QUANTIFYING HYPERPARAMETER TRANSFER & THE IMPORTANCE OF EMBEDDING LAYER LEARNING RATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyperparameter transfer allows extrapolating optimal optimization hyperparameters from small to large scales, making it critical for training large language models (LLMs). This is done either by fitting a scaling law to the hyperparameters or by a judicious choice of parameterization, such as Maximal Update (μP), that renders optimal hyperparameters approximately scale-invariant. In this paper, we first develop a framework to quantify hyperparameter transfer through three metrics: (1) the quality of the scaling law fit, (2) the robustness to extrapolation errors, and (3) the asymptotic loss penalty due to choice of parameterization. Next, we investigate through a comprehensive series of ablations why μP appears to offer high-quality learning rate transfer relative to standard parameterization (SP), as existing theory is inadequate. We find that the overwhelming benefit of μP relative to SP when training with AdamW arises simply from maximizing the learning rate of the embedding layer. In SP, the embedding layer learning rate acts as a bottleneck that induces training instabilities; increasing it by a factor of width to match μP dramatically smooths out training while improving hyperparameter transfer. We also find that weight decay improves the scaling law fits, while, in the fixed token-per-parameter setting, it hurts the robustness of the extrapolation.

1 INTRODUCTION

Training large neural networks requires carefully tuning numerous hyperparameters, including the learning rate, weight decay, and batch size, among others (DeepSeek-AI et al., 2024; Qwen et al., 2025; OLMo et al., 2025; Bergsma et al., 2025). As models scale into trillions of parameters, the cost of hyperparameter tuning at scale becomes prohibitive. The solution to this is *hyperparameter transfer*: one finds the optimal hyperparameters at small scales, along with a scaling law, which can then be used to extrapolate optimal hyperparameters at large scales. There are two main approaches in practice. The first approach fits functional forms to predict how the optimal learning rate η^* scales as the model and/or data is scaled (DeepSeek-AI et al., 2024; Porian et al., 2024; Bjorck et al., 2025). While effective, these fitted relationships may not generalize beyond their fitted domain and themselves require expensive hyperparameter searches to obtain a robust functional form.

The second approach is more structural—it parameterizes the model such that the activations and their updates remain independent of width (Yang & Hu, 2021), keeping the training dynamics invariant across scales (Bordelon & Pehlevan, 2022; Yaida, 2022). Yang & Hu (2021) derived Maximal Update Parameterization (μP) under this desideratum, originally to study feature learning in the infinite width limit. Empirically, μP has a desirable practical property—it maintains *fairly consistent* optimal learning rates across width. This property, termed *learning rate transfer*, allows practitioners to utilize optimal learning rates from a smaller proxy model to train larger ones (Yang et al., 2021), reducing the need for expensive hyperparameter searches at scale. Several works have since extended this framework to depth scaling (Bordelon et al., 2024b; Yang et al., 2024; Dey et al., 2025), alternative architectures (Bordelon et al., 2024a; Dey et al., 2024; Vankadara et al., 2024; Małaśnicki et al., 2025; Jiang et al., 2026; Shigida et al., 2026), and optimizers (Littwin & Yang, 2023; Haas et al., 2024; Qiu et al., 2025), suggesting the underlying principle generalizes to the design space of modern networks.

The current practice raises several questions. First, after fitting a scaling law to the hyperparameters, they can always be reparameterized to be (approximately) scale-invariant. In this case, is there

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

any distinction with μP and its variants? This suggests an urgent need to *quantify* the quality of hyperparameter transfer in order to compare across different schemes. Secondly, the theoretical derivation of μP and its variants makes several assumptions that do not hold in practice. It assumes a finite number of training steps in the infinite width limit, whereas practical training runs are in the opposite limit, of a large number of training steps compared to width. It also assumes full alignment between weight updates and the activations (Yang & Hu, 2021). It also does not take into account aspects of the learning rate schedule, like learning rate warmup, which are crucial to observe transfer, as we show in Section I. Finally, it is derived in the setting of fixed dataset size, whereas typical training runs scale data and number of parameters in tandem, often in the setting of a fixed number of tokens per parameter (TPP) (Hoffmann et al., 2022). Surprisingly, despite these assumptions not holding, μP typically exhibits high-quality learning rate transfer in fixed step settings—though it can break down in specific regimes, such as across the under-to-over-parameterization transition (Kalra & Barkeshli, 2026). This raises the question of why transfer holds as well as it does.

In this paper, we begin by developing a framework to quantify hyperparameter transfer. We develop three key metrics. The first is the quality of the scaling law fit itself. The second is a robustness metric: whether the optimal hyperparameter choice is more robust to perturbations as the model scales. The third regards the asymptotic performance of the loss: it is possible that some parameterizations exhibit higher quality transfer at the cost of degradation in the absolute value of the loss, and this also needs to be taken into account. The three metrics can trade off against each other, as we will see.

Armed with a quantitative framework to measure the quality of hyperparameter transfer, we then investigate in more detail why μP appears to exhibit high-quality transfer. We show that in standard decoder-only Transformers (Vaswani et al., 2017b) trained with AdamW (Loshchilov & Hutter, 2019), μP has four key changes compared to SP. By examining all 16 ablations, we isolate the embedding layer learning rate as the key factor. μP has a much larger embedding layer learning rate compared to SP. By making this one alteration to SP, the hyperparameter transfer quality of SP essentially matches that of μP. We further study training dynamics, finding that the embedding layer learning rate is important throughout training, especially early in training, and if it is not sufficiently large, there are training instabilities. Therefore, one interesting source of training instabilities is that the learning can be bottlenecked by the embedding layer learning rate.

These results highlight the fundamental importance of the embedding layer learning rate. It is known that the embedding layer learns a significant amount of structure from the data (Nanda et al., 2023; He et al., 2024; Tao et al., 2025; Karkada et al., 2026), and as such, perhaps it is not surprising that keeping it too low can throttle training, while maximizing it helps stabilize the learning process.

Our contributions. Our paper contains the following contributions:

- A quantitative framework for measuring the quality of hyperparameter transfer. This involves three metrics: (1) the quality of the scaling law fits in terms of an error $\mathcal{E}$, (2) a transfer robustness exponent κ which measures sensitivity to errors in extrapolation of a hyperparameter from small to large scales, and (3) the asymptotic loss degradation $\mathcal{R}(\infty)$, which measures performance at scale relative to the optimal parameterization.
- Focusing on GPT pre-training with AdamW, we express SP and μP in a common form (Table 1). The two parameterizations differ in four key ways (SP $\to$ μP): (1) embedding learning rate $\Theta(1/n) \to \Theta(1)$, (2) last layer initialization variance $\Theta(1/n) \to \Theta(1/n^2)$, (3) LayerNorm learning rate ($\Theta(1/n) \to \Theta(1)$), (4) attention scaling ($1/\sqrt{d} \to 1/d$).
- We examine all 16 ablations that distinguish μP from SP and isolate the embedding layer learning rate as the primary factor: SP with an appropriately scaled embedding layer learning rate essentially matches μP in quality of hyperparameter transfer.
- We find that the embedding layer learning rate, if not large enough, can cause training instabilities.
- We study the effect of weight decay to show that (1) it can improve the scaling law fit quality (smaller $\mathcal{E}$), (2) in the fixed token per parameter setting, it hurts the robustness of the extrapolation.

