r_n\sqrt{r_L})$ (λ_{base})

When $\lambda_l \neq 0$, the weight-decay parameterization is unchanged from the $k = 2$ Muon implementation because it matches the scale of $\boldsymbol{A}_l$ and does not depend on the residual multiplier. Given the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta(\lambda_l).$$

To satisfy Equation (Υ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right)$, we need to set

$$\lambda_l = \Theta(1).$$

Parameterization of Shampoo ε_l. As in Appendix C.9, Shampoo's stabilization term is added to the left and right preconditioners $\widehat{\boldsymbol{L}}_l^{(t)}$ and $\widehat{\boldsymbol{R}}_l^{(t)}$, so it should match the scale of these preconditioners rather than the elementwise gradient scale. When omitting the momentum, this gives

$$\|\widehat{\boldsymbol{L}}_l^{(t)}\|_2 = \|\widehat{\boldsymbol{R}}_l^{(t)}\|_2 = \|\boldsymbol{G}_l^{(t)}\|_2^2 = \|\nabla_{\boldsymbol{W}_l^{(t)}} \mathcal{L}\|_2^2.$$

The input and output gradient estimates are unchanged from Appendix C.9. For hidden weights, the one-layer condition uses $\alpha_l = \Theta(1/\sqrt{L})$, and the raw-gradient estimate in Equation (25) gives $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_2 = \Theta(\sqrt{n_{\text{in}}/(n_{\text{out}} L)})$. Therefore,

$$\|\widehat{\boldsymbol{L}}_l^{(t)}\|_2 = \|\widehat{\boldsymbol{R}}_l^{(t)}\|_2 = \begin{cases} \Theta(\frac{n_{\text{in}}}{n_{\text{out}}}) = \Theta(\frac{1}{n_{\text{out}}}), & l = 0, \\ \Theta(\frac{1}{L}\frac{n_{\text{in}}}{n_{\text{out}}}) = \Theta(\frac{1}{L}), & l \in [L], \\ \Theta(\frac{1}{n_{\text{in}} n_{\text{out}}}) = \Theta(\frac{1}{n_{\text{in}}}), & l = L+1. \end{cases}$$

Thus, the stabilization term should be parameterized as

$$\varepsilon_l = \begin{cases} \Theta(\frac{1}{n_{\text{out}}}), & l = 0, \\ \Theta(\frac{1}{L}), & l \in [L], \\ \Theta(\frac{1}{n_{\text{in}}}), & l = L+1. \end{cases}$$

This corresponds to the Shampoo ε row in Table 10.

This completes the implementation of Condition B.1 for Muon, Shampoo, and SOAP with weight decay, as summarized in Table 10.

D.4 SGD

In this section, we derive the μP implementation from Condition B.1 for SGD, which recovers and extends the μP scaling rules studied in Yang et al. (2024); Bordelon et al. (2024b).

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, the SGD update rule with weight decay can be written as:

$$\Delta \boldsymbol{W}_l = -\eta_l \big(\underbrace{\nabla_{\boldsymbol{W}_l} \mathcal{L}}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l \big).$$

For SGD, $\boldsymbol{A}_l$ is the raw gradient, so the scale of $\boldsymbol{A}_l$ depends on the residual multiplier used in the spectral update condition. The input and output layer estimates are the same as in Appendix C.5: $\|\boldsymbol{A}_0\|_{\text{R}} = \Theta(n_{\text{in}}/n_{\text{out}}) = \Theta(1/n_{\text{out}})$ and $\|\boldsymbol{A}_{L+1}\|_{\text{R}} = \Theta(1/n_{\text{out}}) = \Theta(1)$. The hidden-layer estimate changes because Condition B.1 uses $\alpha_l = \Theta(1/\sqrt{L})$.

For a hidden weight $\boldsymbol{W}_l$, we have

$$\Theta(\frac{1}{L}) = \Delta_{\boldsymbol{W}_l} \mathcal{L} = \Theta(\langle \Delta \boldsymbol{W}_l, \nabla_{\boldsymbol{W}_l} \mathcal{L} \rangle) = \Theta(\|\Delta \boldsymbol{W}_l\|_{\text{F}} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_{\text{F}}) = \Theta(\|\Delta \boldsymbol{W}_l\|_2 \|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_2).$$

Since we set $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/L)$ to satisfy the hidden update condition in Condition B.1, and use $\alpha_l = \Theta(1/\sqrt{L})$, we have $\|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/\sqrt{L})$ and thus $\|\Delta \boldsymbol{W}_l\|_2 = \Theta(\sqrt{n_{\text{out}}/(n_{\text{in}}L)})$. Therefore, $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_2 = \Theta(\sqrt{n_{\text{in}}/(n_{\text{out}}L)})$, which leads to

$$\|\boldsymbol{A}_l\|_{\text{R}} = \|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_2 = \Theta(\frac{n_{\text{in}}}{n_{\text{out}}\sqrt{L}}) = \Theta(\frac{1}{\sqrt{L}}).$$

To sum up, we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_{\text{R}} = \begin{cases} \Theta(1/n_{\text{out}}), & l = 0, \\ \Theta(1/\sqrt{L}), & l \in [\,L\,], \\ \Theta(1), & l = L+1. \end{cases} \tag{25}$$

D.4.1 Derivation of Parameterization

Input and output layers. The input and output layer conditions in Condition B.1 are identical to those in Condition 3.1. Therefore, as in Appendix C.5, their SGD learning-rate and weight-decay parameterizations are

$$\eta_0 = \Theta(n_{\text{out}}), \quad \eta_{L+1} = \Theta(n_{\text{in}}), \quad \lambda_0 = \Theta(1/n_{\text{out}}), \quad \lambda_{L+1} = \Theta(1/n_{\text{in}}).$$

Hidden layers (first-order). Unlike normalized or preconditioned optimizers, SGD uses the raw gradient, so changing the hidden multiplier also changes the hidden raw-gradient scale. When $\lambda_l = 0$, using $\alpha_l = \Theta(1/\sqrt{L})$ and $\|\boldsymbol{A}_l\|_{\text{R}} = \Theta(1/\sqrt{L})$, we obtain

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/\sqrt{L}) \cdot \eta_l \|\boldsymbol{A}_l\|_{\text{R}} = \Theta(\eta_l/L).$$

Enforcing the hidden update condition in Condition B.1 that $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\text{R}} = \Theta(1/L)$ gives

$$\eta_l = \Theta(1).$$

If weight decay is enabled on hidden matrices, using $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ by Equation (6), we obtain $\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta(\lambda_l)$, so Equation ($\Upsilon$2) implies $\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta(\|\boldsymbol{A}_l\|_{\text{R}}) = \Theta(1/\sqrt{L})$ and therefore

$$\lambda_l = \Theta(1/\sqrt{L}).$$

Biases. As in Appendix C.5, the raw-gradient scale of input biases is $\|\nabla_{\boldsymbol{b}_0} \mathcal{L}\|_{\text{R}} = \Theta(1/n_{\text{out}})$. For hidden biases $\boldsymbol{b}_l \in \mathbb{R}^{n_{\text{out}} \times 1}$, we similarly have

$$\Theta(1/L) = \Delta_{\boldsymbol{b}_l} \mathcal{L} = \Theta(\langle \Delta \boldsymbol{b}_l, \nabla_{\boldsymbol{b}_l} \mathcal{L} \rangle) = \Theta(\|\Delta \boldsymbol{b}_l\|_2 \|\nabla_{\boldsymbol{b}_l} \mathcal{L}\|_2),$$

Table 12: **μP implementation of Condition B.1 ($k = 1$) for AdamW (Loshchilov & Hutter, 2019), Lion (Chen et al., 2023), and Sophia (Liu et al., 2024b) with weight decay under width–depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image. The initial variance of input bias is σ^2_{base}.

	Input weights & biases	Hidden weights	Output weights	Hidden biases
Block Multiplier	α_{base}	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})	α_{base}/r_n (α_{base})	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}	σ^2_{base}
Learning Rate	η_{base}	$\eta_{\text{base}}/(r_n\sqrt{r_L})$ (η_{base})	η_{base}	$\eta_{\text{base}}/\sqrt{r_L}$ (η_{base})
Weight Decay	λ_{base}	$\lambda_{\text{base}} r_n$ (λ_{base})	λ_{base}	λ_{base}
AdamW ε	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_n\sqrt{r_L})$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_n\sqrt{r_L})$ $(\varepsilon_{\text{base}})$

Since we set $\alpha_l\|\Delta \boldsymbol{b}_l\|_{\text{R}} = \Theta(1/\sqrt{L}) \cdot \|\Delta \boldsymbol{b}_l\|_{\text{R}} = \Theta(1/L)$ to satisfy the one-layer bias update condition in Condition B.3, we have $\|\Delta \boldsymbol{b}_l\|_{\text{R}} = \Theta(1/\sqrt{L})$ and thus $\|\Delta \boldsymbol{b}_l\|_2 = \Theta(\sqrt{n_{\text{out}}/L})$. Therefore, $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2 = \Theta(1/\sqrt{n_{\text{out}}L})$, which leads to

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}} = \sqrt{\frac{1}{n_{\text{out}}}}\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2 = \Theta(\frac{1}{n_{\text{out}}\sqrt{L}}).$$

To sum up, we can estimate the scale of $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}}$ as

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}} = \begin{cases} \Theta(1/n_{\text{out}}), & l = 0, \\ \Theta\left(1/(n_{\text{out}}\sqrt{L})\right), & l \in [L]. \end{cases}$$

Requiring $\alpha_0\|\Delta \boldsymbol{b}_0\|_{\text{R}} = \alpha_0\eta_{\boldsymbol{b}_0}\|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_{\text{R}} = \Theta(1)$ and $\alpha_l\|\Delta \boldsymbol{b}_l\|_{\text{R}} = \alpha_l\eta_{\boldsymbol{b}_l}\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}} = \Theta(1/L)$ for $l \in [L]$ gives

$$\eta_{\boldsymbol{b}_l} = \Theta(n_{\text{out}}), \quad l \le L.$$

For the weight decays, we match $\lambda_{\boldsymbol{b}_l}\|\boldsymbol{b}_l\|_{\text{R}}$ to $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}}$. Given $\|\boldsymbol{b}_l\|_{\text{R}} = \Theta(1)$ by the initialization implementation in Condition B.3, we have

$$\lambda_{\boldsymbol{b}_l} = \begin{cases} \Theta(1/n_{\text{out}}), & l = 0, \\ \Theta\left(1/(n_{\text{out}}\sqrt{L})\right), & l \in [L]. \end{cases}$$

This completes the implementation of Condition B.1 for SGD with weight decay, as summarized in Table 11.

