SPECTRAL CONDITION FOR μP UNDER WIDTH–DEPTH SCALING

Chenyu Zheng[1,2,3*], **Rongzhen Wang**[1,2,3], **Xinyu Zhang**[4], **Chongxuan Li**[1,2,3†]
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] Beijing Key Laboratory of Research on Large Models and Intelligent Governance
[3] Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE
[4] ByteDance Seed {cyzheng,wangrz,chongxuanli}@ruc.edu.cn

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. Considering residual networks of varying block depths, we first introduce a spectral μP condition that precisely characterizes how the norms of weights and their per-step updates should scale with width and depth, unifying previously disparate μP formulations as special cases. Building on this condition, we then derive a general recipe for implementing μP across a broad class of optimizers by mapping the spectral constraints to concrete HP parameterizations. This approach not only recovers existing μP formulations (e.g., for SGD and AdamW) but also naturally extends to a wider range of optimizers. Finally, experiments on GPT-2 style language models demonstrate that the proposed spectral μP condition preserves stable feature learning and enables robust HP transfer under width–depth scaling. Our code is available at *GitHub*.

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 across model scales.

Maximal update parameterization (μP) (Yang & Hu, 2021) was originally proposed to address both challenges for width scaling (Yang et al., 2022), 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). By appropriately reparameterizing HPs with model size, μP principle desires to preserve scale-invariant feature learning, while maximizing the feature change induced by parameter updates, leading to 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 and often tightly coupled to specific architectures (Yang et al., 2024; Bordelon et al., 2024b;a; Dey et al.,

*Work done during an internship at ByteDance Seed.
†Correspondence to Chongxuan Li <chongxuanli@ruc.edu.cn>.

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 optimizers and architectures, 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, which leads to a simple and unified framework for characterizing μP in deep residual networks and for systematically deriving μP formulations across a broad class of optimizers. Our main contributions are summarized as follows.

First, we introduce a unified spectral scaling condition (Condition 3.1) that characterizes the μP principle for residual networks under width-depth scaling, which specifies how the RMS operator norms of weights and their per-step updates should scale with the model size. By analyzing residual blocks with varying depths, we clarify how deeper residual blocks impose stricter spectral constraints, and show that previously disparate μP formulations (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025; Qiu et al., 2025) arise as special cases of the unified spectral framework. Notably, unlike prior derivations based on more complicated techniques, our analysis relies only on elementary linear algebra and probability, so it is much easier to follow.

Second, building on the proposed spectral condition, we present a unified recipe for implementing μP across a broad class of optimizers by directly mapping the condition to concrete HP parameterizations. Our framework recovers existing μP formulations under joint width-depth scaling as special cases, such as those for SGD (Bordelon et al., 2024a), AdamW (Dey et al., 2025), and matrix-preconditioned optimizers (Qiu et al., 2025). Moreover, we systematically extend the μP principle to a wider range of modern optimizers, including Muon-Kimi (Liu et al., 2025), Spectral Sphere Optimizer (SSO) (Xie et al., 2026), Sophia (Liu et al., 2024b), and Lion (Chen et al., 2023), yielding practical and theoretically grounded μP formulations derived from their update rules, rather than ad hoc tuning heuristics.

Finally, through controlled experiments on GPT-2 style language models (Radford et al., 2019; Karpathy, 2022) trained by Muon-Kimi, we empirically demonstrate that the μP formulation derived from the proposed spectral condition enables scale-invariant feature learning and robust HP transfer under joint width–depth scaling. These results validate the practical effectiveness of the proposed spectral framework.

2 PRELIMINARIES

We begin by establishing the necessary background for mathematical techniques and μP. Detailed discussion of additional related work is placed in Appendix A.

2.1 MATHEMATICAL NOTATIONS AND PROPERTIES

We define $[n] = \{1, 2, \dots, n\}$. 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:** $\|\boldsymbol{A}\boldsymbol{B}\|_{\mathrm{R}} \leq \|\boldsymbol{A}\|_{\mathrm{R}}\|\boldsymbol{B}\|_{\mathrm{R}}$ and $\|\boldsymbol{A}\boldsymbol{v}\|_{\mathrm{R}} \leq \|\boldsymbol{A}\|_{\mathrm{R}}\|\boldsymbol{v}\|_{\mathrm{R}}$.
- **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 multilayer 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 scaling behavior of feature learning (Yang et al., 2023). Moreover, μP formulations derived under this setup can be directly applied to 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. 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 principle (Yang & Hu, 2021).

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

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

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

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 characterization of 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).

Current Limitation. 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 present the spectral μP condition and discuss its implications.

3.1 PROBLEM SETUP

Our setup mainly follows Section 2.2 under width scaling, with the key difference being the introduction of residual connections, which are essential for stabilizing deep network training (He et al., 2016). Since practical residual blocks often comprise multiple transformations (e.g., attention or FFN module in Transformers), we study residual blocks with a multi-layer main branch. To keep the analysis minimal while capturing core behaviour, we focus on the two-layer linear block in the main text; extensions to arbitrary fixed residual block depth are deferred to Appendix B.3. Formally, the network 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 sections, we try to seek a sufficient and unified spectral condition to satisfy the μP Principle 2.1 under joint width-depth scaling.

3.2 Spectral Scaling Condition

Analogous to Condition (1) under width scaling, our spectral condition also consists of two components. The initial condition on $\boldsymbol{W}_l$ suffices controlled feature propagation, yielding $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, while the update condition on $\Delta\boldsymbol{W}_l$ guarantees that feature changes induced by a single optimization step remain $\|\Delta\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ and maximally updated according to Principle (P2). We now formally state the spectral μP condition for width-depth scaling.

Condition 3.1 (Spectral condition for μP under joint width-depth scaling, proof in Appendix B.1)**.** *To ensure 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.***

  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):

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

In contrast to the width-only scaling (cf. Condition (1)), our spectral condition shows that the RMS operator norm of hidden weights and their updates should shrink with depth as $\Theta(L^{-1})$ to prevent feature explosion caused by accumulation along the residual connections.

To the best of our knowledge, prior important but disparate μP results (Yang et al., 2024; Bordelon et al., 2024a;b; Dey et al., 2025; Qiu et al., 2025) under width-depth scaling can be unified within our

[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.

spectral framework by varying the residual block depth. When the block depth is 1, the same derivation for Condition 3.1 yields the corresponding spectral result in Condition B.1 of Appendix B.2. In this case, the absence of second-order update constraints (C2.3) leads to a looser condition, which naturally induces residual multipliers $\alpha_l = \Theta(1/\sqrt{L})$ (see Appendix B.2 for details), thus recovering the early results in Yang et al. (2024); Bordelon et al. (2024b). When the block depth is 2, the additional second-order condition leads to fundamentally different and stronger residual multipliers $\alpha_l = \Theta(1/L)$ (see Section 4 and Appendix C for detailed HP parameterizations), thereby revisiting the recent results in Bordelon et al. (2024a); Dey et al. (2025); Qiu et al. (2025).