We discuss related work in Section A.

2 PRELIMINARIES: NEURAL NETWORK PARAMETERIZATIONS

We consider a neural network f_θ with parameters θ trained by minimizing a loss $L(\theta)$ using AdamW optimizer (Loshchilov & Hutter, 2019) with learning rate η and weight decay λ. A model's performance can be improved by scaling along various dimensions, such as width, depth, context length, and data. A *parameterization* is a set of rules that specify how key network and optimizer hyperparameters must be adjusted as the model is scaled to maintain stable training. In this work, we focus on the width n as the scaling dimension and the learning rate η as the hyperparameter scaled.

Following Yang & Hu (2021), we parameterize the network and optimizer using four scalar exponents $\{a_l, b_l, c_l, d_l\}$ per layer l, which control different aspects of width scaling. The exponent a_l controls the forward pass scaling $\mathbf{h}^{(l+1)} = n^{-a_l} W^{(l+1)} \phi(\mathbf{h}^{(l)})$, where $\mathbf{h}^{(l)}$ and $W^{(l)}$ denote the pre-activations and weights in layer l, and $\phi(\cdot)$ is the activation function. The exponent b_l scales the initialization variance $W^{(l)} \sim \mathcal{N}(0, n^{-2b_l})$. The exponent c_l scales the layer-wise learning rate $\eta^{(l)} = \eta \cdot n^{-c_l}$, where η is the global learning rate. Finally, the exponent d_l scales the weight decay strength $\lambda^l = \lambda \cdot n^{-d_l}$ such that the product $\eta^l \cdot \lambda^l$ is width-independent. For Transformers, the usual scaled dot-product attention comes with a per-head scaling by the head dimension d, with the standard choice being $1/\sqrt{d}$ (Vaswani et al., 2017a).

Given this general setup, specific parameterizations are obtained by imposing stability conditions on the training dynamics. Two canonical examples of *abcd* parameterization are SP (Sohl-Dickstein et al., 2020) and μP (Yang & Hu, 2021). SP requires that activations at initialization do not blow up or vanish as the width grows, imposing one constraint per layer; any additional freedom is used to make the simplest choices, such as a uniform learning rate for all layers. μP imposes a stronger condition: both the activations and their updates must be width-independent, resulting in two constraints per layer. For analytical tractability, these additional constraints are derived under three additional assumptions: (1) a finite number of training steps as the width $n \to \infty$, (2) full alignment between weight updates and the activations they act on (Yang & Hu, 2021), and (3) a fixed dataset as n scales. We provide a self-contained derivation of μP for both SGD and Adam optimizers in Section K.

Since one of our goals is to understand which aspects of μP are essential for transfer, we need to express SP and μP in a common form. To this end, SP is defined with a weight initialization variance of $\Theta(1/\text{fan-in})$, and a global learning rate $\eta_l = \eta \cdot n^{-1}$ for all layers. Note that we choose a convention where we peel off a $1/n$ factor in the learning rate, which differs from other conventions. To make the comparison between SP and μP more transparent, we use the symmetry of abcd parameterization (Yang & Hu, 2021) to set the multipliers $a = 0$ in our description of μP. As shown in Table 1, the two differ in four key ways: (1) embedding layer learning rate, (2) last layer initialization variance, (3) LayerNorm learning rate, and (4) attention scaling. Weight decay strength λ is scaled with learning rate so that its contribution to the weight update, $\eta\lambda\theta$, is $\Theta(1)$ in both parameterizations.

The intuitive reason for these changes is as follows. SP applies a uniform $\Theta(1/n)$ learning rate to all layers. This scaling is needed for the hidden layers, which compute their output as a sum of n input terms $h_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{l+1} h_j^{(l)}$; without a $\Theta(1/n)$ learning rate the activations updates $\Delta h_i^{(l+1)}$ would scale as $\Theta(n)$. By comparison, the embedding (a per-token lookup) and LayerNorm (an elementwise operation) do not involve a sum over the width dimension, so a $\Theta(1)$ learning rate is a natural choice for these layers. Next, SP's larger last layer initialization variance leads to higher Hessian sharpness at initialization, making training unstable at large learning rates (Kalra et al., 2025). However, in practice, learning rate warmup mitigates this by gradually reducing sharpness to a level determined by the peak learning rate (Kalra & Barkeshli, 2024), making the initialization difference between SP and μP negligible. Prior work reaches the same conclusion by different arguments (Everett et al., 2024; Haas et al., 2025). Finally, μP replaces SP's $1/\sqrt{d}$ attention scaling with $1/d$, motivated by the observation that key and query vectors are projections of the same input and therefore would be more aligned than independent random vectors during training. In this work, we show that the embedding layer learning rate is the primary driver of μP's advantage, with the remaining modifications contributing little.

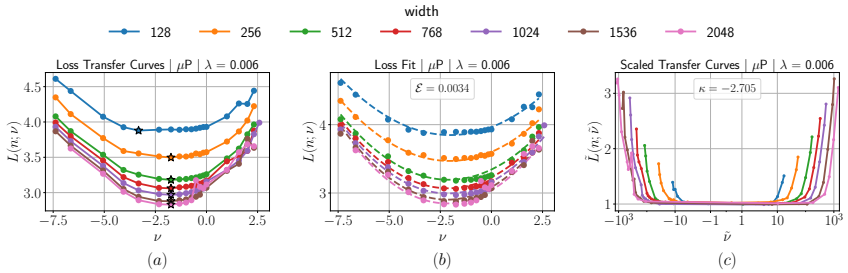


Figure 1: *Transfer curves for μP.* (a) Loss vs. log learning rate ν, with star marking the optimum $\nu^*(n)$, (b) Joint fit of the loss model (Equation (6), dashed lines), with a low predictability error $\mathcal{E} = 0.0034$, (c) Loss curves in the normalized coordinates (Equation (8)), with $\kappa = -2.640$ indicating robust transfer. The corresponding scaling laws are shown in Figure 8.