D.5 AdamW, Sophia and Lion

In this section, we derive the μP implementation from Condition B.1 for AdamW (same for Sophia and Lion), which recovers and extends the μP scaling rules studied in Yang et al. (2024).

As in Appendix C.6, we reduce AdamW to sign gradient descent by setting $\beta_1 = 0$, $\beta_2 = 0$, and $\varepsilon_l = 0$:

$$\Delta \boldsymbol{W}_l = -\eta_l\big(\underbrace{\operatorname{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)}_{\boldsymbol{A}_l} + \lambda_l \boldsymbol{W}_l\big).$$

Using the norm estimate in Equation (21), we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \|\operatorname{sign}\left(\nabla_{\boldsymbol{W}_l}\mathcal{L}\right)\|_{\text{R}} = \Theta(n_{\text{in}}) = \begin{cases} \Theta(1), & l = 0, \\ \Theta(n_{\text{in}}), & l \in [L], \\ \Theta(n_{\text{in}}), & l = L + 1. \end{cases}$$

As discussed in Appendices C.8 and C.7, Sophia and Lion share the same parameterization as AdamW under the simplifications used in this section.

D.5.1 Derivation of Parameterization

Input and output layers. The input and output layer conditions in Condition B.1 are identical to those in Condition 3.1. Therefore, as in Appendix C.6, their AdamW learning-rate and weight-decay parameterizations are

$$\eta_0 = \Theta(1), \quad \eta_{L+1} = \Theta(1), \quad \lambda_0 = \Theta(1), \quad \lambda_{L+1} = \Theta(1).$$

Hidden layers (first-order). Unlike SGD, the sign-style update direction has norm determined by dimension rather than by the raw-gradient magnitude. Thus, the only change from the $k = 2$ AdamW implementation is the hidden multiplier: Condition B.1 uses the Depth-μP-style choice $\alpha_l = \Theta(1/\sqrt{L})$ rather than $\Theta(1/L)$. When $\lambda_l = 0$, given the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ above, we have

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/\sqrt{L}) \cdot \eta_l \|\boldsymbol{A}_l\|_{\mathrm{R}} = \Theta(\eta_l n_{\mathrm{in}}/\sqrt{L}).$$

As desired in Equation (Υ1), to satisfy the first-order update condition on hidden weights in Condition B.1, $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L)$, we need to set

$$\eta_l = \Theta(\frac{1}{n_{\mathrm{in}}\sqrt{L}}).$$

When $\lambda_l \neq 0$, the weight-decay parameterization is unchanged from the $k = 2$ AdamW implementation because it matches the scale of $\boldsymbol{A}_l$ and does not depend on the residual multiplier. Given the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(\lambda_l).$$

To satisfy Equation (Υ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$, we need to set

$$\lambda_l = \Theta(n_{\mathrm{in}}).$$

Biases. For bias parameters $\boldsymbol{b}_l \in \mathbb{R}^{n_{\mathrm{out}} \times 1}$, by the definition we have

$$\|\mathrm{sign}\left(\nabla_{\boldsymbol{b}_l} \mathcal{L}\right)\|_{\mathrm{R}} = \Theta(1), \quad 0 \leq l \leq L.$$

To satisfy the one-layer bias update condition, $\alpha_0 \|\Delta \boldsymbol{b}_0\|_{\mathrm{R}} = \Theta(\eta_{\boldsymbol{b}_0}) = \Theta(1)$ and $\alpha_l \|\Delta \boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1/\sqrt{L} \cdot \eta_{\boldsymbol{b}_l}) = \Theta(1/L)$ for $l \in [L]$, we need to set

$$\eta_{\boldsymbol{b}_0} = \Theta(1), \quad \eta_{\boldsymbol{b}_l} = \Theta(1/\sqrt{L}), \quad l \in [L].$$

For the weight decays, we match $\lambda_{\boldsymbol{b}_l} \|\boldsymbol{b}_l\|_{\mathrm{R}}$ to $\|\mathrm{sign}\left(\nabla_{\boldsymbol{b}_l} \mathcal{L}\right)\|_{\mathrm{R}} = \Theta(1)$. Given $\|\boldsymbol{b}_l\|_{\mathrm{R}} = \Theta(1)$ by the initialization implementation in Condition B.3, we have

$$\lambda_{\boldsymbol{b}_l} = \Theta(1), \quad 0 \leq l \leq L.$$

Parameterization of ε_l. To make the stabilization term ε_l effective and not dominate the gradient, we desire it to be of the same scale as $\sqrt{\hat{\boldsymbol{v}}_l^{(t)}}$. When omitting the momentum, we have $\sqrt{\hat{\boldsymbol{v}}_l} = \nabla_{\boldsymbol{W}_l} \mathcal{L}$. Therefore, we need to ensure $\varepsilon_l = \Theta(\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_l)} \mathcal{L}\|_{\mathrm{R}})$. The latter can be estimated from the raw-gradient derivation for $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_{\mathrm{R}}$ in Equation (25) of Appendix D.4.

For the input weights $\boldsymbol{W}_0$, we have

$$\|\nabla_{\mathrm{Vec}(\boldsymbol{W}_0)} \mathcal{L}\|_{\mathrm{R}} = \frac{1}{\sqrt{n_{\mathrm{in}} n_{\mathrm{out}}}} \|\nabla_{\boldsymbol{W}_0} \mathcal{L}\|_{\mathrm{F}} = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}} n_{\mathrm{out}}}} \|\nabla_{\boldsymbol{W}_0} \mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}} n_{\mathrm{out}}}} \sqrt{\frac{n_{\mathrm{in}}}{n_{\mathrm{out}}}}\right) = \Theta(\frac{1}{n_{\mathrm{out}}}).$$

Therefore, we set

$$\varepsilon_0 = \Theta(\frac{1}{n_{\mathrm{out}}}).$$

Table 13: **μP implementation of Condition B.1 ($k = 1$) for SSO (Xie et al., 2026) with weight decay under width-depth scaling.** Entries in purple indicate differences between μP and SP, while gray shows the corresponding SP choices. Here, r_n and r_L denote the width and depth scaling ratios relative to the base model. The variance of input weights is σ^2_{base} for language and $\sigma^2_{\text{base}}/d_0$ for image.

	Input weights	Hidden weights	Output weights
Block Multiplier	α_{base}	$\alpha_{\text{base}}/\sqrt{r_L}$ (α_{base})	α_{base}/r_n (α_{base})
Initial Variance	$\sigma^2_{\text{base}}/d_0$ or σ^2_{base}	$\sigma^2_{\text{base}}/r_n$ (σ^2_{base})	σ^2_{base}
Learning Rate	η_{base}	$\eta_{\text{base}}/\sqrt{r_L}$ (η_{base})	$\eta_{\text{base}} r_n$ (η_{base})
Weight Decay	λ_{base}	λ_{base}	$\lambda_{\text{base}}/r_n$ (λ_{base})

For the hidden weights $\boldsymbol{W}_l$ where $l \in [L]$, we have

$$\|\nabla_{\text{Vec}(\boldsymbol{W}_l)}\mathcal{L}\|_{\text{R}} = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}n_{\text{out}}}}\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}n_{\text{out}}}}\sqrt{\frac{n_{\text{in}}}{n_{\text{out}}L}}\right) = \Theta(\frac{1}{n_{\text{out}}\sqrt{L}}).$$

Therefore, we set

$$\varepsilon_l = \Theta(\frac{1}{n_{\text{out}}\sqrt{L}}), \quad l \in [L].$$

For the output weights $\boldsymbol{W}_{L+1}$, we have

$$\|\nabla_{\text{Vec}(\boldsymbol{W}_{L+1})}\mathcal{L}\|_{\text{R}} = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}n_{\text{out}}}}\|\nabla_{\boldsymbol{W}_{L+1}}\mathcal{L}\|_2\right) = \Theta\left(\frac{1}{\sqrt{n_{\text{in}}n_{\text{out}}}}\sqrt{\frac{1}{n_{\text{in}}n_{\text{out}}}}\right) = \Theta(\frac{1}{n_{\text{in}}}).$$

Therefore, we set

$$\varepsilon_{L+1} = \Theta(\frac{1}{n_{\text{in}}}).$$

Similarly, for the biases we have derived in Appendix D.4 that

$$\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\text{R}} = \begin{cases}\Theta(1/n_{\text{out}}), & l = 0,\\ \Theta(1/(n_{\text{out}}\sqrt{L})), & l \in [L].\end{cases}$$

Therefore, we set the stabilization term as

$$\varepsilon_{\boldsymbol{b}_l} = \begin{cases}\Theta(1/n_{\text{out}}), & l = 0,\\ \Theta(1/(n_{\text{out}}\sqrt{L})), & l \in [L].\end{cases}$$

This completes the implementation of Condition B.1 for AdamW, Lion, and Sophia with weight decay, as summarized in Table 12.

D.6 Spectral Sphere Optimizer (SSO)

In this section, we derive the μP implementation from Condition B.1 for SSO.

As in Appendix C.11, SSO (Xie et al., 2026) uses the update

$$\Delta\boldsymbol{W}_l = -\eta_l\big(\underbrace{R\boldsymbol{\Phi}_l}_{\boldsymbol{A}_l} + \lambda_l\boldsymbol{W}_l\big),$$

where

$$R = \Theta\left(\sqrt{\frac{n_{\text{out}}}{n_{\text{in}}}}\right), \quad \text{and} \quad \boldsymbol{\Phi}_l = \arg\max_{\boldsymbol{\Phi}}\langle\nabla_{\boldsymbol{W}_l}\mathcal{L}, \boldsymbol{\Phi}\rangle \text{ s.t. } \|\boldsymbol{\Phi}\|_2 = 1,\ \|\boldsymbol{W}_l - \eta_l\boldsymbol{\Phi}\|_2 = \|\boldsymbol{W}_l\|_2 = R.$$

Using the norm computation in Equation (24), we have

$$\|\boldsymbol{A}_l\|_{\mathrm{R}} = \Theta(1).$$

D.6.1 Derivation of Parameterization

Input and output layers. The input and output layer conditions in Condition B.1 are identical to those in Condition 3.1. Therefore, as in Appendix C.11, their SSO learning-rate and weight-decay parameterizations are

$$\eta_0 = \Theta(1), \quad \eta_{L+1} = \Theta(n_{\mathrm{in}}), \quad \lambda_0 = \Theta(1), \quad \lambda_{L+1} = \Theta(1/n_{\mathrm{in}}).$$

Hidden layers (first-order). The only change from the $k = 2$ SSO implementation is the hidden multiplier: Condition B.1 uses the Depth-μP-style choice $\alpha_l = \Theta(1/\sqrt{L})$ rather than $\Theta(1/L)$. When $\lambda_l = 0$, given the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ above, we have

$$\alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/\sqrt{L}) \cdot \eta_l \|\boldsymbol{A}_l\|_{\mathrm{R}} = \Theta(\eta_l/\sqrt{L}).$$

As desired in Equation (Υ1), to satisfy the first-order update condition on hidden weights in Condition B.1, $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L)$, we need to set

$$\eta_l = \Theta(1/\sqrt{L}).$$

When $\lambda_l \neq 0$, the weight-decay parameterization is unchanged from the $k = 2$ SSO implementation because it matches the scale of $\boldsymbol{A}_l$ and does not depend on the residual multiplier. Given the weight norm $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (6), we have

$$\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(\lambda_l).$$

To satisfy Equation (Υ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right)$, we need to set

$$\lambda_l = \Theta(1).$$

This completes the implementation of Condition B.1 for SSO with weight decay, as summarized in Table 13.