Moreover, our analysis naturally extends to residual blocks with any fixed depth k (Condition B.2 in Appendix B.3) and to architectures with biases (Condition B.3 in Appendix B.4). For residual blocks of depth k, the initialization requires the product of the α_l and the norms of the k hidden weights to scale as $\Theta(1/L)$, while 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.3 and B.4, these additional constraints do not lead to different HP parameterizations compared to the two-layer case. Therefore, the two-layer residual block in Equation (2) constitutes the minimal setting that captures the core μP behavior under joint width-depth scaling.

Though Condition 3.1 is derived under a linear residual MLP with one-step update, it generalizes naturally to more general and practical training regimes. Theoretically, we introduce and verify some mild assumptions (Yang et al., 2023) in Appendix F, under which the spectral results generalize to multiple gradient steps, nonlinearities, and 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 indicate that our setup captures the core scaling behavior in practice.

4 Implementation of Spectral Condition

In this section, our goal is to determine proper parameterizations of block multiplier α_l, initial variance σ_l^2, and learning rate η_l to satisfy Condition 3.1 for Muon-Kimi. Discussions of additional HPs (e.g., weight decay) and optimizers are deferred to Appendix C.

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 initial variance to the standard width-scaling μP algorithm (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:

$$\|\boldsymbol{W}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\boldsymbol{W}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \cdot \Theta\left(\sigma_l(\sqrt{n_{\text{in}}} + \sqrt{n_{\text{out}}})\right) = \begin{cases} \Theta(1), & 0 \le l \le L, \\ \Theta(n_{\text{in}}), & l = L+1, \end{cases} \tag{4}$$

where we used the spectral norm property of random matrices reviewed in Section 2. Based on Equation (4), 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

$$\alpha_0 = \Theta(1), \quad \alpha_{L+1} = \Theta(1/n_{\text{in}}). \tag{5}$$

For the hidden layers, given

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

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

$$\alpha_l = \Theta(1/L),\ l \in [L]. \tag{6}$$

4.2 Update Condition for Muon-Kimi

Since different optimizers take different scales of $\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}}$, the implementation of the update condition depends on the choice of optimizer. In the main text, we focus on the Muon-Kimi (Liu et al., 2025). Other optimizers: Muon (Jordan et al., 2024), SGD, AdamW (Loshchilov & Hutter, 2019), Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2024), SSO (Xie et al., 2026), Lion (Chen et al., 2023), and Sophia (Liu et al., 2024b) 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., 2024) 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 $\boldsymbol{W}_l \in \mathbb{R}^{n_{\mathrm{out}} \times n_{\mathrm{in}}}$, the update rule (without weight decay) is:

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

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

$$\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \sqrt{\frac{n_{\mathrm{in}}}{n_{\mathrm{out}}}} \|\Delta \boldsymbol{W}_l\|_2 = \Theta\left(\eta_l \sqrt{n_{\mathrm{in}}} \max\left\{1, \sqrt{\frac{n_{\mathrm{in}}}{n_{\mathrm{out}}}}\right\}\right) = \begin{cases} \Theta(\eta_l), & l = 0, \\ \Theta(\eta_l \sqrt{n_{\mathrm{in}}}), & l \in [L], \\ \Theta(\eta_l n_{\mathrm{in}}), & l = L+1. \end{cases} \tag{7}$$

Based on Equation (7), 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 (5), we have:

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

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 (4), and the α_l parameterization in Equation (6), we have:

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

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

$$\eta_l^{(2)} = \Theta\left(\frac{1}{\sqrt{n_{\mathrm{in}}}}\right),\ l \in [L]. \tag{8}$$

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

We note that the second-order update condition (C2.3) is automatically satisfied once the initial condition (C1.2) and the first-order update condition (C2.2) are met, as discussed in Appendix B.1.3, so no further constraint is needed for implementing the second-order condition (C2.3).

Until now, we have obtained the proper parameterization of the learning rate η_l for Muon-Kimi.

Table 1: **μP implementation 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}

4.3 Practical HP Parameterization and Transfer

In practice, μP is often implemented using a ratio-based approach Yang et al. (2022); 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 (8). Table 1 summarizes the complete HP parameterization for Muon-Kimi under width-depth scaling derived in former sections.

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 HP parameterization in Table 1 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 verify that the μP formulation derived from our spectral condition (Condition 3.1) enables scale-invariant feature learning and robust HP transfer across model scales.

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 dimension of $4n$. According to common practice (Jordan et al., 2024; Liu et al., 2025), hidden matrix parameters are optimized by Muon-Kimi with a Nesterov-style momentum (Nesterov, 1983) of 0.95. While all other parameters (e.g., all biases, embedding layer) are updated by AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-16}$. We do not use weight decay in all experiments.

We define a base model $(n_{\text{base}}, L_{\text{base}}) = (256, 4)$ and scale HPs according to the μP Condition 3.1, with concrete HP parameterizations summarized in Table 1 for Muon-Kimi and Table 5 in Appendix C for AdamW. We refer the reader to Appendix D for detailed experimental configurations.

5.2 Feature Learning and HP Transfer

In this section, we compare the feature learning stability and HP transferability of SP and the proposed μP formulation. The main results are shown in Figure 1, with complete results deferred to Appendix D.

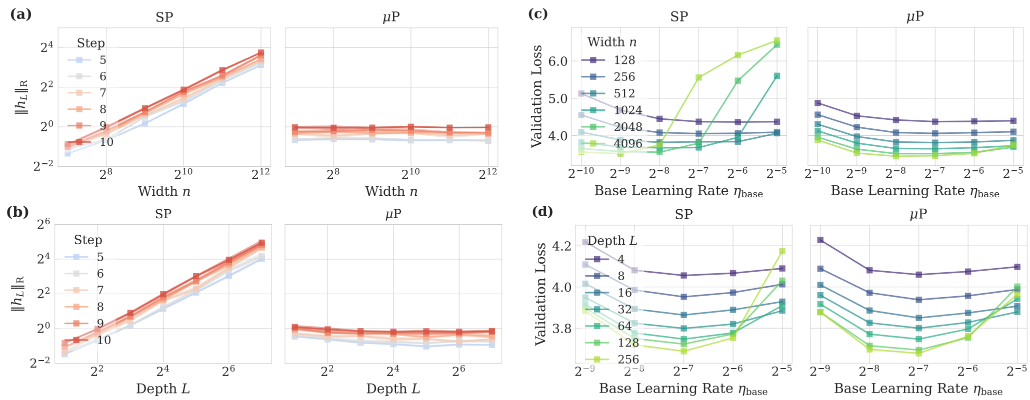

Figure 1: **Feature learning and HP transfer under SP and μP.** We train GPT-2 style Transformer language models with Muon-Kimi and AdamW using SP and the width-depth μP derived in Tables 1 and 5. μP maintains stable feature norms and enables robust HP transfer across both width and depth scaling, while consistently achieving lower loss than SP as the width and depth increase.