3 Quantifying Hyperparameter Transfer

In this section, we focus on learning rate η as the hyperparameter of interest and consider scaling the width n. In principle, the discussion can be extended to include additional hyperparameters (e.g., batch size, weight decay) and scaling other quantities such as depth, context length, training steps.

A parameterization is effective for learning rate transfer if the optimal learning rate can be reliably extrapolated from small to large widths without sacrificing performance at scale. Reliability requires two conditions. First, the loss at the end of training must be predictable, for example, by satisfying a simple functional scaling law form. Second, even if the loss is predictable, the transferred learning rate will inevitably carry some residual error (e.g., from finite sampling at the small scale). The loss at scale must be robust to such errors—if a small error in predicting the learning rate induces an increasingly large loss gap at larger widths, transfer becomes sensitive despite predictability in the loss. Accordingly, we introduce three metrics: *Loss Predictability Error* $\mathcal{E}$, *Transfer Robustness Exponent* κ, and *Asymptotic Loss Degradation* $\mathcal{R}(\infty)$ that capture loss predictability, robustness to prediction errors, and performance at scale, respectively.

To formalize these metrics, we first need to model how the loss landscape changes with width and learning rate. Following standard practice in neural scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Barkeshli et al., 2026), we model the optimal loss $L^*(n)$ as a power law in width:

$$L^*(n) = L^*(\infty) + An^{-\alpha}, \tag{1}$$

where $L^*(\infty)$ is the irreducible loss, A is the scaling coefficient, and α is the scaling exponent. Here we can consider the dataset size to be held fixed or increasing along with n as in compute-optimal training. We model the optimal learning rate using a scaling law with an irreducible term as well:

$$\eta^*(n) = \eta^*(\infty) + B'n^{-\beta}, \tag{2}$$

where $0 < \eta^*(\infty) < \infty$ is the asymptotic optimal learning rate, B' is the scaling coefficient and $\beta > 0$ controls the rate of convergence. Note that this differs from the form $\eta^*(n) \sim n^{-\beta'}$ used elsewhere (Everett et al., 2024; Li et al., 2025), which implicitly assumes that the optimal learning rate vanishes ($\beta' > 0$) or diverges ($\beta' < 0$) at large widths. In Equation (2), we assume the dominant scaling law has been peeled off (as in our convention for SP), leaving the residual correction $B'n^{-\beta}$.

The scaling law above is expressed in terms of the learning rate η. However, the loss as a function of η is typically asymmetric around its optimum η^*—beyond it training becomes unstable, and the loss increases sharply, while below it the loss degrades gradually. We therefore work in log-learning-rate space $\nu := \log_2 \eta$ for the rest of the paper, where the loss landscape is more symmetric around its optimum (see Figure 1(a)). Equation (2) then takes the form:

$$\nu^*(n) = \nu^*(\infty) + \log\left(1 + \frac{B'n^{-\beta}}{\eta^*(\infty)}\right) \approx \nu^*(\infty) + Bn^{-\beta},$$

where $B = B'/\eta^*(\infty)$. Since Eq. 2 captures the residual width dependence after the dominant scaling has been removed, the term $B'n^{-\beta}/\eta^*(\infty)$ is small, justifying the first-order expansion above. The above scaling laws hold for each parameterization separately; different parameterizations may, for example, exhibit different irreducible loss. As such, a parameterization with high-quality transfer might sacrifice loss performance. This motivates the following definition.

Definition 3.1 (Loss Degradation $\mathcal{R}(\infty)$)**.** For a parameterization, let $L^*(\infty)$ denote its irreducible loss, and let $L^*_{\text{best}}(\infty)$ be the best possible irreducible loss across hyperparameters and parameterizations. We define loss degradation as the asymptotic loss gap:

$$\mathcal{R}(\infty) = L^*(\infty) - L^*_{\text{best}}(\infty) \geq 0. \tag{3}$$

A parameterization with $\mathcal{R}(\infty) = 0$ achieves best-in-class loss at scale, while $\mathcal{R}(\infty) > 0$ indicates a performance gap. In practice, L^*_{best} is computed using a set of parameterizations under consideration.

Next, to capture the reliability of transfer, we model the loss around $\nu^*(n)$ as a local quadratic form:

$$L(\nu; n) = L^*(n) + \frac{1}{2} H(n) \cdot (\nu - \nu^*(n))^2 + \mathcal{O}(\nu^3), \tag{4}$$

where $H(n) = \nabla^2_\nu L(\nu^*(n); n)$ is the loss Hessian evaluated at $\nu^*(n)$. In addition to the loss and log-learning rate, we model the Hessian scaling as a power law[1]:

$$H(n) = Cn^\gamma, \tag{5}$$

where C is the scaling coefficient and γ is the scaling exponent. Substituting the scaling laws into Equation (4), we obtain:

$$L(\nu; n) = L^*(\infty) + An^{-\alpha} + \frac{1}{2} Cn^\gamma \cdot \left(\nu - \nu^*(\infty) - Bn^{-\beta}\right)^2. \tag{6}$$

This functional form serves as our scaling ansatz for learning rate transfer.

Definition 3.2 (Loss Predictability Error $\mathcal{E}$)**.** Given losses $\{L(\nu_i; n_j)\}$ at N_ν log-learning-rates $\{\nu_i\}_{i=1}^{N_\nu}$ and N_n widths $\{n_j\}_{j=1}^{N_n}$, define loss predictability error as the normalized mean squared difference between observed and predicted loss $\{\hat{L}(\nu_i; n_j)\}$ from Equation (6) with fitted parameters:

$$\mathcal{E} = \frac{1}{N_\nu N_n} \sum_{i,j}^{N_\nu, N_n} \left[L(\nu_i; n_j) - \hat{L}(\nu_i; n_j)\right]^2. \tag{7}$$

When $\mathcal{E} \approx 0$, the loss is well captured by Equation (6), suggesting reliable extrapolation across widths is possible. High $\mathcal{E}$, in contrast, suggests a complex landscape arising from sources such as training instabilities, finite-size effects, or phase transitions, making an extrapolation unreliable.

While $\mathcal{E}$ captures the predictability of the landscape, it does not reveal how errors in extrapolating the optimal learning rate impact performance at scale. To analyze this sensitivity, we normalize loss and log-learning-rate, focusing on the scale-invariant landscape defined by the normalized coordinates:

$$\tilde{L}(n) = \frac{L(\nu; n) - L^*(\infty)}{An^{-\alpha}}, \qquad \tilde{\nu}(n) = \frac{\nu - \nu^*(\infty)}{Bn^{-\beta}}. \tag{8}$$

In these normalized coordinates, Equation (6) becomes:

$$\tilde{L}(n) = 1 + \frac{C}{2AB^2} n^{\alpha - 2\beta + \gamma} \left(\tilde{\nu}(n) - 1\right)^2. \tag{9}$$

The normalized loss curvature scales as n^κ, where $\kappa = \alpha - 2\beta + \gamma$, determines whether the landscape flattens ($\kappa < 0$) or sharpens ($\kappa > 0$) as width increases. The sign of κ thus determines the sensitivity of the loss to learning rate prediction errors at scale.