E Additional Details and Results of GPT-2 Experiments

This section provides the experimental details and complete numerical results for the GPT-2 style language-model experiments in Section 5. We organize the results around three comparisons. First, we compare SP with the μP formulation derived from Condition 3.1 ($k \geq 2$), which corresponds to the CompleteP-style scaling predicted for residual branches with multiple transformations. Second, we compare this formulation with the μP formulation derived from Condition B.1 ($k = 1$), which corresponds to the Depth-μP-style scaling for one-layer residual branches. Third, we report additional weight-decay transfer results.

E.1 Assets and Licenses

All used assets (datasets and codes) and their licenses are listed in Table 14.

E.2 Additional Details of Feature Learning Experiments

This subsection reports the coordinate-check experiments used to evaluate feature-scale stability under width and depth scaling. We measure the RMS norm at the output of the final Transformer block, $\|\boldsymbol{h}_L\|_{\mathrm{R}}$, after short training runs. Results are averaged over three independent runs with random seeds 1, 2, and 3. The base initialization variance for matrix weights and biases is set to 0.02^2 and 0, respectively. All models are trained with a constant learning-rate schedule, batch size 8, gradient clipping 1.0, and no weight decay for 10 training steps. The HP scaling rule can be found in Tables 2 and 8 for μP ($k \geq 2$) and μP ($k = 1$), respectively. Optimizer-specific HPs are listed below.

Table 14: **Used assets and their licenses.**

URL and citation	License
https://github.com/EleutherAI/nanoGPT-mup/tree/completep (Dey et al., 2025)	MIT
https://github.com/karpathy/nanoGPT (Karpathy, 2022)	MIT
https://skylion007.github.io/OpenWebTextCorpus/ (Gokaslan & Cohen, 2019)	Creative Commons CC0

Muon-Kimi-AdamW. Following common practice (Liu et al., 2025), hidden matrix parameters are optimized by Muon-Kimi with a Nesterov-style momentum (Nesterov, 1983) of 0.95. All other parameters (e.g., embedding layer, LM head, all biases) are updated by AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon_{\text{base}} = 10^{-16}$. The learning rate is 2^{-7} for both Muon-Kimi and AdamW. The results are presented in Figure 1(a,b).

Muon-AdamW. Following common practice (Jordan et al., 2024b), hidden matrix parameters are optimized by Muon with a learning rate of 0.02 and a Nesterov-style momentum of 0.95. All other parameters (e.g., embedding layer, LM head, all biases) are updated by AdamW with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon_{\text{base}} = 10^{-16}$. The results are presented in Figure 5(a,b).

Shampoo-AdamW. Following common practice (Jordan et al., 2024a), hidden matrix parameters are optimized by Shampoo with a learning rate of 0.001, $\beta_1 = 0.95$, $\beta_2 = 0.95$, $\epsilon_{\text{base}} = 10^{-16}$, a shampoo precondition frequency of 1, and a maximal precondition dimension of 20000. All other parameters (e.g., embedding layer, LM head, all biases) are updated by AdamW with a learning rate of 0.002, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon_{\text{base}} = 10^{-16}$. The results are presented in Figure 6(a,b).

Sophia. Following common practice (Liu et al., 2024b), parameters are updated by Sophia with a learning rate of 0.001, $\beta_1 = 0.965$, $\beta_2 = 0.99$, $\rho = 0.05$, and a Hessian update frequency of 1. The results are presented in Figure 8(a,b).

Overview of results. Across these optimizer settings, the coordinate-check results show the same qualitative pattern as in the main text: SP exhibits feature-scale growth under width or depth scaling, whereas the μP formulation from Condition 3.1 ($k \geq 2$) keeps the final-block feature norm approximately scale-invariant.

E.3 Additional Details of HP Transfer Experiments

This subsection provides the detailed setup and complete numerical results for the HP-transfer experiments in Section 5. We organize the results by optimizer. For each optimizer, we first compare SP with the μP formulation from Condition 3.1 ($k \geq 2$) under width and depth scaling. When applicable, we then compare this formulation with the μP formulation from Condition B.1 ($k = 1$) to test the role of residual-block depth. For Muon-Kimi-AdamW, we additionally report weight-decay transfer and the no-LayerNorm depth-scaling diagnostic discussed in the main text.

E.3.1 Basic Experimental Setup

Unless otherwise stated, all HP-transfer experiments use GPT-2 style models trained on OpenWebText with sequence length 1024 and the GPT-2 tokenizer, following the setup in Section 5. The base model has width $n_{\text{base}} = 256$ and depth $L_{\text{base}} = 4$. For width-scaling experiments, we vary n while keeping $L = 4$; for depth-scaling experiments, we vary L while keeping $n = 256$. The base initialization variance for matrix weights is set to 0.02^2, and biases are initialized to zero.

All models are trained with batch size 240 for 1221 iterations, corresponding to about 300M tokens, using 120 warmup iterations followed by cosine learning-rate decay to a minimum learning rate of 3×10^{-5}, gradient clipping of 1.0, and the optimizer-specific settings described below.

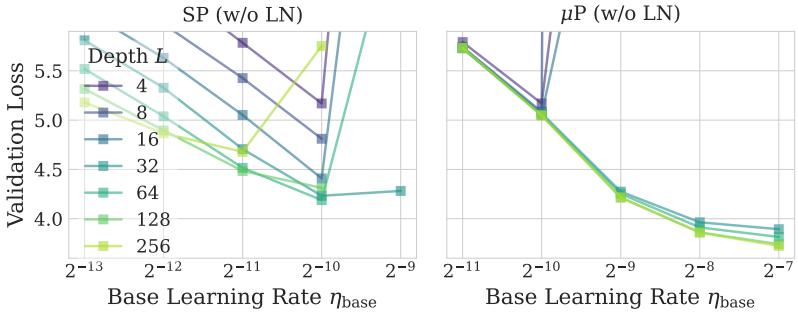

Figure 3: **Feature learning and HP transfer of Muon-Kimi-AdamW under SP and μP from Condition 3.1 ($k \geq 2$) without LayerNorm.** First, in terms of training stability, SP becomes increasingly prone to loss divergence as depth increases in the absence of LayerNorm, whereas μP enables stable training. Second, unlike SP, μP preserves HP transferability at large depths without LayerNorm.

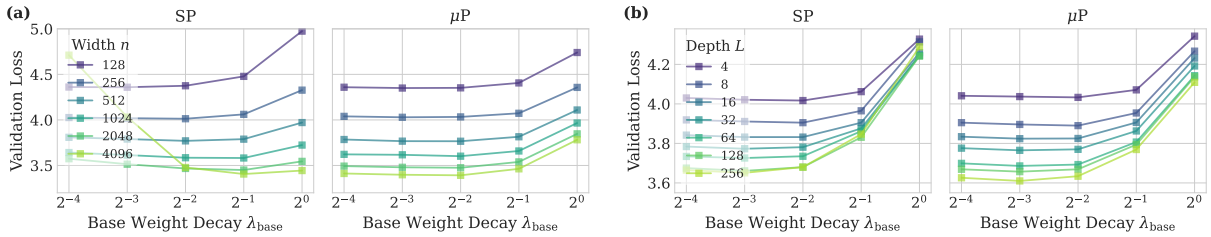

Figure 4: **Weight decay transfer of Muon-Kimi-AdamW under SP and μP from Condition 3.1 ($k \geq 2$).** We train GPT-2 style models with Muon-Kimi and AdamW using SP and μP derived from Condition 3.1 (see Tables 3 and 6). μP enables robust HP transfer across both width and depth scaling, while consistently achieving lower loss than SP as the model depth increases. The corresponding numerical values are reported in Appendix E.3.2.

The HPs shown in the tables are base HPs, such as η_{base} or λ_{base}; the actual model HPs are obtained by applying the corresponding SP or μP scaling rules. *The μP implementation can be found in Tables 2 and 8 for Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$)*, respectively. We report the final validation loss.

E.3.2 Additional Details of Muon-Kimi-AdamW

This subsection provides the complete Muon-Kimi-AdamW results underlying Figures 1, 2, 3, and 4. We first report learning-rate transfer under width and depth scaling, including the comparison between Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$). We then report the no-LayerNorm diagnostic and the weight-decay transfer experiments.

Experimental setup. Following common practice (Liu et al., 2025), hidden matrix parameters are optimized by Muon-Kimi with Nesterov-style momentum 0.95, while all other parameters (e.g., embeddings, LM head, and biases) are optimized by AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon_{\text{base}} = 10^{-16}$. For learning-rate transfer experiments, both Muon-Kimi and AdamW use the same base learning rate η_{base}, and weight decay is disabled. For weight-decay transfer experiments, we fix the base learning rate to 2^{-7} and sweep the base weight decay λ_{base}.