Feature learning. We first examine the stability of feature learning 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 the 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 the μP Principle 2.1. This confirms that the proposed μP formulation 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, indicating poor HP transfer. In contrast, μP preserves a nearly invariant optimal learning rate across both width and depth scaling (Figure 1(c,d)). Such stable HP transferability can significantly reduce tuning cost when scaling model size, particularly for pretraining large models (Singh et al., 2025; Team et al., 2025; Nie et al., 2025). Moreover, we note that μP consistently achieves lower loss as the width and depth increase.

Discussion. One may notice that under SP the optimal learning rate appears to transfer reasonably well along the depths in our experiments. 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 experiments in Appendix D. The results (please see Figure 2 in Appendix D.3) show that *SP training becomes unstable and depth-wise HP transfer breaks down, while μP remains stable even at large depths (up to $L = 256$) and continues to exhibit robust HP transfer*.

6 Conclusion

In this paper, we present a simple and unified spectral framework for μP under joint width–depth scaling. Our spectral μP condition precisely characterizes the scaling of weights and their updates, enabling a general recipe for HP choices across a broad class of optimizers. Empirical results on GPT-2 style language models validate that our approach preserves scale-invariant feature learning and facilitates robust HP transfer, offering a simple and principled solution for efficient scaling of generative foundation models.

ACKNOWLEDGMENTS

This work was supported by the Beijing Major Science and Technology Project under Contract No. Z251100008425002; the National Natural Science Foundation of China (Nos. 62522609, 92470118); the Beijing Natural Science Foundation (No. L247030); the ByteDance Seed Research Fund; and the fund for building world-class universities (disciplines) of Renmin University of China. Chenyu Zheng was also supported by the National Natural Science Foundation of China (No. 625B2177) and the Outstanding Innovative Talents Cultivation Funded Programs 2025 of Renmin University of China.

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, 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, 2024.

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.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *CoRR*, abs/2502.09992, 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 scaling (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 that characterizes 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 residual multiplier of order $\Theta(1/\sqrt{L})$ suffices to preserve stable feature learning, but observes 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 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 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 Transformer (e.g., GPT-2). This method (Bordelon et al., 2024a; Dey et al., 2025) is further extended to matrix-preconditioned optimizers such as Muon and SOAP (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 present a detailed derivation of the spectral condition for general residual networks, extending and unifying existing results in the literature (Yang et al., 2024; Bordelon et al., 2024b;a; Dey et al., 2025; Qiu et al., 2025). We begin with the theoretical derivation for residual networks with two-layer residual blocks (Condition 3.1 in the main text). We then investigate the simplest case of residual networks with one-layer residual blocks, where our spectral condition recovers previously studied width-depth μP formulations (Yang et al., 2024; Bordelon et al., 2024b) and clarifies the role of block depth in determining μP condition. After that, building on analysis for the two-layer block in Appendix B.1, we further generalize the spectral condition to residual blocks with an arbitrary but finite number of layers, and show that, despite differences in block depth, these architectures admit essentially the same scaling rules from an algorithmic perspective. Finally, we extend the spectral condition to bias parameters and show that they can be incorporated in a consistent and scale-invariant manner under the same framework.

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).

B.1 Derivation for Two-layer Residual Block

In this section, we derive the spectral condition in Condition 3.1. The derivation proceeds in three steps: (i) establishing a preliminary initialization condition ensuring $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, (ii) deriving the update condition required for $\|\Delta \boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ while maximally updated according to Principle (P2), and (iii) using the update condition to refine the initial condition. Unlike prior analyses that rely on complex techniques (Yang et al., 2024; Bordelon et al., 2024a;b), our argument uses only elementary linear algebra and probability, which is easier to follow.

Throughout the analysis, we use upper bounds derived from subadditivity and submultiplicativity inequalities to characterize the scale of $\|\boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$ and $\|\Delta \boldsymbol{h}_l(\boldsymbol{x})\|_{\mathrm{R}}$; for instance, we treat $\|\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} = \alpha_0\|\boldsymbol{W}_0\boldsymbol{x}\|_{\mathrm{R}} = \Theta(\alpha_0\|\boldsymbol{W}_0\|_{\mathrm{R}}\|\boldsymbol{x}\|_{\mathrm{R}})$. We argue that such bounds are typically tight under standard neural network training, as claimed under width scaling (Yang et al., 2023), with a detailed justification in the width-depth scaling regime deferred to Appendix E.

B.1.1 Derivation for 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 hidden features' scale, we expand the residual recursion in Equation (2), which yields

$$\boldsymbol{h}_s(\boldsymbol{x}) = \boldsymbol{h}_0(\boldsymbol{x}) + \sum_{l=1}^{s} \alpha_l \boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\boldsymbol{h}_{l-1}(\boldsymbol{x}),\ s \in [L]. \tag{9}$$

Applying subadditivity, we can estimate their order as

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

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 independent zero-mean random vectors, so the RMS norm of their sum (standard deviation) scales with the square root of the sum of their squared RMS norms (variance) (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 in (C1.2), which will be further refined once update constraints are incorporated.

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 $\|\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ by initial hidden features. Thus choosing $\alpha_{L+1}\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$ keeps the output stable.

B.1.2 Derivation for 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 (9) 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

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

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

According to the degree of weight updates, 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 $\boldsymbol{\epsilon}_2(s)$. By the subadditivity of vector norms, we have

$$\|\Delta\boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\big(\|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(1)}(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(2)}(s)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_2(s)\|_{\mathrm{R}}\big). \tag{10}$$

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 next control terms on the right-hand side of Equation (10).

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 Appendix B.1.1 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 $\|\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ 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

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

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 natrually 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 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. *We note that this term will vanish when the residual block is single-layer.* Using subadditivity and submultiplicativity as for $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}}$, we can estimate its scale as

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

Principle (P2) requires maximizing each summand and ensure $\|\boldsymbol{\epsilon}_2(L)\|_{\mathrm{R}} = \Theta(1)$ 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 have

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

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

B.1.3 Derivation for 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 implies $\alpha_l\|\Delta\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}}\|\Delta\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} = \Theta(1/L)$ for all $l \in [L]$. Combining the two relations 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.

We note that, in the presence of the first-order update condition (C2.2), the above initial condition (C1.2) is equivalent to the second-order update condition (C2.3) on hidden weights. **Thus, retaining either one in Condition 3.1 would be sufficient.** Nevertheless, to more clearly emphasize the underlying μP principle, we explicitly include both in Condition 3.1.

B.2 One-layer Residual Block

B.2.1 Problem Setup

We consider a residual network with one-layer residual blocks, 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 $\mathcal{L}(\boldsymbol{h}_{L+1}(\boldsymbol{x}), \boldsymbol{y})$. As in the two-layer block case, our goal is to characterize the spectral condition that ensures μP Principle 2.1 in this regime.

B.2.2 Spectral Scaling Condition

We now state the spectral scaling condition for the above residual network with one-layer blocks that characterizes 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 ensure 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.2.4). As a result, the μP Principle 2.1 requires controlling a single class of direct update effects, 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, Bordelon et al. (2024b); Yang et al. (2024) adopt 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.2.4). The parameterization of other HPs (e.g., learning rate) in Bordelon et al. (2024b); Yang et al. (2024) can also be exactly derived from Condition B.1. Since the derivation follows the same steps as implementations in Section 4 and Appendix C, we omit it here.