Definition 3.3 (Transfer Robustness Exponent κ)**.** We say that a parameterization under our ansatz (Equation (6)) exhibits *robust transfer* if $\kappa = \alpha - 2\beta + \gamma \leq 0$.

The sign of κ controls how prediction errors propagate to loss at scale. Negative κ means the landscape flattens, so errors in extrapolating $\nu^*(n)$ from small widths results in diminishing loss penalties at scale. By comparison, positive κ amplifies these errors and degrades transfer reliability, as the landscape sharpens with width. When $\gamma = 0$, the condition $\alpha \leq 2\beta$ coincides with the *fast transfer* criterion of Ghosh et al. (2025).

Taken together, these metrics capture three complementary axes of transfer. $\mathcal{R}(\infty)$ captures performance at scale, $\mathcal{E}$ quantifies if the loss is predictable, and κ measures whether prediction errors amplify or dampen at large widths. A parameterization can excel on one while failing on others, and as we show in later sections, the metrics can even be at odds with each other.

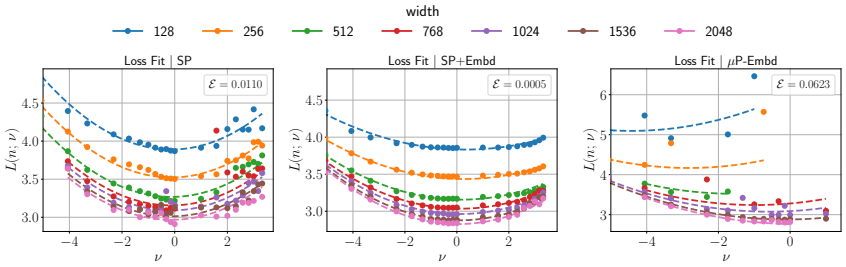


Figure 2: *Embedding layer learning rate is the critical difference between SP and μP.* Loss vs. log learning rate ν across widths for: (a) SP with $\Theta(1/n)$ embedding learning rate; (b) SP modified to use $\Theta(1)$ embedding learning rate (SP+Embd); (c) μP modified to use $\Theta(1/n)$ embedding learning rate (μP-Embd). Speeding up the embedding in SP eliminates training instabilities and yields smooth training, while slowing it down in μP introduces instabilities.

4 Examining μP and SP Through the Lens of the Transfer Metrics

We pre-train GPT-style Transformers on FineWeb-Edu (Penedo et al., 2024) using AdamW for a fixed number of steps ($T = 10,000$) using Warmup-Stable-Decay (WSD) schedule (Hu et al., 2024) (20% warmup, 60% stable, 20% decay) with batch size 1024 ($\approx$ 1M tokens per step). We scale the embedding dimension (width) $n \in [128, 2048]$ by increasing the number of heads, while keeping the head dimension fixed at $d = 64$. For each width, we sweep the peak learning rate and weight decay strength λ. For full experimental details, see Section B. We consider compute-optimal (fixed TPP) training in Section 6.

4.1 Methodology

Filtering and Interpolation. For each parameterization and width, we retain runs with loss within $1.35\times$ the per-width optimal loss to focus on the landscape around the minimum (Figure 1(a)). Since discrete sampling introduces noise in the estimated $\nu^*(n)$, we interpolate the resulting per-width curves using a cubic spline to obtain smoother, denser curves (see Figure 6a, Section D), enabling reliable estimation of $\nu^*(n)$ and $H(n)$. For the optimal loss, we use the raw per-width minimum directly, as it's already reliable.

Estimating the Three Transfer Metrics. By fitting scaling laws to $L^*(n)$ and $\nu^*(n)$ (Figure 8(d, e)), we obtain the exponents α, β and the asymptotic loss $L^*(\infty)$, from which we compute $\mathcal{R}(\infty)$. Since $\mathcal{R}(\infty)$ is non-negative by definition (Definition 3.1), we clamp fitted values near zero when they are negative due to finite-size fitting artifacts. For the curvature $H(n)$, we first fit a quadratic centered at $\nu^*(n)$ to the per-width loss curves (see Figure 6b in Section D), and then fit the scaling law $H(n) = Cn^\gamma$ (Figure 8(f)) to obtain the exponent γ. Combined with α and β from above, this gives $\kappa = \alpha - 2\beta + \gamma$, whose sign can be directly read off from the normalized transfer curves (Figure 1(c)), with a flattening landscape indicating robust transfer. Finally, to compute the loss predictability error $\mathcal{E}$, we jointly fit all the parameters using the interpolated curves, then measure $\mathcal{E}$ using the fit evaluated on the raw filtered data points. Fitting on the interpolated data gives a smoother fit, while evaluating on the raw data measures the fit on the observed data. Throughout, we cap all scaling exponents at 2.0 to prevent spuriously large values, which would otherwise dominate comparisons across parameterization.

When $\nu^*(n)$ is nearly constant across widths, its scaling law admits two degenerate solutions: $\beta \to 0$ (constant) and $\beta \to \infty$ (rapid convergence). We develop a procedure to distinguish between these two cases, and prefer the $\beta \to \infty$ solution as it is consistent with our framework's convergence-based interpretation. We provide full details on the fitting procedures and degeneracy resolution in Section D.

4.2 What Exactly is μP's Advantage over SP?

To find out the essential elements required for reliable transfer, we compare μP (Figure 1) and SP (Figure 2(a) and Figure 9 in Section E) using the three metrics. SP exhibits visibly noisier loss curves than μP due to training instabilities. Despite this, the two parameterizations are surprisingly

[1]Empirically, we find this form well approximates the Hessian scaling for parameterizations we considered.