Additional results of width-wise learning rate transfer. Complete numerical results of base learning rate transferability across different widths are presented in Table 15 and Table 16 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Table 15: **For Muon-Kimi-AdamW, SP fails to transfer the optimal base learning rate across widths.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-10	-9	-8	-7	-6	-5
128	5.127	4.685	4.45	4.373	**4.364**	4.372
256	4.552	4.219	4.081	**4.053**	4.062	4.091
512	4.093	3.886	**3.819**	3.833	3.837	4.062
1024	3.817	3.699	**3.672**	3.68	3.952	5.603
2048	3.654	3.571	**3.555**	3.798	5.472	6.438
4096	3.56	**3.516**	3.747	5.557	6.159	6.552

Table 16: **For Muon-Kimi-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across widths, and achieves lower loss than SP as the width increases.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-10	-9	-8	-7	-6	-5
128	4.875	4.53	4.42	**4.374**	4.383	4.397
256	4.561	4.227	4.081	**4.059**	4.079	4.104
512	4.305	3.974	3.83	**3.811**	3.828	3.873
1024	4.125	3.798	3.654	**3.646**	3.676	3.726
2048	3.957	3.636	3.516	**3.515**	3.552	3.689
4096	3.882	3.531	**3.446**	3.461	3.523	3.752

Additional results of depth-wise learning rate transfer with LayerNorm. With LayerNorm, complete numerical results of base learning rate transferability across different depths are presented in Table 17, Table 18, and Table 19 for SP, μP from Condition 3.1 ($k \geq 2$), and μP from Condition B.1 ($k = 1$), respectively. As discussed in Section 5, this apparent depth-wise transfer under SP should be interpreted cautiously, because LayerNorm and the tested depth range can partially mask feature-scale instability.

Additional results of depth-wise learning rate transfer without LayerNorm. Without LayerNorm, depth-wise base learning-rate transfer results for SP and μP from Condition 3.1 ($k \geq 2$) are shown in Figure 3, with complete numerical results in Tables 20 and 21.

Additional results of width-wise weight decay transfer. Width-wise base weight-decay transfer results are shown in Figure 4(a). Complete numerical results are presented in Table 22 and Table 23 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Additional results of depth-wise weight decay transfer with LayerNorm. With LayerNorm, depth-wise base weight-decay transfer results are shown in Figure 4(b). Complete numerical results are presented in Table 24 and Table 25 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Table 17: **For Muon-Kimi-AdamW, with LayerNorm, SP transfers the optimal base learning rate across the tested depths.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-9	-8	-7	-6	-5
4	4.219	4.081	**4.056**	4.067	4.09
8	4.109	3.985	**3.952**	3.973	4.013
16	4.016	3.893	**3.864**	3.889	3.929
32	3.949	3.824	**3.799**	3.82	3.885
64	3.916	3.777	**3.747**	3.777	3.91
128	3.898	3.75	**3.723**	3.772	4.031
256	3.883	3.719	**3.688**	3.753	4.174

Table 18: **For Muon-Kimi-AdamW, with LayerNorm, μP from Condition 3.1 ($k \geq 2$) transfers the optimal base learning rate across depths, and achieves lower loss than SP as the depth increases.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-9	-8	-7	-6	-5
4	4.228	4.081	**4.06**	4.075	4.098
8	4.089	3.972	**3.938**	3.957	3.988
16	4.01	3.886	**3.85**	3.874	3.907
32	3.96	3.826	**3.8**	3.828	3.879
64	3.917	3.771	**3.747**	3.796	3.942
128	3.878	3.715	**3.694**	3.754	4.002
256	3.878	3.697	**3.678**	3.761	3.964

Table 19: **For Muon-Kimi-AdamW, with LayerNorm, μP from Condition B.1 ($k = 1$) fails to transfer the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 2, where the best validation loss for each depth is highlighted in **bold**. The implementation can be found in Table 9 and 12.

$L/\log_2(\eta_{\text{base}})$	-9	-8	-7	-6	-5	-4
4	4.225	4.081	**4.06**	4.067	4.095	4.971
8	4.149	3.988	**3.923**	3.934	3.951	4.03
16	4.198	3.963	3.871	**3.857**	3.88	3.922
32	4.325	3.987	3.843	**3.796**	3.814	3.855
64	4.517	4.052	3.825	3.728	**3.725**	3.768
128	4.666	4.242	3.912	3.745	**3.7**	3.746
256	4.774	4.454	4.025	3.788	**3.667**	3.681

Table 20: **For Muon-Kimi-AdamW, without LayerNorm, SP fails to preserve stable training.** NaN data points indicate training instability, where the loss explodes.

$L/\log_2(\eta_{\text{base}})$	-13	-12	-11	-10	-9
4	7.318	6.394	5.784	5.169	13.77
8	6.775	5.974	5.426	4.811	NaN
16	6.115	5.631	5.052	4.409	10.814
32	5.809	5.328	4.706	4.233	4.282
64	5.519	5.038	4.516	4.189	7.251
128	5.316	4.896	4.484	4.313	NaN
256	5.179	4.867	4.678	5.752	NaN

Table 21: **For Muon-Kimi-AdamW, without LayerNorm, μP from Condition 3.1 ($k \geq 2$) has stable runs and approximately transfers the optimal base learning rate at large depth $L \geq 32$.** NaN data points indicate training instability, where the loss explodes. The best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-11	-10	-9	-8	-7
4	5.791	**5.169**	11.6	NaN	345.85
8	5.741	**5.084**	131.43	NaN	NaN
16	5.73	**5.059**	8.732	246.45	122.99
32	5.734	5.069	4.275	3.964	**3.894**
64	5.73	5.051	4.253	3.912	**3.815**
128	5.728	5.052	4.214	3.862	**3.742**
256	5.733	5.045	4.217	3.859	**3.724**

Table 22: **For Muon-Kimi-AdamW, SP fails to transfer the optimal base weight decay across widths.** This table reports the numerical results corresponding to Figure 4, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\lambda_{\text{base}})$	-4	-3	-2	-1	0
128	4.361	**4.359**	4.375	4.478	4.975
256	4.026	4.019	**4.013**	4.061	4.327
512	3.809	3.79	**3.77**	3.789	3.971
1024	3.642	3.614	3.585	**3.582**	3.724
2048	3.574	3.514	3.467	**3.45**	3.545
4096	4.711	4.032	3.478	**3.406**	3.444

Table 23: **For Muon-Kimi-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base weight decay across widths.** This table reports the numerical results corresponding to Figure 4, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\lambda_{\text{base}})$	-4	-3	-2	-1	0
128	4.359	**4.35**	4.352	4.406	4.74
256	4.039	**4.029**	4.033	4.073	4.357
512	3.785	3.767	**3.766**	3.815	4.109
1024	3.622	3.617	**3.602**	3.659	3.966
2048	3.493	3.48	**3.475**	3.539	3.847
4096	3.412	3.398	**3.392**	3.463	3.782

Table 24: This table reports the numerical results corresponding to **depth scaling of Muon-Kimi-AdamW under SP** in Figure 4, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\lambda_{\text{base}})$	-4	-3	-2	-1	0
4	4.03	4.021	**4.017**	4.062	4.328
8	3.919	3.911	**3.905**	3.965	4.314
16	3.842	3.832	**3.832**	3.905	4.259
32	3.784	**3.773**	3.781	3.876	4.246
64	3.734	**3.726**	3.735	3.862	4.244
128	3.675	**3.661**	3.68	3.831	4.242
256	3.665	**3.651**	3.68	3.847	4.289

Table 25: **For Muon-Kimi-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base weight decay across depths, and achieves lower loss than SP as the depth increases.** This table reports the numerical results corresponding to Figure 4, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\lambda_{\text{base}})$	-4	-3	-2	-1	0
4	4.041	4.037	**4.033**	4.071	4.343
8	3.905	3.896	**3.89**	3.954	4.267
16	3.834	**3.824**	3.825	3.906	4.232
32	3.776	**3.765**	3.77	3.863	4.192
64	3.699	**3.686**	3.693	3.807	4.143
128	3.669	**3.657**	3.669	3.794	4.137
256	3.626	**3.61**	3.634	3.769	4.109

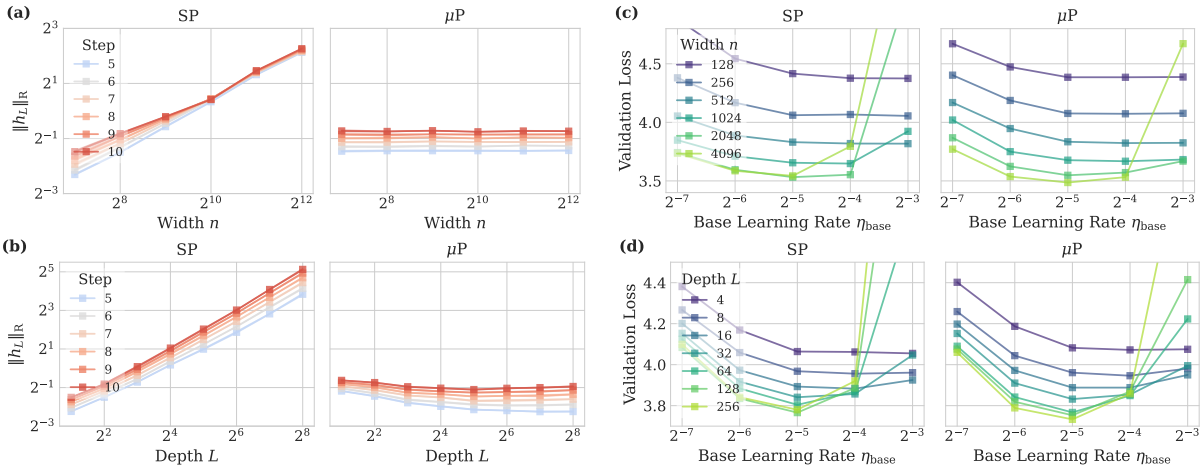

Figure 5: **Feature learning and HP transfer of Muon-AdamW under SP and μP.** We train GPT-2 style models with Muon-AdamW using SP and μP derived from Condition 3.1 (see Tables 4 and 6). μP maintains stable feature norms and enables robust HP transfer across both width and depth scaling, while generally achieving lower loss than SP as the model size increases. The detailed numerical values are provided in Appendix E.3.3.

Table 26: **For Muon-AdamW, SP fails to transfer the optimal base learning rate across widths.** This table reports the numerical results corresponding to Figure 5, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
128	4.88	4.544	4.416	4.377	**4.375**
256	4.38	4.168	4.061	4.067	**4.055**
512	4.054	3.888	3.831	3.819	**3.818**
1024	3.848	3.711	3.655	**3.648**	3.923
2048	3.742	3.595	**3.532**	3.553	5.137
4096	3.735	3.586	**3.543**	3.795	5.992

E.3.3 Additional Details of Muon-AdamW

This subsection provides the complete Muon-AdamW results underlying Figures 5 and 2. We report learning-rate transfer under width and depth scaling, including the comparison between SP, Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$).

Experimental setup. Following common practice (Jordan et al., 2024b;a), hidden matrix parameters are optimized by Muon with a base learning rate of η_{base}, and a Nesterov-style momentum of 0.95, while all other parameters (e.g., embedding layer, LM head, all biases) are updated by AdamW with a base learning rate of $\eta_{\text{base}}/10$, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon_{\text{base}} = 10^{-16}$. We do not use weight decay in all learning rate transfer experiments.