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 such μP formulations in Yang et al. (2024); Bordelon et al. (2024b) do not directly extend to practical architectures (residual block with two or more layers), nor do they support robust HP transfer across depth (Yang et al., 2024; Dey et al., 2025).

B.2.3 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 independent zero-mean random vectors, so the RMS norm of their sum (standard deviation) scales with the square root of the sum of their squared RMS norms (variance) (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.2.4 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

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

where we used $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally set $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. We note that if we furhter set $\alpha_l \|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/\sqrt{L})$ for all $l \in [L]$ in inital condition, then $\|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} = \Theta(1)$ and is maximized, which leads to the μP formulations in 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 $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ if we finally set $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$, we can obtain $\|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} = \Theta\big(\sum_{l=1}^{L} \alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}}\big)$. 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 natrually 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 update satisfies

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

By subadditivity and submultiplicativity,

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

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

B.3 Multi-layer Residual Block

B.3.1 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. 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 characterize the spectral condition that ensures 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 characterization, and do not fundamentally alter the algorithmic implementation of μP.

B.3.2 Spectral Scaling Condition

We now state the spectral scaling condition for the above residual network with k-layer residual block that characterizes 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 ensure 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 satisfies

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

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.3.3 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

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

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 independent zero-mean random vectors, so the RMS norm of their sum (standard deviation) scales with the square root of the sum of their squared RMS norms (variance) (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. Finally, for the output layer, we have

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

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

B.3.4 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

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

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

$$\|\Delta \boldsymbol{h}_s(\boldsymbol{x})\|_{\mathrm{R}} = \Theta\left(\|\Delta \boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \sum_{j=0}^{k} \|\boldsymbol{\epsilon}_0(s)\|_{\mathrm{R}} \right).$$

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 used $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally enfore $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$.

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,

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

where we used $\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ by the preliminary initial condition and $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally enfore $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. 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 natrually 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 subadditivity and submultiplicativity, the j-th order update terms satisfy

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

where we used $\|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ by the preliminary initial condition and $\|\Delta \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$ for $l \in [L]$ if we finally enfore $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} = \Theta(1)$. Principle (P2) requires maximizing each summand and ensure $\|\boldsymbol{\epsilon}_j(L)\|_{\mathrm{R}} = \Theta(1)$ in the meanwhile, it suffices 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 Appendix B.1 yields $\alpha_{L+1}\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1)$.

B.3.5 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 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 3.1 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.

B.4 Bias Parameters

B.4.1 Problem Setup

As shown in Appendix B.3, residual blocks with an arbitrary fixed internal depth $k \geq 2$ admit spectral scaling conditions that are algorithmically equivalent to the two-layer case. Therefore, to simplify the presentation while retaining full generality, we focus on two-layer residual blocks with bias. 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 characterize the spectral condition that ensures μP Principle 2.1 in this setting.

B.4.2 Spectral Scaling Condition

We now state the spectral scaling condition for the residual network with biases that characterizes 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 ensure 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 HP parameterization 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.4.3 Derivation for Preliminary Initialization Condition

Input layer. By subadditivity and submultiplicativity,

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

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 independent zero-mean random vectors, so the RMS norm of their sum (standard deviation) scales with the square root of the sum of their squared RMS norms (variance) (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 argument and submultiplicativity inequality as in the derivation without biases (e.g., see Appendix B.1), 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.4.4 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

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

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 Appendix B.1). 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

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

Accoding 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 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

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

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

B.4.5 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 (Appendix B.1), 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.

B.4.6 Derivation for Efficient Implementation

Recall that, based on the HP parameterization introduced for matrix weights in Section 4 and Appendix C, we have $\alpha_0 = \Theta(1)$ in Equation (5), $\alpha_l = \Theta(1/L)$ for $l \in [L]$ in Equation (6), $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ for $0 \le l \le L$ in Equation (4) 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):
  * $\|\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.
  * $\|\Delta\boldsymbol{b}_l^{(1)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.
  * $\|\Delta\boldsymbol{b}_l^{(2)}\|_{\mathrm{R}} = \Theta(1), \quad \forall l \in [L]$.

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

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

C IMPLEMENTING SPECTRAL CONDITION FOR VARIOUS OPTIMIZERS AND HPS

Recall that in Section 4.1 of the main text, we implemented the spectral condition 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 the spectral condition 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* and incorporate *additional HPs*, including the weight decay and bias learning rate.

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 $\boldsymbol{W}_l$ (e.g., $\boldsymbol{A}_l = \nabla_{\boldsymbol{W}_l}\mathcal{L}$ for SGD, $\boldsymbol{A}_l = 0.2\sqrt{\max\{n_{\mathrm{in}}, n_{\mathrm{out}}\}}\,\boldsymbol{U}_l\boldsymbol{V}_l^\top$ for Muon-Kimi), 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 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 $\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \eta_l\|\boldsymbol{A}_l\|_{\mathrm{R}}$. In this case, we expect:

$$\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}} = \eta_l\|\boldsymbol{A}_l\|_{\mathrm{R}} \text{ satisfies Condition 3.1.} \qquad (\Delta 1)$$

With weight decay. When weight decay is enabled ($\lambda_l \neq 0$), we expect the weight decay term to be comparable in scale to the gradient-driven term so that weight decay is effective on the update dynamics:

$$\|\lambda_l\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\mathrm{R}}\right). \qquad (\Delta 2)$$

In the following contents, we will derive the 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 applied to bias parameters, we restrict the bias parameterization analysis to vector-based optimizers (such as SGD and AdamW). The bias parameters $\boldsymbol{b}_l$ follow a similar formulation as the matrix parameters presented above (Equation (Δ1) and (Δ2)).

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 norm, omitting the momentum is regarded as an acceptable simplification. Moreover, analyzing the update without momentum can be interpreted as studying the *first* update 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.

We also present a useful preliminary for the gradient norm here. It has been widely observed in practice that gradients arising during neural network training exhibit a *low-rank* structure (Yang et al., 2023; Balzano et al., 2025; Zhao et al., 2024), that is, only a small number of dominant singular directions carry most of the gradient. As a consequence, the spectral norm and the Frobenius norm of the gradient matrix are of the same order, i.e.,

$$\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_2 = \Theta(\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\mathrm{F}}), \tag{15}$$

up to constants independent of width and depth. This property will be used to estimate the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$.

C.1 Muon-Kimi (with Weight Decay)

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

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

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 (Δ1) that

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

According to the update norm in Equation (7), we have

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

Given the magnitude of $\|\boldsymbol{W}_l\|_{\mathrm{R}}$ in Equation (4), as desired by (Δ2), the parameterizations of λ_l need to be set as follows:

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

This completes the implementation of the update condition for Muon-Kimi with weight decay, as summarized in Table 2.

C.2 Muon

We recover and extend the μP formulation under width-depth scaling of Muon in Qiu et al. (2025) in this section.

C.2.1 Update Rule

For a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{n_{\mathrm{out}}\times n_{\mathrm{in}}}$, the update rule of Muon (Jordan et al., 2024) 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 $0.2\sqrt{\max\{n_{\mathrm{in}}, n_{\mathrm{out}}\}}$ prefactor.