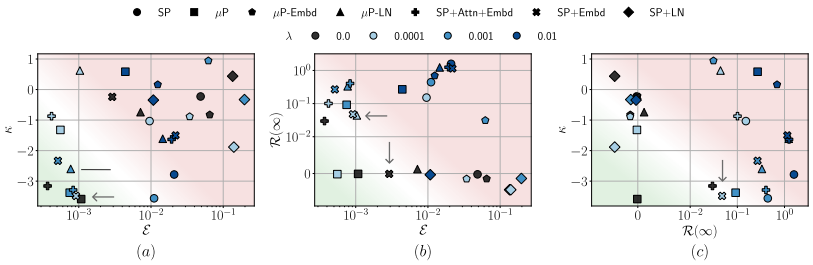


Figure 3: *Transfer metrics for parameterizations interpolating between SP and μP.* Parameterizations with '$+$' denote incremental changes from SP towards μP, while '$-$' denotes changes from μP towards SP. Green and red regions indicate desirable and undesirable regimes, respectively. The arrows highlight SP+Embd (SP with $\Theta(1)$ embedding learning rate), which matches μP across all three metrics, suggesting the embedding layer learning rate is the primary driver of μP's advantage.

similar on most metrics. The asymptotic loss gap $\mathcal{R}(\infty)$ of SP is slightly worse but still comparable to μP. For both parameterizations, $\nu^*(n)$ empirically converges to a finite asymptotic value. Finally, the normalized loss flattens for both parameterizations, with large negative robustness exponents, indicating transfer is robust in both cases. Where SP falls short is in the loss predictability: the loss vs. ν curves are visibly noisier due to training instabilities. As a result, the predictability error $\mathcal{E}$ is roughly $3\times$ larger than for μP, suggesting that the loss is poorly described by our ansatz. While SP can exhibit transfer in principle, training instabilities make it unreliable in practice.

4.3 A Step by Step Journey from SP to μP

The results above suggest that SP has the right ingredients for reliable learning rate transfer, but is held back by training instability. Since SP and μP differ in only four ways (Section 2), a natural question is which change, if any, is most significant. To answer this, we perform systematic modifications, starting from SP and making one change at a time, sweeping the peak learning rate and weight decay across widths. We caution that these modifications interact non-linearly, so the effect of one change may depend on the scaling of other layers. Figure 3 shows the three transfer metrics for selected ablations and weight decay values. We defer the full results to Figure 18 in Section E and summarize the key findings here.

The embedding layer learning rate emerges as the most critical modification (Figure 2): training SP with an $\Theta(1)$ embedding layer learning rate (SP+Embd) matches μP across metrics, while training μP with $\Theta(1/n)$ embedding layer learning rate (μP-Embd) degrades it. Attention scale has a more subtle effect: both adding $1/d$ scaling to SP (SP+Attn) and removing it from μP (μP-Attn) result in large positive κ, making transfer brittle. Since the head dimension is fixed at $d = 64$ in our experiments, this ablation measures sensitivity to the attention normalization constant rather than to a width scaling. Increasing the LayerNorm learning rate to $\Theta(1)$ in SP worsens instability, while decreasing it to $\Theta(1/n)$ in μP has a negligible effect, suggesting that LayerNorm parameters can be trained slowly without hurting performance. Finally, the last layer initialization variance has a negligible effect, though μP with a $1/n$ initialization (μP-Last) exhibits instabilities at small widths. We leave a detailed understanding of these observations to future work.

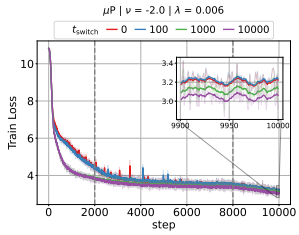
5 The Importance of Embedding Layer Learning Rate

The critical role of the embedding layer learning rate in stabilizing SP's training is surprising. Since the embedding layer performs a width-independent lookup, one would expect its learning rate to scale as $\Theta(1)$. But what is unexpected is that a small embedding learning rate results in noisy loss curves in SP[2](Figure 2), as one might expect later layers to compensate for a poorly trained embedding. In this section, we dig deeper into why training the embedding layer fast enough is critical.

To better understand the role of training the embedding layer, we examine when it matters most by switching its learning rate at various points during training. We perform two experiments: in μP, we slow down the embedding learning rate from $\Theta(1)$ to $\Theta(1/n)$ at step t_{switch}, and in SP, we speed it up from $\Theta(1/n)$ to $\Theta(1)$. Together, these experiments reveal that a small embedding layer

[2]These instabilities appear near the optimal learning rate; at smaller learning rates, SP trains stably but converges to worse final loss.

(a)



(b)

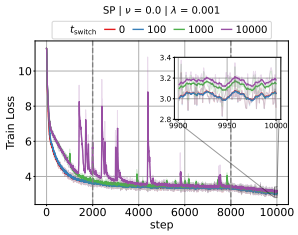


Figure 4: (a) Reducing the embedding learning rate from $\Theta(1)$ to $\Theta(1/n)$ in μP at step t_{switch} causes a persistent loss gap that grows larger for earlier switches (inset). (b) Increasing it from $\Theta(1/n)$ to $\Theta(1)$ in SP eliminates training instabilities and improves performance over SP ($t_{\text{switch}} = 10^4$).

learning rate not only slows down training but also causes instabilities, with early training being the most critical. In the μP case (Figure 4a), switching to $\Theta(1/n)$ embedding learning rate early in training drastically slows down training. While further training eventually closes much of this gap, a residual loss difference of $\sim$ 0.1-0.2 persists at the end of training, with a larger gap for earlier switches. By comparison, in the SP case (Figure 4b), switching to $\Theta(1)$ embedding learning rate at an early stage simultaneously eliminates the training instabilities and improves performance.

We additionally perform two complementary experiments to better understand the role of different layers. First, in Section J, we completely freeze the embedding layer at initialization, finding that this hurts both parameterizations, but μP much more than SP. This is surprising, as one might expect later layers to learn useful representations even with a randomly initialized embedding; however, the persistent gap indicates that they do not compensate for an untrained embedding. Second, we test whether this holds beyond the vanilla GPT setting. In Section F, we show that for LLaMA-style Transformers with RoPE, RMSNorm and SwiGLU, SP exhibits a large training-loss spike at its optimal learning rate that both μP and SP+Embd avoid. In Section G, we show that for CNNs trained on CIFAR-100 (Krizhevsky et al.), SP with a $\Theta(1)$ input-layer learning rate matches μP's transfer quality, while changing only the last-layer initialization has little effect. Together, these results suggest that the first and last layers play a special role across architectures, likely because they sit at the network boundary with no upstream or downstream processing to compensate for poor training. Therefore, extra care should be taken in setting their hyperparameters.

A natural question is: why does the embedding layer learning rate matter so much while the last-layer initialization does not? First, the two are not on equal footing in μP's derivation (see Section K). The $\Theta(1)$ embedding learning rate is required for the embedding's activation updates to be $\Theta(1)$. This first-layer update has only a single term contributing to it, so its scaling has to be correct. By contrast, the last-layer initialization contributes to only one of the three terms of the function update Δf_1, and its specific value depends on whether the last-layer weights are *fully aligned* with their activation updates during training (see Assumption K.6 in Section K). The constraint is therefore weaker and depends on an empirical alignment value that may not hold in practice—indeed, Everett et al. (2024) showed that relaxing this assumption still yields transfer. Relatedly, Haas et al. (2025) show that when the output dimension exceeds the width, the $\Theta(1/n)$ last-layer initialization of SP already gives width-independent logit updates, so μP's $\Theta(1/n^2)$ correction is not needed. Second, learning rate warmup steers training away from the higher sharpness induced by a larger last-layer initialization, into flatter regions of the landscape (Kalra & Barkeshli, 2024), compensating for the difference in initialization scale.