Additional results of width-wise learning rate transfer. Complete numerical results of base learning rate transferability across different widths are presented in Table 26 and Table 27 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Table 27: **For Muon-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across widths, and achieves lower loss than SP as the width increases.** This table reports the numerical results corresponding to Figure 5, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
128	4.671	4.473	**4.385**	4.385	4.387
256	4.403	4.187	4.076	**4.073**	4.077
512	4.169	3.947	3.834	**3.823**	3.825
1024	4.02	3.75	3.677	**3.668**	3.682
2048	3.868	3.624	**3.548**	3.571	3.669
4096	3.771	3.537	**3.486**	3.532	4.672

Table 28: **For Muon-AdamW, SP shifts the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 5, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
4	4.381	4.169	4.064	4.062	**4.055**
8	4.267	4.059	3.968	**3.956**	3.961
16	4.201	3.973	3.893	**3.883**	3.925
32	4.153	3.917	**3.841**	3.857	4.047
64	4.124	3.884	**3.804**	3.868	4.772
128	4.084	3.835	**3.766**	3.886	5.798
256	4.099	3.841	**3.78**	3.92	6.705

Additional results of depth-wise learning rate transfer. Complete numerical results of base learning rate transferability across different depths are presented in Table 28, Table 29, and Table 30 for SP, μP from Condition 3.1 ($k \geq 2$), and μP from Condition B.1 ($k = 1$), respectively.

Table 29: **For Muon-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across depths, and achieves lower loss than SP as the depth increases.** This table reports the numerical results corresponding to Figure 5, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
4	4.402	4.187	4.082	**4.072**	4.075
8	4.259	4.044	3.961	**3.946**	3.981
16	4.198	3.972	**3.888**	3.888	3.95
32	4.152	3.91	**3.832**	3.854	3.995
64	4.09	3.842	**3.767**	3.849	4.223
128	4.076	3.819	**3.753**	3.869	4.415
256	4.06	3.789	**3.733**	3.858	5.201

Table 30: **For Muon-AdamW, μP from Condition B.1 ($k = 1$) fails to transfer the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 2, where the best validation loss for each depth is highlighted in **bold**. The implementation can be found in Table 10 and 12.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3	-2
4	4.402	4.189	4.078	4.07	4.073	**4.068**
8	4.323	4.067	3.97	**3.937**	3.96	3.983
16	4.368	4.052	3.909	**3.862**	3.881	3.947
32	4.477	4.073	3.884	**3.809**	3.823	3.902
64	4.649	4.12	3.874	3.754	**3.735**	3.834
128	4.793	4.262	3.946	3.78	**3.722**	3.795
256	4.887	4.447	4.031	3.815	**3.707**	3.72

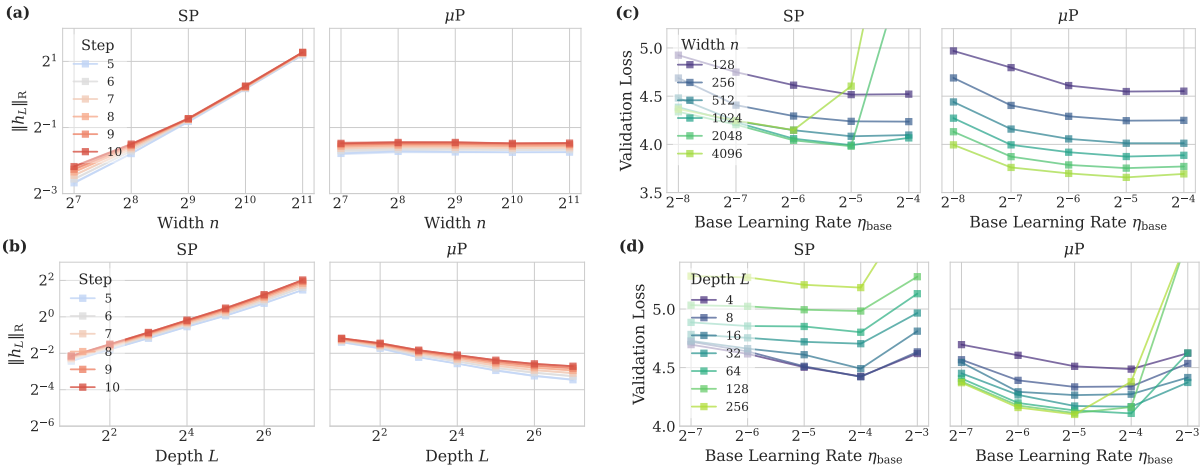

Figure 6: **Feature learning and HP transfer of Shampoo-AdamW under SP and μP.** We train GPT-2 style models with Shampoo-AdamW using SP and μP derived from Condition 3.1 (see Tables 4 and 6). μP maintains stable feature norms and enables robust HP transfer across both width and depth scaling, while generally achieving lower loss than SP as the model size increases. The detailed numerical values are provided in Appendix E.3.4.

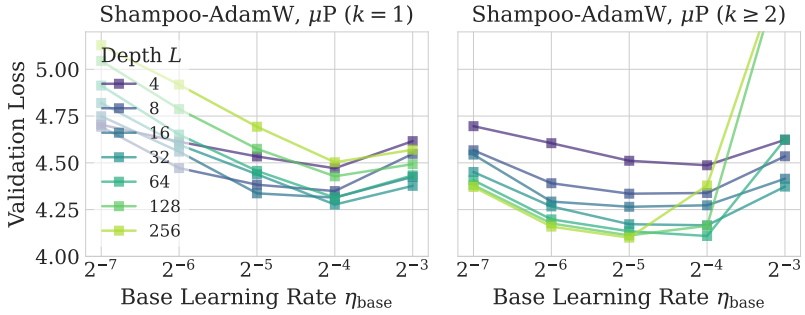

Figure 7: **Validating the role of residual block depth k.** We compare two μP implementations for GPT-2-style models trained with Shampoo-AdamW: Depth-μP-style formulation from Condition B.1 ($k = 1$) and CompleteP-style formulation from Condition 3.1 ($k \geq 2$). Condition 3.1 yields more stable HP transfer and lower validation loss, empirically supporting it as the appropriate μP condition for architectures with multi-transformation residual branches. The numerical values are provided in Appendix E.3.4.

E.3.4 Additional Details of Shampoo-AdamW

This subsection provides the complete Shampoo-AdamW results underlying Figures 6 and 7. We report learning-rate transfer under width and depth scaling, including the comparison between SP, Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$).

Experimental setup. Following common practice (Jordan et al., 2024a), hidden matrix parameters are optimized by Shampoo with a base learning rate of η_{base}, $\beta_1 = 0.95$, $\beta_2 = 0.95$, a shampoo precondition frequency of 1, a maximal precondition dimension of 20000, $\epsilon_{\text{base}} = 10^{-8}$ for width scaling, and $\epsilon_{\text{base}} = 10^{-5}$ for depth scaling, while all other parameters (e.g., embedding layer, LM head, all biases) are updated by AdamW with a base learning rate of $2\eta_{\text{base}}$, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon_{\text{base}} = 10^{-16}$. We do not use weight decay in all learning rate transfer experiments.

Table 31: **For Shampoo-AdamW, SP fails to transfer the optimal base learning rate across widths.** This table reports the numerical results corresponding to Figure 6, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-8	-7	-6	-5	-4
128	4.925	4.749	4.614	**4.516**	4.521
256	4.688	4.407	4.294	4.239	**4.236**
512	4.481	4.242	4.147	**4.084**	4.097
1024	4.385	4.229	4.059	**3.992**	4.067
2048	4.338	4.205	4.042	**3.981**	6.002
4096	4.366	4.247	**4.148**	4.604	7.476

Table 32: **For Shampoo-AdamW, μP from Condition 3.1 ($k \geq 2$) transfers the optimal base learning rate across widths, and achieves lower loss than SP as the width increases.** This table reports the numerical results corresponding to Figure 6, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-8	-7	-6	-5	-4
128	4.969	4.796	4.61	**4.548**	4.552
256	4.689	4.404	4.291	**4.246**	4.249
512	4.44	4.159	4.057	**4.012**	4.012
1024	4.272	3.995	3.919	**3.874**	3.887
2048	4.131	3.873	3.787	**3.754**	3.771
4096	3.995	3.761	3.699	**3.658**	3.694

Additional results of width-wise learning rate transfer. Complete numerical results of base learning rate transferability across different widths are presented in Table 31 and Table 32 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Additional results of depth-wise learning rate transfer. Complete numerical results of base learning rate transferability across different depths are presented in Table 33, Table 34, and Table 35 for SP, μP from Condition 3.1 ($k \geq 2$), and μP from Condition B.1 ($k = 1$), respectively.

Table 33: **For Shampoo-AdamW, SP yields increasing validation loss under depth scaling.** This table reports the numerical results corresponding to Figure 6.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
4	4.696	4.617	4.503	4.425	4.62
8	4.721	4.638	4.511	4.421	4.634
16	4.728	4.662	4.61	4.491	4.811
32	4.783	4.754	4.72	4.704	4.966
64	4.886	4.855	4.851	4.802	5.131
128	5.033	5.023	4.994	4.983	5.276
256	5.281	5.27	5.207	5.183	5.945

Table 34: **For Shampoo-AdamW, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across depths, and achieves lower loss than SP as the depth increases.** This table reports the numerical results corresponding to Figure 6, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
4	4.696	4.605	4.511	**4.487**	4.623
8	4.567	4.391	**4.335**	4.339	4.535
16	4.544	4.293	**4.265**	4.273	4.415
32	4.451	4.267	4.172	**4.166**	4.373
64	4.407	4.198	4.134	**4.109**	4.625
128	4.381	4.176	**4.112**	4.163	5.603
256	4.371	4.159	**4.1**	4.379	5.601

Table 35: **For Shampoo-AdamW, μP from Condition B.1 ($k = 1$) gives weaker depth scaling and higher validation loss than μP from Condition 3.1 ($k \geq 2$).** This table reports the numerical results corresponding to Figure 7.