Table 2: **μP implementation 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}

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.2.2 Derivation of Parameterization

Input and output layers. When $\lambda_0 = 0$, 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 (5), 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 $\alpha_0\|\Delta\boldsymbol{W}_0\|_{\text{R}}, \alpha_{L+1}\|\Delta\boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(1)$, 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 (4) that $\|\boldsymbol{W}_0\|_{\text{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(n_{\text{in}})$, we have

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

To satisfy (Δ2) that $\|\lambda_l\boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right)$, 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 $\lambda_l = 0$, given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ in Equation (4), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (6), and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ in Equation (17), we have

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

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

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

When $\lambda_l \neq 0$, given the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ Equation (4), we have

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

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

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

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

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

Table 3: μ**P implementation for Muon (Jordan et al., 2024), 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})

Hidden layers (second-order). As discussed in Appendix B.1.3, 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 $\alpha_l^2\|\boldsymbol{W}_l^{(2)}\|_{\text{R}}\|\boldsymbol{W}_l^{(1)}\|_{\text{R}}\,\|\Delta\boldsymbol{W}_l^{(2)}\|_{\text{R}}\,\|\Delta\boldsymbol{W}_l^{(1)}\|_{\text{R}} = \Theta(\frac{1}{L^2})$. Combining this with (C1.2) that $\alpha_l\|\boldsymbol{W}_l^{(2)}\|_{\text{R}}\,\|\boldsymbol{W}_l^{(1)}\|_{\text{R}} = \Theta(1/L)$ 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 3.

C.3 SGD

We recover and extend the μP formulation under width-depth scaling of SGD in Bordelon et al. (2024a) in this section.

C.3.1 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:

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

Here, we follow the method in Yang et al. (2023) to estimate the scale of gradient $\nabla_{\boldsymbol{W}_l}\mathcal{L}$. From the derivation of the update condition in Appendix B.1, we can observe that gradient updates $\Delta\boldsymbol{W}_0, \Delta\boldsymbol{W}_{L+1}$ induce a change $\|\Delta\boldsymbol{h}_{L+1}\| = \Theta(1)$ 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 structure of the gradient (see 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 (5), 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 $\alpha_l \|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} \|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1/L)$ to satisfy the update condition (C2.2), and use $\alpha_l = \Theta(1/L)$ in Equation (6), $\|\boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ in Equation (4) by initial implementation, we have $\|\Delta \boldsymbol{W}_l\|_{\mathrm{R}} = \Theta(1)$ so $\|\Delta \boldsymbol{W}_l\|_2 = \Theta(\sqrt{n_{\mathrm{out}}/n_{\mathrm{in}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}\|_2 = \Theta(L^{-1}\sqrt{n_{\mathrm{in}}/n_{\mathrm{out}}})$, which leads to

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

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

$$\begin{aligned}\Theta(1) = \Delta_{\boldsymbol{W}_{L+1}} \mathcal{L} &= \Theta(\langle \Delta \boldsymbol{W}_{L+1}, \nabla_{\boldsymbol{W}_{L+1}} \mathcal{L}\rangle) = \Theta(\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{F}} \|\nabla_{\boldsymbol{W}_{L+1}} \mathcal{L}\|_{\mathrm{F}}) \\ &= \Theta(\|\Delta \boldsymbol{W}_{L+1}\|_2 \|\nabla_{\boldsymbol{W}_{L+1}} \mathcal{L}\|_2),\end{aligned}$$

Since we will set $\alpha_{L+1} \|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(1/L)$ to realize the update condition (C2.1) and use $\alpha_{L+1} = \Theta(1/n_{\mathrm{in}})$ in initial implementation in Equation (5), we have $\|\Delta \boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\mathrm{in}})$ so $\|\Delta \boldsymbol{W}_{L+1}\|_2 = \Theta(\sqrt{n_{\mathrm{out}} n_{\mathrm{in}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{W}_{L+1}} \mathcal{L}\|_2 = \Theta(1/\sqrt{n_{\mathrm{in}} n_{\mathrm{out}}})$, which leads to

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

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.3.2 Derivation of Parameterization

Input and output layers. When $\lambda_0 = 0$, 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 (5), 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 inital implementation Equation (4)), 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 (6) and $\|\boldsymbol{W}_l^{(i)}\|_{\mathrm{R}} = \Theta(1)$ in Equation (4). 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 (4), 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

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

Table 4: **μP implementation 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})

Hidden layers (second-order). As illustrated in Appendix B.1.3 and Appendix C.2, the second-order update condition is satisfied automatically once the initial condition and the first-order update condition are met.

Biases. Similar to the above derivation for $\|\nabla_{\boldsymbol{W}_l}\mathcal{L}\|_{\mathrm{R}}$, we can obtain the scale of $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_{\mathrm{R}}$. For the input biases $\boldsymbol{b}_0 \in \mathbb{R}^{n_{\text{out}}\times 1}$, 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 $\|\Delta \boldsymbol{b}_0\|_2 = \Theta(\sqrt{n_{\text{out}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{b}_0}\mathcal{L}\|_2 = \Theta(1/\sqrt{n_{\text{out}}})$, which leads to

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

Similarly, for the hidden biases $\boldsymbol{b}_l \in \mathbb{R}^{n_{\text{out}}\times 1}$, 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 $\|\Delta \boldsymbol{b}_l\|_2 = \Theta(\sqrt{n_{\text{out}}})$. Therefore, we obtain $\|\nabla_{\boldsymbol{b}_l}\mathcal{L}\|_2 = \Theta(1/(L\sqrt{n_{\text{out}}}))$, which leads to

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

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

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

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

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

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

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

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

C.4 AdamW

In this section, we recover the μP formulation under width-depth scaling of AdamW in Dey et al. (2025).

C.4.1 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.

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

where

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

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:

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

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

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

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

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

C.4.2 DERIVATION OF PARAMETERIZATION

Input and output layers. When $\lambda_0 = 0$, given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the multiplier parameterizations $\alpha_0 = \Theta(1), \alpha_{L+1} = \Theta(1/n_{\mathrm{in}})$ in Equation (5), and the scale of $\|\boldsymbol{A}_l\|_{\mathrm{R}}$ in Equation (21), we have

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

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 (4) that $\|\boldsymbol{W}_0\|_{\mathrm{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\mathrm{R}} = \Theta(n_{\mathrm{in}})$, we have

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

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 $\lambda_l = 0$, 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 (4), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (6), 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 (4), 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 Appendix B.1.3 or in Appendix C.2 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.3.

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

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

Therefore, we set

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

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

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

Table 5: **μP implementation 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	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_L r_n)$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/r_n$ $(\varepsilon_{\text{base}})$	$\varepsilon_{\text{base}}/(r_L r_n)$ $(\varepsilon_{\text{base}})$

Therefore, we set

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

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

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

Similarly, for the biases we have derived in Appendix C.3 that

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

Therefore, we set the stabilization term as

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

This completes the implementation of the update condition for AdamW, which is summarized in Table 5.