6 The Effect of Weight Decay and Compute Optimal Training

Most studies examining transfer focus on the fixed-step setting, often with little or no weight decay (Yang et al., 2021; Everett et al., 2024). While recent work has begun examining the effect of weight decay (Kosson et al., 2026) and compute-optimal scaling regime (Bergsma et al., 2025), their effect on transfer quality remains unclear. We find that weight decay improves loss predictability error $\mathcal{E}$ but consistently hurts asymptotic performance in the fixed-step setting. By comparison, this performance penalty disappears in the compute optimal scaling regime (20 tokens per parameter, following Hoffmann et al. (2022)), but transfer robustness degrades with weight decay, likely because the appropriate weight decay scaling in this regime is not well understood.

Weight Decay. In the fixed-step setting, weight decay consistently hurts asymptotic performance: $\mathcal{R}(\infty)$ monotonically increases with λ across parameterizations, from $\mathcal{R}(\infty) \approx 0.01$ to $\mathcal{R}(\infty) \approx 1$ at large λ (Figure 5a). The effect on loss predictability error $\mathcal{E}$ is more nuanced and depends on the

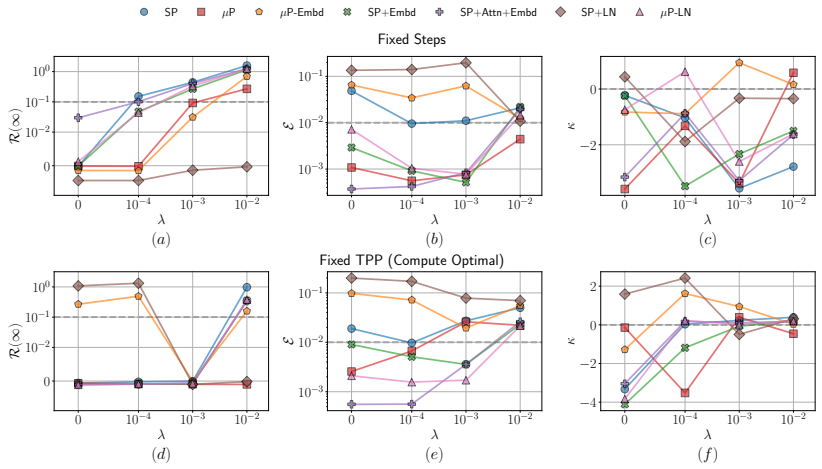


Figure 5: *Effect of weight decay on the three transfer metrics in the fixed-step (top) and fixed TPP (bottom) settings.* In the fixed-step setting, weight decay improves predictability error $\mathcal{E}$ but worsens asymptotic performance $\mathcal{R}(\infty)$. In the TPP setting, stable parameterizations achieve near-zero $\mathcal{R}(\infty)$, and predictability trends are similar, but robustness κ degrades as weight decay increases.

baseline stability of the parameterization without weight decay (Figure 5b). For μP and SP+Embd, which already exhibit low $\mathcal{E}$ at $\lambda = 0$, small weight decay further improves predictability before worsening at large λ. For SP and SP+LN, which suffer from training instabilities at $\lambda = 0$, weight decay steadily reduces $\mathcal{E}$ but is never sufficient to match the stable parameterizations. Interestingly, at large λ, $\mathcal{E}$ converges to ≈ 0.01 across parameterizations, suggesting that strong weight decay regularizes the landscape to a similar level regardless of parameterization. The improvement in loss predictability error $\mathcal{E}$ with weight decay has a natural interpretation: weight decay regularizes the loss landscape, reducing its complexity and making it better captured by our loss ansatz (Equation (6)). The transfer robustness exponent κ remains largely negative and shows no clear dependence on λ (Figure 5c).

Compute Optimal Training. In the compute-optimal setting, most parameterizations achieve near-zero $\mathcal{R}(\infty)$ (except for SP+LN and μP-Embd due to exhibiting training instabilities), suggesting that the loss is not sensitive to the choice of parameterization. Predictability error $\mathcal{E}$ follows similar trends to the fixed-step setting, with minor trend differences. A notable difference from the fixed-step setting lies in κ: at $\lambda = 0$, most parameterizations exhibit negative κ, but this robustness degrades sharply with increasing weight decay, with most parameterizations converging to $\kappa \approx 0$. A natural explanation is that μP assumes $\Theta(1)$ training steps relative to width, but in the compute optimal regime, steps T scale as n^2, violating this assumption and making the current convention $\eta \cdot \lambda = \Theta(1)$ inadequate. In Section H, we find that scaling weight decay as $\eta \cdot \lambda = \Theta(1/n^2)$ reduces the shift in optimal learning rate, but the shape of the loss curves around the minimum changes across widths.

7 Discussion and Conclusion

In this work, we introduced a quantitative framework for evaluating hyperparameter transfer, with three metrics $\mathcal{R}(\infty)$, $\mathcal{E}$, and κ that together serve as a diagnostic lens for identifying what a transfer setup is lacking. Our framework generalizes beyond learning rate transfer with width and can be applied to any hyperparameter and scaling dimension. For instance, it can help determine which depth scaling strategy yields more reliable transfer, whether learning rate transfer across tokens is as robust as across width, and whether current batch size and expert scaling conventions are brittle.

Using this framework, we find that under AdamW, the primary advantage of μP over SP comes from training the embedding layer at a sufficiently fast learning rate, and training the embedding layer slowly not only slows down learning but can also cause training instabilities. This suggests that the full set of μP conditions is excessive, and practitioners using SP can recover comparable transfer performance by simply correcting the embedding layer learning rate. Our results also show that the weight decay scaling convention $\eta \cdot \lambda = \Theta(1)$, derived under μP's fixed step assumption, is inadequate in the compute-optimal regime where the training horizon itself scales as $\Theta(n^2)$. We leave finding the correct weight decay scaling in this regime to future work.

AI USE STATEMENT

We used large language models (LLMs) only for editing, pruning, and formatting the text of this manuscript. The authors carried out ideation, derivations, experiments and interpretation of results themselves. We have reviewed AI-assisted edits and take responsibility for the final content of this work, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies optimization hyperparameters for neural network pre-training. It does not involve human subjects, personally identifiable information, or sensitive data, and all experiments use publicly available datasets: FineWeb-Edu (Penedo et al., 2024) and CIFAR-100 (Krizhevsky et al.). We do not foresee direct negative societal impacts beyond the general risks associated with advances in machine learning.