$L/\log_2(\eta_{\text{base}})$	-7	-6	-5	-4	-3
4	4.704	4.613	4.534	4.471	4.616
8	4.693	4.472	4.383	4.349	4.548
16	4.749	4.56	4.337	4.314	4.425
32	4.82	4.597	4.439	4.277	4.377
64	4.913	4.651	4.457	4.314	4.433
128	5.045	4.787	4.575	4.428	4.493
256	5.128	4.918	4.693	4.503	4.57

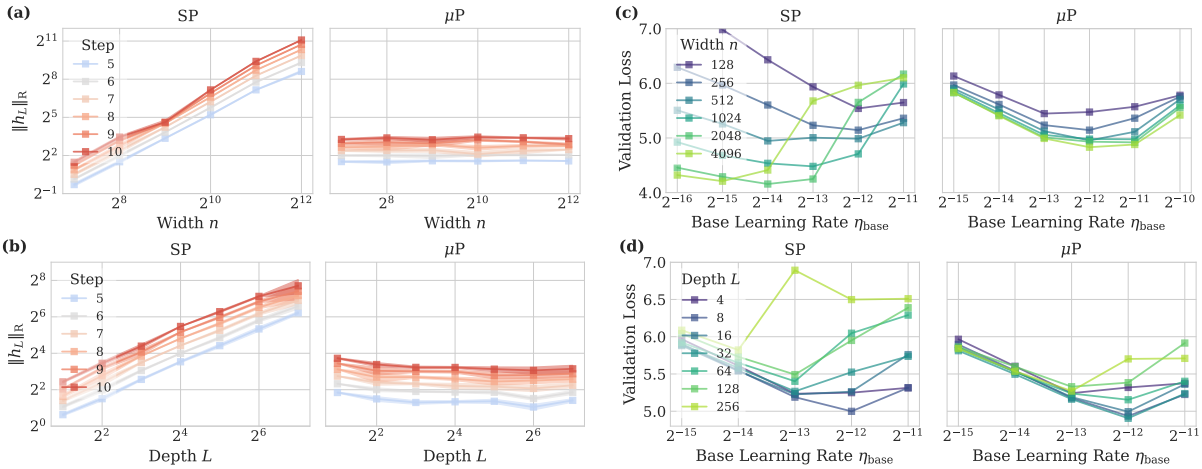

Figure 8: **Feature learning and HP transfer of Sophia under SP and μP.** We train GPT-2 style models with Sophia using SP and μP derived from Condition 3.1 (see Table 6). μP maintains stable feature norms and enables robust HP transfer across both width and depth scaling. The detailed numerical values are provided in Appendix E.3.5.

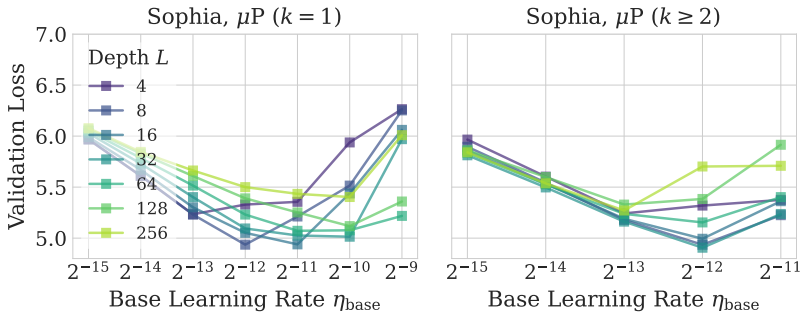

Figure 9: **Validating the role of residual block depth k.** We compare two μP implementations for GPT-2-style models trained with Sophia: Depth-μP-style formulation from Condition B.1 ($k = 1$) and CompleteP-style formulation from Condition 3.1 ($k \geq 2$). Condition 3.1 yields more stable HP transfer, empirically supporting it as the appropriate μP condition for architectures with multi-transformation residual branches. The numerical values are provided in Appendix E.3.5.

E.3.5 Additional Details of Sophia

This subsection provides the complete Sophia results underlying Figures 8 and 9. We report learning-rate transfer under width and depth scaling, including the comparison between SP, Condition 3.1 ($k \geq 2$) and Condition B.1 ($k = 1$).

Experimental setup. Following common practice (Liu et al., 2024b), parameters are updated by Sophia with a base learning rate of η_{base}, $\beta_1 = 0.965$, $\beta_2 = 0.99$, $\rho = 0.05$, and a Hessian update frequency of 10.

Additional results of width-wise learning rate transfer. Completed numerical results of base learning rate transferability across different widths are presented in Table 36 and Table 37 for SP and μP from Condition 3.1 ($k \geq 2$), respectively.

Additional results of depth-wise learning rate transfer. Complete numerical results of base learning rate transferability across different depths are presented in Table 38, Table 39, and Table 40 for SP, μP from Condition 3.1 ($k \geq 2$), and μP from Condition B.1 ($k = 1$), respectively.

Table 36: **For Sophia, SP fails to transfer the optimal base learning rate across widths.** This table reports the numerical results corresponding to Figure 8, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-16	-15	-14	-13	-12	-11
128	7.485	6.982	6.43	5.935	**5.54**	5.648
256	6.293	5.97	5.603	5.232	**5.143**	5.361
512	5.506	5.257	4.946	5.005	**4.986**	5.28
1024	4.924	4.682	4.536	**4.48**	4.708	5.988
2048	4.457	4.289	**4.157**	4.249	5.651	6.169
4096	4.321	**4.209**	4.413	5.677	5.964	6.101

Table 37: **For Sophia, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across widths larger than** 256**.** This table reports the numerical results corresponding to Figure 8, where the best validation loss for each width is highlighted in **bold**.

$n/\log_2(\eta_{\text{base}})$	-15	-14	-13	-12	-11	-10
128	6.133	5.788	**5.446**	5.474	5.571	5.781
256	5.969	5.613	5.235	**5.143**	5.362	5.756
512	5.903	5.521	5.124	**4.955**	5.113	5.707
1024	5.863	5.455	5.055	4.98	**4.976**	5.598
2048	5.827	5.419	5.01	4.932	**4.919**	5.555
4096	5.833	5.406	4.991	**4.831**	4.88	5.419

Table 38: **For Sophia, SP yields increasing validation loss under depth scaling.** This table reports the numerical results corresponding to Figure 8.

$L/\log_2(\eta_{\text{base}})$	-15	-14	-13	-12	-11
4	5.967	5.61	5.235	5.249	5.314
8	5.902	5.547	5.189	4.999	5.315
16	5.881	5.541	5.226	5.263	5.76
32	5.894	5.58	5.267	5.525	5.737
64	5.922	5.65	5.402	6.047	6.29
128	6.041	5.731	5.492	5.95	6.389
256	6.087	5.822	6.892	6.498	6.51

Table 39: **For Sophia, μP from Condition 3.1 ($k \geq 2$) approximately transfers the optimal base learning rate across depths, and achieves lower loss than SP as the depth increases.** This table reports the numerical results corresponding to Figure 8, where the best validation loss for each depth is highlighted in **bold**.

$L/\log_2(\eta_{\text{base}})$	-15	-14	-13	-12	-11
4	5.966	5.603	**5.242**	5.319	5.375
8	5.895	5.544	5.178	**4.935**	5.225
16	5.861	5.532	5.19	**4.994**	5.362
32	5.812	5.497	5.162	**4.906**	5.237
64	5.837	5.527	5.238	**5.154**	5.402
128	5.861	5.599	**5.329**	5.384	5.915
256	5.841	5.537	**5.271**	5.702	5.709

Table 40: **For Sophia, μP from Condition B.1 ($k = 1$) fails to transfer the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 9, where the best validation loss for each depth is highlighted in **bold**. The implementation can be found in Table 12.

$L/\log_2(\eta_{\text{base}})$	-15	-14	-13	-12	-11	-10	-9
4	5.969	5.609	**5.234**	5.329	5.356	5.938	6.264
8	5.966	5.612	5.231	**4.936**	5.212	5.515	6.252
16	5.99	5.67	5.302	5.052	**4.939**	5.438	6.061
32	6.028	5.73	5.402	5.095	5.026	**5.014**	5.97
64	6.051	5.785	5.515	5.231	**5.072**	5.077	5.218
128	6.066	5.83	5.609	5.39	5.252	**5.118**	5.359
256	6.078	5.845	5.663	5.5	5.434	**5.403**	6.009

F Justification of Upper Bound Estimation

In the derivation in Section 3, we rely on the assumption that the subadditivity and submultiplicativity inequalities used throughout the analysis are tight under standard neural network initialization and training dynamics. Under this assumption, controlling the upper bounds of $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ and $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ is sufficient to characterize the typical scaling behavior of $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ and $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ themselves, up to constant factors. In this section, we provide a more concrete justification for the validity of this assumption.

F.1 Subadditivity Inequalities

Subadditivity inequalities are used in the derivation of the update conditions to control the norm of the accumulated feature update. For instance, by decomposing $\Delta\boldsymbol{h}_s(\boldsymbol{x})$ into several layerwise contributions, we obtain

$$\begin{aligned}\Delta\boldsymbol{h}_s(\boldsymbol{x}) = \Delta\boldsymbol{h}_0(\boldsymbol{x}) &+ \underbrace{\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})}_{\boldsymbol{\epsilon}_0(s)} + \underbrace{\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\Delta\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(1)}(s)} \\ &+ \underbrace{\sum_{l=1}^{s}\alpha_l\Delta\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(2)}(s)} + \underbrace{\sum_{l=1}^{s}\alpha_l\Delta\boldsymbol{W}_l^{(2)}\Delta\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_2(s)} \qquad (26)\end{aligned}$$

which leads to the upper bound in Equation (5):

$$\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} \le \|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(2)}(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_2(L)\|_{\mathrm{R}}.$$

A similar subadditivity argument is further applied to each term, e.g.,

$$\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} \le \sum_{l=1}^{L}\alpha_l\big\|\boldsymbol{W}_l^{(2)}\Delta\boldsymbol{W}_l^{(1)}\big(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\big)\big\|_{\mathrm{R}}.$$

In principle, such subadditivity bounds may be loose when the summands point in largely different or canceling directions. However, due to the *chain rule in backpropagation, the parameter updates* $\{\Delta\boldsymbol{W}_l\}_{l=1}^{L}$ *across different layers are strongly correlated* (e.g., see discussion in Dey et al. (2025); Yang et al. (2024)). More precisely, each $\Delta\boldsymbol{W}_l$ is proportional to the product of a forward feature $\boldsymbol{h}_{l-1}(\boldsymbol{x})$ and a backpropagated error signal, which itself is obtained by repeatedly multiplying upstream Jacobians. As a result, the layerwise update contributions to $\Delta\boldsymbol{h}_L(\boldsymbol{x})$ share similar directions in feature space rather than behaving as independent or adversarial vectors.