C.5 SHAMPOO

Denoting $\boldsymbol{G}_l^{(t)} = \nabla_{\boldsymbol{W}_l^{(t)}}\mathcal{L}$, the update rule of Shampoo is

$$\boldsymbol{W}_l^{(t)} = \boldsymbol{W}_l^{(t-1)} - \eta^{(t)}\left(\left(\boldsymbol{L}_l^{(t)}\right)^{-\frac{1}{4}}\boldsymbol{G}_l^{(t)}\left(\boldsymbol{R}_l^{(t)}\right)^{-\frac{1}{4}} + \lambda_l \boldsymbol{W}_l^{(t)}\right), \tag{22}$$

where

$$\boldsymbol{L}_l^{(t)} = \boldsymbol{L}_l^{(t-1)} + \boldsymbol{G}_l^{(t)}\boldsymbol{G}_l^{(t)\top}, \quad \boldsymbol{R}_l^{(t)} = \boldsymbol{R}_l^{(t-1)} + \boldsymbol{G}_l^{(t)\top}\boldsymbol{G}_l^{(t)}. \tag{23}$$

We simplify the full update rule by omitting the momentum, i.e., setting $\boldsymbol{L}_l^{(t-1)} = \boldsymbol{0}$ and $\boldsymbol{R}_l^{(t-1)} = \boldsymbol{0}$. Then, we have

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

Applying SVD to $\boldsymbol{G}_l^{(t)}$ as in Muon, we have

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

and thus

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

Then, the Shampoo is reduced to:

$$\begin{aligned}
&\boldsymbol{W}_l^{(t)} \\
&= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\left(\boldsymbol{U}_l^{(t)} {\boldsymbol{\Sigma}_l^{(t)}}^2 {\boldsymbol{U}_l^{(t)}}^\top \right)^{-\frac{1}{4}} \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} {\boldsymbol{V}_l^{(t)}}^\top \left(\boldsymbol{V}_l^{(t)} {\boldsymbol{\Sigma}_l^{(t)}}^2 {\boldsymbol{V}_l^{(t)}}^\top \right)^{-\frac{1}{4}} + \lambda_l \boldsymbol{W}_l^{(t)} \right) \\
&= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{U}_l^{(t)} {\boldsymbol{\Sigma}_l^{(t)}}^{-\frac{1}{2}} {\boldsymbol{U}_l^{(t)}}^\top \boldsymbol{U}_l^{(t)} \boldsymbol{\Sigma}_l^{(t)} {\boldsymbol{V}_l^{(t)}}^\top \boldsymbol{V}_l^{(t)} {\boldsymbol{\Sigma}_l^{(t)}}^{-\frac{1}{2}} {\boldsymbol{V}_l^{(t)}}^\top + \lambda_l \boldsymbol{W}_l^{(t)} \right) \\
&= \boldsymbol{W}_l^{(t-1)} - \eta^{(t)} \left(\boldsymbol{U}_l^{(t)} {\boldsymbol{V}_l^{(t)}}^\top + \lambda_l \boldsymbol{W}_l^{(t)} \right),
\end{aligned}$$

which matches exactly the update rule of Muon in Equation (16). Therefore, Shampoo shares Muon's parameterizations in Table 3.

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 $\boldsymbol{R}_l^{(t)}$ in Equation (22) which equal $\frac{1}{4}$ in the standard Shampoo, 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 3.

C.6 SOAP

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 weighted version of Shampoo's Equation (23) 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:

$$\boldsymbol{L}_l^{(t)} = \boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)} \boldsymbol{\Lambda}_{\boldsymbol{L}_l}^{(t)} {\boldsymbol{Q}_{\boldsymbol{L}_l}^{(t)}}^\top, \quad \boldsymbol{R}_l^{(t)} = \boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)} \boldsymbol{\Lambda}_{\boldsymbol{R}_l}^{(t)} {\boldsymbol{Q}_{\boldsymbol{R}_l}^{(t)}}^\top. \tag{24}$$

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.4.1.

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 (24), 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:

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

which, again, matches exactly the update rule of Muon in Equation (16). Therefore, SOAP shares Muon's parameterizations in Table 3.

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 3.

C.7 Spectral Sphere Optimizer (SSO)

C.7.1 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

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

Thus we have

$$\|\boldsymbol{A}_l\|_{\text{R}} = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \|\boldsymbol{A}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} R \|\boldsymbol{\Phi}_l\|_2 = \sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}} \Theta\left(\sqrt{\frac{n_{\text{out}}}{n_{\text{in}}}}\right) = 1. \tag{25}$$

C.7.2 Derivation of Parameterization

Input and output layers. When $\lambda_0 = 0$, 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 (5), and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ in Equation (25), 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 (4) that $\|\boldsymbol{W}_0\|_{\text{R}} = \Theta(1)$ and $\|\boldsymbol{W}_{L+1}\|_{\text{R}} = \Theta(n_{\text{in}})$, we have

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

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

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

Table 6: **μP implementation 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})

Hidden layers (first-order). When $\lambda_l = 0$, given the dimension assumption $d_0, d_{L+1} = \Theta(1), n_l = \Theta(n)$ in Equation (3), the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ in Equation (4), the multiplier parameterization $\alpha_l = \Theta(1/L)$ in Equation (6), and the scale of $\|\boldsymbol{A}_l\|_{\text{R}}$ in Equation (25), we have

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

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

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

When $\lambda_l \neq 0$, given the weight norm $\|\boldsymbol{W}_l\|_{\text{R}} = \Theta(1)$ Equation (4), we have

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

To satisfy (Δ2) that $\|\lambda_l \boldsymbol{W}_l\|_{\text{R}} = \Theta\left(\|\boldsymbol{A}_l\|_{\text{R}}\right)$ 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 or in Appendix C.2 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 6.

C.8 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

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

If the momentum terms are omitted, i.e., setting $\beta_1 = 0$ and $\beta_2 = 0$, and $\varepsilon_l = 0$, the Lion is reduced to sign gradient descent as AdamW with the update rule in Equation (20). Therefore, we reuse AdamW's parameterizations in Table 5 for Lion.

C.9 SOPHIA

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

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

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 that

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

where the elements of $\hat{\boldsymbol{h}}_l^{(t)}$ is the second-order derivate regarding $\boldsymbol{W}_l^{(t)}$ as $\left(\hat{\boldsymbol{h}}_l^{(t)}\right)_{ij} = \frac{\partial^2 \mathcal{L}}{\partial (\boldsymbol{W}_l^{(t)})_{ij}^2}$.

Letting $\boldsymbol{A}_l^{(t)} = \text{clip}\left(\frac{\boldsymbol{m}_l^{(t)}}{\max\{\gamma \boldsymbol{h}_l^{(t)}, \varepsilon\}}, 1 \right)$, we use the upper bound below to estimate its norm following Ngom et al. (2025) where they derive 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). Observing that the order of $\|\boldsymbol{A}_l^{(t)}\|_{\text{R}}$ equals that for AdamW in Equation (21), thus Sophia shares AdamW's parameterizations in Table 5.

D ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

In this section, we present the additional experimental details and results that are omitted from the main text.

D.1 ASSETS AND LICENSES

All used assets (datasets and codes) and their licenses are listed in Table 7.

Table 7: **Used assets and their licenses.**

URL	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

D.2 ADDITIONAL DETAILS OF FEATURE LEARNING EXPERIMENTS

The feature learning stability results in Figure 1 are averaged over three independent runs with different random seeds. The base initialization variance for matrix weights and biases is set to 0.02^2 and 0, respectively. All models are trained using a constant learning rate of 2^{-7}, a batch size of 8, a gradient clipping of 1, for 10 training steps.

D.3 ADDITIONAL DETAILS OF HP TRANSFER EXPERIMENTS

D.3.1 EXPERIMENTAL SETUP

For all HP transfer results shown in Figure 1, the base initialization variance for matrix weights and biases is set to 0.02^2 and 0, respectively. All models are trained with a batch size of 240 for a total of 1221 iterations (corresponding to 300M tokens), using a warm-up of 120 iterations followed by cosine base learning rate decay to 3×10^{-5}, and gradient clipping of 1.

Table 8: **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**. Under SP, the optimal base learning rate does not transfer across different widths.

$n/\log_2(\eta_{\text{base}})$	-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

D.3.2 Additional Results of Width-wise HP Transfer

Completed results of base learning rate transferability across different widths are presented in Table 8 and Table 9 for SP and μP, respectively.

Table 9: **μP succeeds in transferring 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**. Under μP, the optimal base learning rate (approximately) transfers across different widths.

$n/\log_2(\eta_{\text{base}})$	-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

D.3.3 Additional Results of Depth-wise HP Transfer with LayerNorm

With LayerNorm, completed results of base learning rate transferability across different depths are presented in Table 10 and Table 11 for SP and μP, respectively.

D.3.4 Additional Results of Depth-wise HP Transfer without LayerNorm

Without LayerNorm, the base learning rate transferability across different depths of SP and μP across depth are presented in Figure 2, with completed numerical results are presented in Table 12 and Table 13 for SP and μP, respectively.

Table 10: **With LayerNorm, SP succeeds in transferring the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each depth is highlighted in **bold**. Under SP, the optimal base learning rate transfers across different depths.

$L/\log_2(\eta_{\text{base}})$	-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 11: **With LayerNorm, μP succeeds in transferring the optimal base learning rate across depths.** This table reports the numerical results corresponding to Figure 1, where the best validation loss for each depth is highlighted in **bold**. Under μP, the optimal base learning rate transfers across different depths.

$L/\log_2(\eta_{\text{base}})$	-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

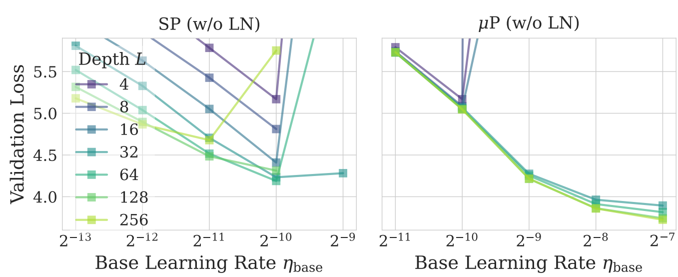

Figure 2: **Feature learning and HP transfer under SP and μP without LayerNorm.** We compare SP and μP along two dimensions. First, in terms of training stability, SP becomes increasingly prone to loss divergence as depth increases in the absence of LayerNorm, whereas μP enables stable training. Second, unlike SP, μP preserves HP transferability at large depths without LayerNorm.

Table 12: **Without LayerNorm, SP fails to preserve stable training or transfer optimal base learning rate across depths.** NaN data points indicate training instability, where the loss explodes. The best validation loss for each depth is highlighted in **bold**. Under SP, the optimal base learning rate does not transfer across large depths.

$L/\log_2(\eta_{\text{base}})$	-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 13: **Without LayerNorm, μP succeeds in preserving stable training and transferring the optimal base learning rate across depths.** NaN data points indicate training instability, where the loss explodes. The best validation loss for each depth is highlighted in **bold**. Under μP, the optimal base learning rate transfers across different depths larger than 32.

$L/\log_2(\eta_{\text{base}})$	-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**

E JUSTIFICATION OF UPPER BOUND ESTIMATION

In the derivation in Appendix B.1, we implicitly 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 actual 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.

E.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

$$\begin{aligned}\Delta\boldsymbol{h}_s(\boldsymbol{x}) &= \Delta\boldsymbol{h}_0(\boldsymbol{x}) + \underbrace{\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x})}_{\boldsymbol{\epsilon}_0(s)} + \underbrace{\sum_{l=1}^{s}\alpha_l\boldsymbol{W}_l^{(2)}\Delta\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(1)}(s)} \\ &+ \underbrace{\sum_{l=1}^{s}\alpha_l\Delta\boldsymbol{W}_l^{(2)}\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_1^{(2)}(s)} + \underbrace{\sum_{l=1}^{s}\alpha_l\Delta\boldsymbol{W}_l^{(2)}\Delta\boldsymbol{W}_l^{(1)}(\boldsymbol{h}_{l-1}(\boldsymbol{x})+\Delta\boldsymbol{h}_{l-1}(\boldsymbol{x}))}_{\boldsymbol{\epsilon}_2(s)}\end{aligned} \tag{26}$$

which leads to the upper bound in Equation (10):

$$\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}} \le \|\Delta\boldsymbol{h}_0(\boldsymbol{x})\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_0(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(1)}(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_1^{(2)}(L)\|_{\mathrm{R}} + \|\boldsymbol{\epsilon}_2(L)\|_{\mathrm{R}}.$$

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 Dey et al. (2025)). 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). 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 true scaling behavior of the feature updates.

E.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.

E.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 E.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

$$\begin{aligned}\|\boldsymbol{W}_l^{(2)} \boldsymbol{W}_l^{(1)} \boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}} &= \|\boldsymbol{W}_l^{(2)} \boldsymbol{z}_l\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l^{(2)}\|_{\mathrm{R}} \|\boldsymbol{z}_l\|_{\mathrm{R}}) \\ &= \Theta(\|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{W}_l^{(1)}\|_{\mathrm{R}} \|\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}),\end{aligned}$$

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. ☐

E.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 E.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 E.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 E.1. Therefore, the estimation of $\|\Delta \boldsymbol{h}_L(\boldsymbol{x})\|_{\mathrm{R}}$ by using submultiplicativity inequalities in Appendix B.1 is tight.

F 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: *multi-step* updates of a *non-linear* residual MLP on a *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 F.2.