REPRODUCIBILITY STATEMENT

Our implementation is based on nanoGPT (Karpathy, 2022). Full experimental details are provided in Section B, including the model architecture, the optimizer and its hyperparameters, the learning rate schedule, and the dataset. The parameterizations we compare are specified in full in Table 1, and the procedures used to fit the scaling laws and to resolve degeneracies in the fits are described in Section D. Taken together, these sections provide sufficient details for reproducing our results.

REFERENCES

Maissam Barkeshli, Alberto Alfarano, and Andrey Gromov. On the origin of neural scaling laws: from random graphs to natural language, 2026. URL https://arxiv.org/abs/2601.10684.

Shane Bergsma, Nolan Simran Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in LLM pre-training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=bFXbLQzRoZ.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal LR across token horizons. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WYL4eFLcxG.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=sipwrPCrIS.

Blake Bordelon, Hamza Tahir Chaudhry, and Cengiz Pehlevan. Infinite limits of multi-head transformer dynamics. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=p0BBKhD5aI.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=KZJehvRKGD.

DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun,

Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism, 2024. URL https://arxiv.org/abs/2401.02954.

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

Nolan Simran Dey, Shane Bergsma, and Joel Hestness. Sparse maximal update parameterization: A holistic approach to sparse training dynamics. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=OWmu3QOa0O.

Katie E Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Peter J Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=0ksNeD1SJT.

Nikhil Ghosh, Denny Wu, and Alberto Bietti. Understanding the mechanisms of fast hyperparameter transfer, 2025. URL https://arxiv.org/abs/2512.22768.

Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OcKMT-36vUs.

Moritz Haas, Jin Xu, Volkan Cevher, and Leena Chennuru Vankadara. $\boldsymbol{\mu}\mathbf{P^2}$: Effective sharpness aware minimization requires layerwise perturbation scaling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=pR5g1bBqoV.

Moritz Haas, Sebastian Bordt, Ulrike von Luxburg, and Leena Chennuru Vankadara. On the surprising effectiveness of large learning rates under standard width scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=hTxnm6H93P.

Soufiane Hayou. A proof of learning rate transfer under μp. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026. URL https://openreview.net/forum?id=sZHpz3DHPj.

Soufiane Hayou and Liyuan Liu. Optimal embedding learning rate in llms: The effect of vocabulary size, 2025. URL https://arxiv.org/abs/2506.15025.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=aVh9KRZdRk.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Tianze Jiang, Blake Bordelon, Cengiz Pehlevan, and Boris Hanin. Hyperparameter transfer with mixture-of-expert layers, 2026. URL https://arxiv.org/abs/2601.20205.

Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=NVl4SAmz5c.

Dayal Singh Kalra and Maissam Barkeshli. Learning rates do not transfer across double descent. In *High-dimensional Learning Dynamics 2026*, 2026. URL https://openreview.net/forum?id=tLix7nN1XW.

Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. Universal sharpness dynamics in neural network training: Fixed point analysis, edge of stability, and route to chaos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VZN0irKnl0.

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, 2020. URL https://arxiv.org/abs/2001.08361.

Dhruva Karkada, Daniel J. Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. Symmetry in language statistics shapes the geometry of model representations, 2026. URL https://arxiv.org/abs/2602.15029.

Andrej Karpathy. nanoGPT: The simplest, fastest repository for training/finetuning medium-sized gpts. https://github.com/karpathy/nanoGPT, 2022.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than μp for learning rate transfer in practice. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PvTxIdZc1E.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Houyi Li, Wenzhen Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining, 2025. URL https://arxiv.org/abs/2503.04715.

Etai Littwin and Greg Yang. Adaptive optimization in the ∞-width limit. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=zgVDqw9ZUES.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Jan Małaśnicki, Kamil Ciebiera, Mateusz Boruń, Maciej Pióro, Jan Ludziejewski, Maciej Stefaniak, Michał Krutul, Sebastian Jaszczur, Marek Cygan, Kamil Adamczewski, and Jakub Krajewski. μ-parametrization for mixture of experts, 2025. URL https://arxiv.org/abs/2508.09752.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto. Super consistency of neural network landscapes and learning rate transfer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=rgwhJ7INtZ.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2025.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4fSSqpk1sM.

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales, 2025. URL https://arxiv.org/abs/2512.05620.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lina, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Daniel A. Roberts, Sho Yaida, and Boris Hanin. *Frontmatter*, pp. i–iv. Cambridge University Press, 2022.

Boris Shigida, Boris Hanin, and Andrey Gromov. Learning rate transfer in normalized transformers, 2026. URL https://arxiv.org/abs/2604.27077.

Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

Tao Tao, Darshil Doshi, Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. (how) can transformers predict pseudo-random numbers? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=asDx9sPAUN.

Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher. On feature learning in structured state space models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=aQv5AbN1wF.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Sho Yaida. Meta-principled family of hyperparameter scaling strategies, 2022. URL https://arxiv.org/abs/2210.04909.

Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yang21c.html.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Bx6qKuBM2AD.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=17pVDnpwwl.

A Related Works

Our work is closely related to several recent works on hyperparameter transfer (Everett et al., 2024; Ghosh et al., 2025; Kosson et al., 2026; Bergsma et al., 2025). Kosson et al. (2026) argue that weight decay stabilizes feature learning and is central to learning rate transfer. Our three transfer metrics provide complementary insights into the role of weight decay: its primary effect is to improve the loss predictability error $\mathcal{E}$, but it comes at the cost of increasing the asymptotic loss gap $\mathcal{R}(\infty)$ in the fixed step setting. Furthermore, very stable parameterizations such as SP+Embd+Attn achieve reliable transfer even without weight decay, and weight decay alone is insufficient to stabilize unstable parameterizations such as SP+LN. The analysis of Bergsma et al. (2025) implies that $\eta \cdot \lambda$ should scale as $\Theta(1/n^2)$ in the compute-optimal setting. Our preliminary experiments in this regime suggest that additional scaling considerations are needed, leaving the appropriate weight decay scaling as an open question. Hayou & Liu (2025) analyze the optimal embedding learning rate when the vocabulary size is large compared to width. By comparison, we find that the embedding learning rate accounts for μP's transfer quality at a fixed vocabulary size.

B Experimental Details

We pre-trained GPT-style Transformers on FineWeb-Edu (Penedo et al., 2024), building on the nanoGPT codebase (Karpathy, 2022). All experiments use a depth of 12 Transformer blocks without biases, a context length of 1024, with the data tokenized using the GPT-2 tokenizer (Radford et al., 2019), resulting in a vocabulary size of $50,304$. We scale the embedding dimension (width) $n \in [128, 2048]$ by increasing the number of heads, while keeping the head dimension fixed at $d = 64$. We train the models using AdamW ($\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-8}$) with a Warmup-Stable-Decay (WSD) schedule (Hu et al., 2024) (20% warmup, 60% stable, 20% decay). For each width, we sweep the peak learning rate and weight decay strength λ.