Consequently, the terms appearing in the sums defining $\boldsymbol{\epsilon}_0(L)$, $\boldsymbol{\epsilon}_1^{(1)}(L)$, and $\boldsymbol{\epsilon}_1^{(2)}(L)$ tend to be positively aligned, and cancellations between different layers are atypical (Dey et al., 2025; Yang et al., 2024). In this regime, the norm of the sum scales proportionally to the sum of the norms, implying that the subadditivity inequality provides an accurate characterization of the magnitude of $\Delta\boldsymbol{h}_L(\boldsymbol{x})$ up to constant factors. Therefore, under standard training dynamics, controlling the subadditive upper bounds suffices to capture the typical scaling behavior of the feature updates.

F.2 Submultiplicativity Inequalities

Submultiplicativity inequalities are extensively used in the analysis of both the initial condition and the update condition. In this section, we discuss these two scenarios separately and clarify why the resulting upper bounds are typically tight under standard neural network initialization and training dynamics. Our reasoning is closely aligned with that of Yang et al. (2023), which employs a similar perspective in deriving spectral conditions for width scaling.

F.2.1 Initalization Condition

In the derivation of the initialization conditions, submultiplicativity inequalities are applied to the input, hidden, and output layers. For the input and output layers, the analysis is the same as for the width-scaling setting, since each involves a single linear transformation (e.g., we used $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} \leq \alpha_0 \|\boldsymbol{W}_0\|_{\mathrm{R}} \|\boldsymbol{x}\|_{\mathrm{R}}$). Accordingly, the tightness of the corresponding bounds directly follows from Claim 1 in Yang et al. (2023). In contrast, the hidden layers in our setting require additional justification, since each residual block consists of two or more stacked linear transformations rather than a single mapping (e.g., we used $\|\alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \leq \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}$). In what follows, we therefore focus on establishing the tightness of the submultiplicativity bounds for these multi-layer residual blocks.

Claim F.1 (Alignment of initial weight matrices)**.** *Fix a feature vector $\boldsymbol{h}_{l-1}(\boldsymbol{x}) \in \mathbb{R}^n$. Recall that $\boldsymbol{W}_l^{(1)} \in \mathbb{R}^{n_l \times n}, \boldsymbol{W}_l^{(2)} \in \mathbb{R}^{n \times n_l}$ are initialized with $\left(\boldsymbol{W}_l^{(1)}\right)_{ij}, \left(\boldsymbol{W}_l^{(2)}\right)_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_l^2)$. Provided that $n_l = \Theta(n)$, then with high probability:*

$$\|\boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\Big(\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\Big),$$

which means that the submultiplicativity inequalities used in the initialization regime are tight.

Proof. We first consider the intermediate feature $\boldsymbol{z}_l := \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) \in \mathbb{R}^{n_l}$. Since $\boldsymbol{W}_l^{(1)}$ has i.i.d. Gaussian entries with zero mean and variance σ_l^2, by the law of large numbers, we have

$$\|\boldsymbol{z}_l\|_{\mathrm{R}} = \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \approx \sigma_l \sqrt{n} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}.$$

In the meanwhile, by the standard concentration inequalities for random matrices (Vershynin, 2018) we have, with high probability that $\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} = \sqrt{n/n_l} \cdot \sigma_l(\sqrt{n} + \sqrt{n_l}) = \Theta(\sigma_l \sqrt{n})$. Therefore, we obtain

$$\|\boldsymbol{z}_l\|_{\mathrm{R}} = \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}). \tag{27}$$

Next, we apply $\boldsymbol{W}_l^{(2)}$ to $\boldsymbol{z}_l$. Again, $\boldsymbol{W}_l^{(2)}$ is an i.i.d. Gaussian matrix with variance σ_l^2, so by the law of large numbers, we have

$$\|\boldsymbol{W}_l^{(2)} \boldsymbol{z}_l\|_{\mathrm{R}} \approx \sigma_l \sqrt{n_l} \|\boldsymbol{z}_l\|_{\mathrm{R}}.$$

As well, by the standard concentration inequalities for random matrices (Vershynin, 2018) we have, with high probability that $\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \sqrt{n_l/n} = \sigma_l(\sqrt{n} + \sqrt{n_l}) = \Theta(\sigma_l \sqrt{n_l})$. Therefore, we obtain

$$\|\boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \|\boldsymbol{W}_l^{(2)} \boldsymbol{z}_l\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{z}_l\|_{\mathrm{R}}) = \Theta(\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}),$$

which shows that the submultiplicativity $\|\alpha_l \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \leq \alpha_l \|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}$ used to derive the initial condition is tight. □

F.2.2 Update Condition

We now justify the use of submultiplicativity inequalities in the update regime and argue that the resulting upper bounds on $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|$ are tight in terms of scaling. For the input and output layers, the analysis is identical to that in the width-scaling regime. As a consequence, the tightness of the corresponding bounds follows directly from Claim 2 in Yang et al. (2023). In contrast, the hidden layers in our setting require additional justification, as each residual block consists of multiple stacked linear transformations and gives rise to a more involved update structure due to the presence of residual connections. Analogous to Claim 2 in Yang et al. (2023), we therefore begin by establishing the following observation.

Claim F.2 (Alignment of updates)**.** *For any $l \in [L]$, an update $\Delta \boldsymbol{W}_l^{(2)}$ given by gradient descent with batch size 1, we have*

$$\|\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right).$$

Proof. By the chain rule, we can write $\Delta \boldsymbol{W}_l^{(2)}$ as

$$\Delta \boldsymbol{W}_l^{(2)} = -\eta_l^{(2)} \nabla_{\boldsymbol{h}_l(\boldsymbol{x})} \mathcal{L} \cdot (\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}))^\top,$$

which is rank-one and aligns with the incoming feature. Therefore, we have

$$\begin{aligned}
\|\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} &= \|\eta_l^{(2)} \nabla_{\boldsymbol{h}_l(\boldsymbol{x})} \mathcal{L} \cdot (\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}))^\top \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \\
&= \eta_l^{(2)} \|\nabla_{\boldsymbol{h}_l(\boldsymbol{x})} \mathcal{L}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_2^2 \\
&= \sqrt{\frac{n_l}{n}} \cdot \eta_l^{(2)} \sqrt{n} \|\nabla_{\boldsymbol{h}_l(\boldsymbol{x})} \mathcal{L}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_2 \cdot \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \\
&= \sqrt{\frac{n_l}{n}} \cdot \eta_l^{(2)} \|\nabla_{\boldsymbol{h}_l(\boldsymbol{x})} \mathcal{L}\|_2 \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_2 \cdot \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \\
&= \sqrt{\frac{n_l}{n}} \cdot \|\Delta \boldsymbol{W}_l^{(2)}\|_2 \cdot \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} \\
&= \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \cdot \|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}.
\end{aligned}$$

Furthermore, by the initial alignment $\|\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}})$ in Equation (27), we obtain

$$\|\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right),$$

which completes the proof.

□

Based on Claim F.2, we demonstrate how the tightness of such submultiplicativity inequality directly leads to a tight upper bound on $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ in terms of scaling. In particular, the claim ensures that the norms of the layerwise update contributions are accurately captured by their submultiplicative estimates, so that summing these bounds yields an upper bound that faithfully reflects the true magnitude of the accumulated feature update. We can rewrite the expression of the hidden layer update in Equation (26) as

$$\Delta \boldsymbol{h}_L(\boldsymbol{x}) = \sum_{l=1}^{L} \alpha_l \Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \cdots$$

Therefore, as long as the term $\sum_{l=1}^{s} \alpha_l \Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})$ does not perfectly cancel with other terms, we have

$$\begin{aligned}
\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} &= \Omega\left(\|\sum_{l=1}^{L} \alpha_l \Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right) = \Omega\left(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right) \\
&= \Omega\left(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}\right) \\
&= \Omega(1),
\end{aligned}$$

where the second equality uses the tightness of subadditivity inequalities under the principle in Appendix F.1. Therefore, the estimation of $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ by using submultiplicativity inequalities in Section 3 is tight.

G Extension to General Training Settings

As derived in the main text, our theoretical framework primarily investigates a simplified scenario: a *one-step* update of a *linear* residual MLP on a *single* datapoint. In this section, we discuss its extension to the general practical setting: *finite multi-step* updates of a *non-linear* residual MLP on a *finite batch* of datapoints.

This extension relies on three key assumptions, as those justified in the width-scaling literature (Yang et al., 2023). Below, we formally restate these assumptions and empirically verify their validity in the width-depth scaling context using the experimental setup detailed in Appendix G.2.

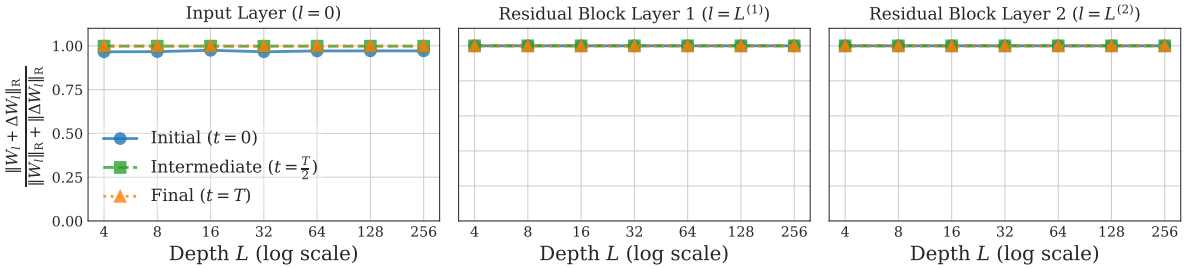

Figure 10: **Validation of Assumption G.1 (weight update).** During the training, the ratio $\frac{\|\boldsymbol{W}_l+\Delta\boldsymbol{W}_l\|_{\mathrm{R}}}{\|\boldsymbol{W}_l\|_{\mathrm{R}}+\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}}}$ remains constant near 1 across depth for the input layer and residual block layers, showing non-vanishing updates throughout multiple-step training.