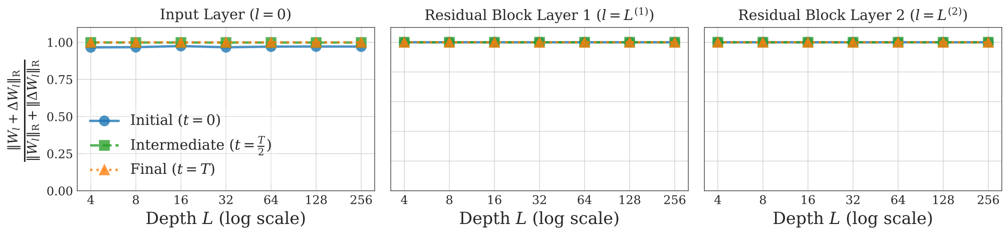

Figure 3: **Validation of Assumption F.1 (Weight Update).** The ratio $\frac{\|\boldsymbol{W}_l+\Delta\boldsymbol{W}_l\|_{\mathrm{R}}}{\|\boldsymbol{W}_l\|_{\mathrm{R}}+\|\Delta\boldsymbol{W}_l\|_{\mathrm{R}}}$ remains constant near 1 across depth for the input layer and residual block layers, showing non-vanishing updates throughout multiple-step training.

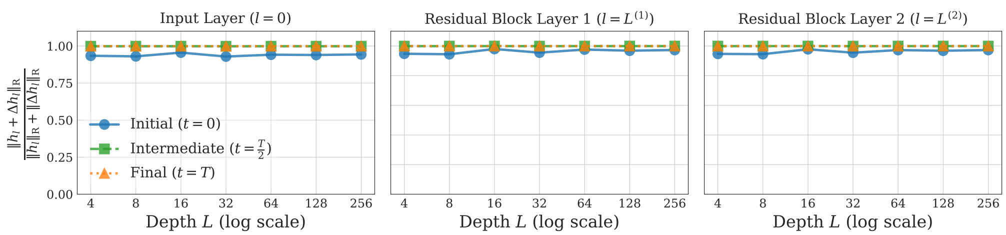

Figure 4: **Validation of Assumption F.1 (Feature Update).** The ratio $\frac{\|\boldsymbol{h}_l+\Delta\boldsymbol{h}_l\|_{\mathrm{R}}}{\|\boldsymbol{h}_l\|_{\mathrm{R}}+\|\Delta\boldsymbol{h}_l\|_{\mathrm{R}}}$ remains around constant near 1 across varying depths, showing non-vanishing updates throughout multiple-step training.

F.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 depth to achieve feature learning. To ensure these properties hold throughout *multi-step* training, the updated parameters must maintain the same scaling order as the first step. This is formalized in Assumption F.1.

Assumption F.1 (Non-vanishing Update)**.** For any layer l, the updated weights and feature vectors 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 F.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 3 and 4, where the norm ratios remain constant across varying depths.

Non-linearity. To extend the analysis to non-linear architectures, 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.

Assumption F.2 (Stable Activation)**.** The activation function ϕ satisfies:

$$\|\phi(\boldsymbol{W}_l\boldsymbol{h}_{l-1}(\boldsymbol{x}))\|_{\mathrm{R}} = \Theta(\|\boldsymbol{W}_l\boldsymbol{h}_{l-1}(\boldsymbol{x})\|_{\mathrm{R}}).$$

Figure 5 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

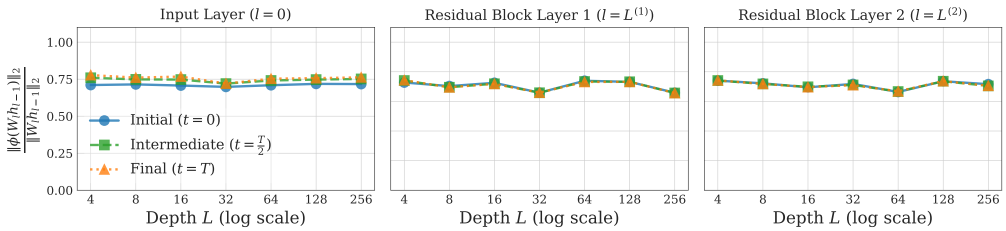

Figure 5: **Validation of Assumption F.2 (Stable Activation).** The ratio of post-activation to pre-activation norms $\frac{\|\phi(\boldsymbol{W}_l \boldsymbol{h}_{l-1})\|_{\mathrm{R}}}{\|\boldsymbol{W}_l \boldsymbol{h}_{l-1}\|_{\mathrm{R}}}$ remains stable across varying depths, confirming that the ReLU activation does not collapse the norm in non-linear networks.

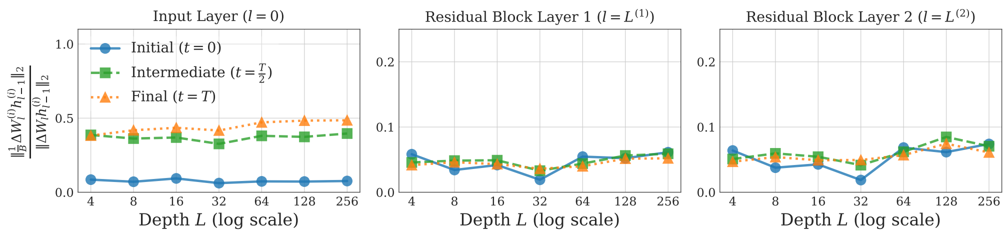

Figure 6: **Validation of Assumption F.3 (Per-sample Update Alignment).** We report the avraged ratio of $\frac{\|\Delta \boldsymbol{W}_l \boldsymbol{h}_{l-1}\|_{\mathrm{R}}}{\frac{1}{B}\sum_i \|\Delta \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}\|_{\mathrm{R}}}$ across the batch of data. The values remain $\Theta(1)$ across varying depths, suggesting that the batch update does not alter the depth-wise scaling of the single-sample update.

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 F.3.

Assumption F.3 (Per-sample Update Alignment)**.** The batch update norm scales consistently with the per-sample update norm:

$$\|\Delta \boldsymbol{W}_l \boldsymbol{h}_{l-1}(\boldsymbol{x}^{(i)})\|_{\mathrm{R}} = \Theta\left(\frac{1}{B}\|\Delta \boldsymbol{W}_l^{(i)} \boldsymbol{h}_{l-1}(\boldsymbol{x}^{(i)})\|_{\mathrm{R}}\right).$$

We verify this in Figure 6, where the alignment ratio remains $\Theta(1)$, indicating that batch-averaged updates preserve the scaling properties of single-sample updates.

F.2 Experimental Details

We conduct simulations to empirically Assumptions F.1–F.3. Our experimental setup largely follows (Yang et al., 2023), with different emphasize 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 Appendix C.3 (see Table 4) as follows:

$$\alpha_0 = \alpha_{\text{base}}, \quad \alpha_l = \frac{\alpha_{\text{base}}}{L}, \quad \alpha_{L+1} = \frac{\alpha_{\text{base}}}{n},$$
$$\sigma_0^2 = \frac{\sigma_{\text{base}}^2}{d_0}, \quad \sigma_l^2 = \frac{\sigma_{\text{base}}^2}{n}, \quad \sigma_{L+1}^2 = \sigma_{\text{base}}^2,$$
$$\eta_0 = \eta_{\text{base}} n, \quad \eta_l = \eta_{\text{base}} L, \quad \eta_{L+1} = \eta_{\text{base}} n,$$

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.