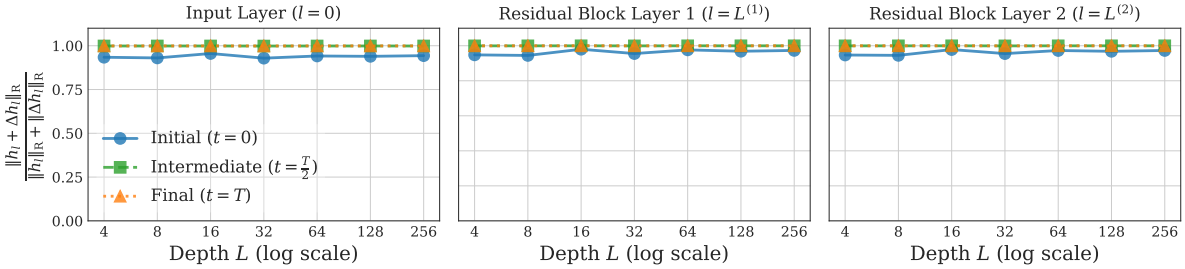

Figure 11: **Validation of Assumption G.1 (feature update).** During the training, the ratio $\frac{\|\boldsymbol{h}_l+\Delta\boldsymbol{h}_l\|_{\mathrm{R}}}{\|\boldsymbol{h}_l\|_{\mathrm{R}}+\|\Delta\boldsymbol{h}_l\|_{\mathrm{R}}}$ remains around constant near 1 across varying depths, showing non-vanishing updates throughout multiple-step training.

G.1 Assumptions for Extensions

Multi-step Training. In the main text, we derived a parameterization that ensures the weight matrices $\boldsymbol{W}_l$ and their first-step updates $\Delta\boldsymbol{W}_l$ (also, $\|\boldsymbol{h}_l\|_{\mathrm{R}}$ and $\|\Delta\boldsymbol{h}_l\|_{\mathrm{R}}$) scale correctly with the model size to achieve feature learning. To ensure these properties hold throughout *finite multi-step* training, the updated parameters must maintain the same scaling order as the first step. This is formalized in Assumption G.1.

Assumption G.1 (Non-vanishing update)**.** During finite-step training, we assume the updated weights and feature vectors for any layer l satisfy:

$$\|\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{W}_l\|_{\mathrm{R}} + \|\Delta\boldsymbol{W}_l\|_{\mathrm{R}}\right),$$
$$\|\boldsymbol{h}_l(\boldsymbol{x}) + \Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} + \|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}\right).$$

In Assumption G.1, the upper bound of the orders of the left-hand side by the right-hand side (or say, $O(\cdot)$) is guaranteed by the subadditivity. The core constraint is the lower bound of the orders (or say $\Omega(\cdot)$), which implies that the update $\Delta\boldsymbol{W}_l$ does not destructively cancel out the existing weight $\boldsymbol{W}_l$ (i.e., the update does not cause the norm to vanish). As discussed in Yang et al. (2023), such exact cancellation is extremely rare in practical neural network training. We empirically verify this assumption in Figures 10 and 11, where the norm ratios remain constant across varying depths.

Non-linearity. To extend the analysis to non-linear activations, we substitute the linear transformation $\boldsymbol{W}_l\boldsymbol{h}_{l-1}(\boldsymbol{x})$ with $\phi\left(\boldsymbol{W}_l\boldsymbol{h}_{l-1}(\boldsymbol{x})\right)$, where $\phi(\cdot)$ is an activation function (e.g., ReLU). The resulting architecture is as in Equation (28) versus Equation (2) in the linear case. We assume that the activation function preserves the asymptotic order of the feature norms, ensuring that the scaling properties derived for the pre-activations remain valid for the post-activations.

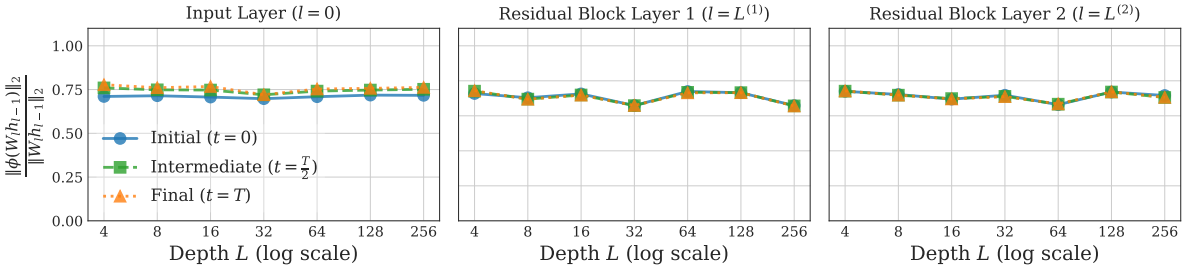

Figure 12: **Validation of Assumption G.2 (stable activation).** During the training, the ratio of post-activation to pre-activation norms $\frac{\|\phi(\boldsymbol{W}_l \boldsymbol{h}_{l-1})\|_{\mathrm{R}}}{\|\boldsymbol{W}_l \boldsymbol{h}_{l-1}\|_{\mathrm{R}}}$ remains stable across varying depths, confirming that the ReLU activation does not collapse the norm in non-linear networks.

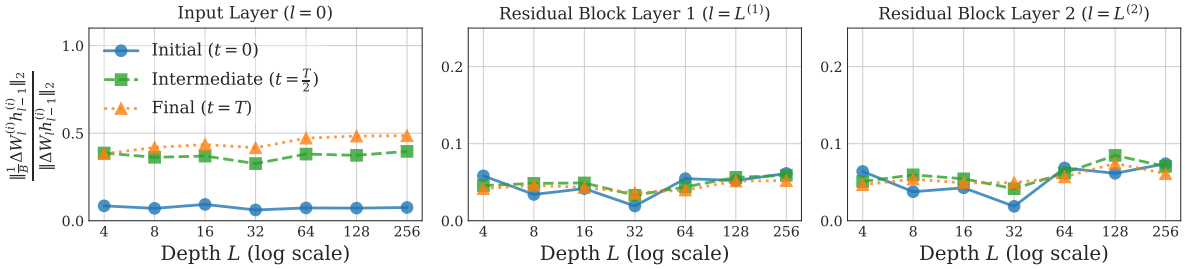

Figure 13: **Validation of Assumption G.3 (per-sample update alignment).** We report the averaged ratio of $\frac{\|\frac{1}{B}\Delta \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}^{(i)}\|_{\mathrm{R}}}{\|\Delta \boldsymbol{W}_l \boldsymbol{h}_{l-1}^{(i)}\|_{\mathrm{R}}}$ across the batch of data. The values remain stable across varying depths, suggesting that the batch update does not alter the depth-wise scaling of the single-sample update.

Assumption G.2 (Stable activation)**.** During the training, we assume that the features before and after the nonlinear activation are of the same scale:

$$\|\phi(\boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}))\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}).$$

Figure 12 empirically verifies this assumption for the ReLU activation, showing that the ratio of post-activation to pre-activation norms is stable across depth.

Training with Mini-batch. Finally, to extend beyond the single-sample setting, we consider updates computed on a batch of data $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{B}$. Let $\Delta \boldsymbol{W}_l^{(i)}$ denote the update contribution from the i-th datapoint (e.g., $\Delta \boldsymbol{W}_l^{(i)} = -\eta_l \nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ for SGD), such that the total batch update is $\Delta \boldsymbol{W}_l = \frac{1}{B} \sum_{i=1}^{B} \Delta \boldsymbol{W}_l^{(i)}$. We expect that the gradient contributions from different samples do not destructively cancel out. This is formalized in Assumption G.3.

Assumption G.3 (Per-sample update alignment)**.** We assume that the batch size satisfies $B = \Theta(1)$, and the batch update norm scales consistently with the per-sample update norm during the training:

$$\|\Delta \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}^{(i)})\|_{\mathrm{R}} = \Theta\left(\frac{1}{B} \|\Delta \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x}^{(i)})\|_{\mathrm{R}}\right).$$

We verify this in Figure 13, where the alignment ratio remains $\Theta(1)$, indicating that batch-averaged updates preserve the scaling properties of single-sample updates.

G.2 Experimental Details

We conduct simulations to empirically Assumptions G.1–G.3. Our experimental setup largely follows Yang et al. (2023), with a different emphasis on depth scaling instead of width scaling. Details are provided below.

Dataset. We construct a binary classification dataset using a subset of CIFAR-10, selecting 100 samples each from the "airplane" and "automobile" classes. The inputs are flattened image vectors in $\mathbb{R}^{3072}$ associated with binary labels in $\{0, 1\}$.

Architecture and Training. The architecture is a deep residual MLP with ReLU activations, consisting of an input layer, L residual blocks, and a final linear output layer:

$$\begin{aligned}
&\boldsymbol{h}_0(\boldsymbol{x}) = \alpha_0 \phi\left(\boldsymbol{W}_0 \boldsymbol{x}\right), \\
&\boldsymbol{h}_l(\boldsymbol{x}) = \boldsymbol{h}_{l-1}(\boldsymbol{x}) + \alpha_l \phi\left(\boldsymbol{W}_l^{(2)} \phi\left(\boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\right)\right), \quad l \in [L], \\
&\boldsymbol{h}_{L+1}(\boldsymbol{x}) = \alpha_{L+1} \boldsymbol{W}_{L+1} \boldsymbol{h}_L(\boldsymbol{x}),
\end{aligned} \tag{28}$$

where ϕ denotes the ReLU activation. The dimensions are set as: input dimension $d_0 = 3072$, model width $n = 256$, residual block width $n_l = n$, and output dimension $d_{L+1} = 1$. This aligns well with the simplified setup discussed in the main text (Section 3.1). Models are trained to minimize the binary cross-entropy loss using full-batch Gradient Descent (GD) for $T = 200$ steps.

Parameterization. We implement the width-depth μP parametrization for SGD derived in Table 5 of Appendix C.5 as follows:

$$\begin{aligned}
&\alpha_0 = \alpha_{\text{base}}, \quad \alpha_l = \frac{\alpha_{\text{base}}}{L}, \quad \alpha_{L+1} = \frac{\alpha_{\text{base}}}{n}, \\
&\sigma_0^2 = \frac{\sigma_{\text{base}}^2}{d_0}, \quad \sigma_l^2 = \frac{\sigma_{\text{base}}^2}{n}, \quad \sigma_{L+1}^2 = \sigma_{\text{base}}^2, \\
&\eta_0 = \eta_{\text{base}} n, \quad \eta_l = \eta_{\text{base}} L, \quad \eta_{L+1} = \eta_{\text{base}} n,
\end{aligned}$$

with base constants set to:

$$\alpha_{\text{base}} = 1, \quad \sigma_{\text{base}}^2 = 2, \quad \eta_{\text{base}} = 0.001.$$

Verification of Assumptions. We perform a depth scaling analysis by training networks with depths $L \in \{4, 8, 16, 32, 64, 128, 256\}$. We track the metrics corresponding to the assumptions above at three distinct training phases: initialization ($t = 0$), intermediate training ($t = T/2$), and the end of training ($t = T$), and different layers: the input layer ($l = 0$) and the internal layers of the final residual block (here, we denote them by $l = L^{(1)}$ and $l = L^{(2)}$), as representatives.