In the fixed-step setting, we train the models for $10,000$ steps with a batch size of 1024 ($\approx$ 1M tokens per step), corresponding to a total of approximately 10B tokens. By comparison, in the compute-optimal setting (fixed token-per-parameter), we scale training tokens proportional to the number of parameters, with a ratio of 20 tokens per parameter following Hoffmann et al. (2022). To ensure that small models are trained for a sufficient number of steps, we reduce the batch size to 256 ($\approx$ 0.25M tokens).

Table 1: Comparison of Standard (SP) and Maximal Update Parameterization (μP) for AdamW. Weight decay is scaled with the learning rate so that its contribution to the weight update, $\eta \cdot \lambda \cdot \theta$, remains $\Theta(1)$ across widths.

Parameterization	Layer	Multiplier (n^{-a})	Variance (n^{-2b})	LR (n^{-c})	WD (n^{-d})
Standard (SP)	Embedding	1	1	$1/n$	n
	Hidden	1	$1/n$	$1/n$	n
	Last	1	$1/n$	$1/n$	n
	LayerNorm	–	–	$1/n$	–
	Attention scale	$1/\sqrt{d}$	–	–	–
μP	Embedding	1	1	1	1
	Hidden	1	$1/n$	$1/n$	n
	Last	1	$1/n^2$	$1/n$	n
	LayerNorm	–	–	1	–
	Attention scale	$1/d$	–	–	–

Layerwise constants. The $abcd$ parameterization specifies the layerwise hyperparameter scaling with width through the exponents. However, each layer can additionally carry a width-independent constant. We can write these explicitly as

$$\eta_l = \eta\, r_l\, n^{-c_l}, \qquad \sigma_l^2 = s_l\, n^{-2b_l}, \tag{10}$$

where r_l is a per-layer learning-rate multiplier and s_l a per-layer initialization-variance multiplier. Unless stated otherwise, we set $r_l = s_l = 1$ for all layers, so that parameterizations differ only in their exponents. We examine the effect of tuning these constants in Section C.

Compute Usage. All experiments were run on H100 GPUs. The total sweep covers 20 learning rates $\times$ 8 weight decay values $\times$ 8 widths $\times$ 16 parameterizations $\times$ 2 training regimes, with each run taking approximately 2 hours on average, for an estimated total of $\sim$160,000 H100 GPU hours.

C Effect of Tuning Layerwise Constants

We now examine the effect of tuning the layerwise constants r_l and s_l introduced in Section B. At a given width, tuning these constants can result in the same overall learning rate η_ℓ and multiplier σ_ℓ^2. The difference across parameterizations appears on scaling the width. We therefore tune the constants at a small base width, freeze them, and then scale up while sweeping only the global learning rate. Concretely, we ran coordinate descent over η and the six constants $\{r_{\text{emb}}, r_{\text{last}}, r_{\text{LN}}, s_{\text{emb}}, s_{\text{hidden}}, s_{\text{last}}\}$ at $n = 128$, and evaluated the selected constants at $n \in \{512, 768, 1024\}$. For these constant-tuning experiments, we used a weight decay of 0.001.

At the base width (where constants are tuned), tuning changed μP and SP+Embd only marginally ($4.182 \to 4.162$ and $4.070 \to 4.034$), indicating that the default constants are nearly optimal. Tuning SP improved it considerably, from 4.287 to 4.040, and the improvement was almost entirely due to the embedding-layer learning-rate multiplier r_{emb}, which increased from 1 to 16. In other words, when SP is allowed to tune its constants, it increases the embedding-layer learning rate, and tuned SP approximately reproduces SP+Embd. Table 2 reports the final test loss at the best global learning rate once the constants are frozen and the width is increased.

A few observations follow. First, SP with default constants remains 0.10–0.15 behind μP at every width, while both SP+Embd and SP with a tuned r_{emb} stay within 0.04. This suggests that constant tuning does not affect the conclusion that the embedding-layer learning rate is the dominant factor. Second, SP's improvement after constant tuning is width-specific by construction: a frozen r_{emb} leaves the effective embedding-layer learning rate scaling as r_{emb}/n, which decays with width, whereas μP and SP+Embd fix it through the exponent and keep it $\Theta(1)$. A constant chosen at one width must therefore lose its effect at sufficiently large widths, even though the range $n \leq 1024$ studied here is too narrow to show this directly. Finally, these results illustrate that tuning constants

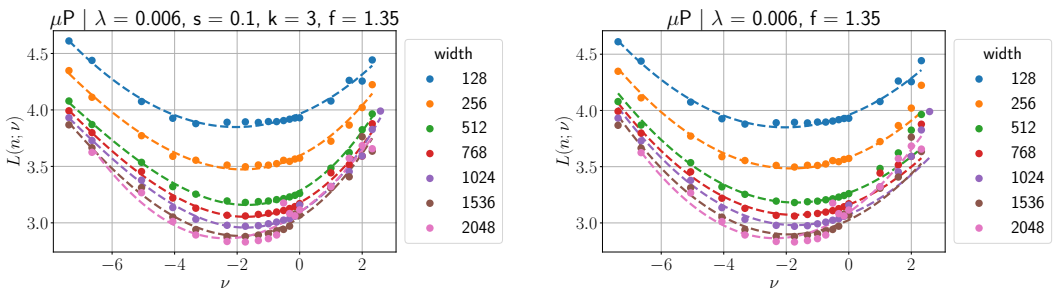
Table 2: Final test loss at the best global learning rate, with layerwise constants tuned at $n = 128$ and then held fixed. "SP ($r_{\text{emb}} = 16$)" modifies only the embedding-layer learning-rate multiplier; "tuned" refers to the full coordinate-descent solution.

Parameterization	$n = 512$	$n = 768$	$n = 1024$
μP (default constants)	3.154	3.017	2.926
μP (tuned)	3.187	3.072	2.984
SP+Embd (default constants)	3.169	3.050	2.966
SP+Embd (tuned)	3.123	3.002	2.904
SP (default constants)	3.277	3.170	3.034
SP ($r_{\text{emb}} = 16$)	3.148	3.039	2.947

at a small base width can overfit finite-width effects: the constants selected for μP at $n = 128$ are slightly worse than the default at every larger width.

D Estimating the Three Transfer Metrics

This appendix describes the filtering, interpolation, and fitting procedures used to compute the transfer framework metrics $\mathcal{R}(\infty), \mathcal{E}$, and κ.

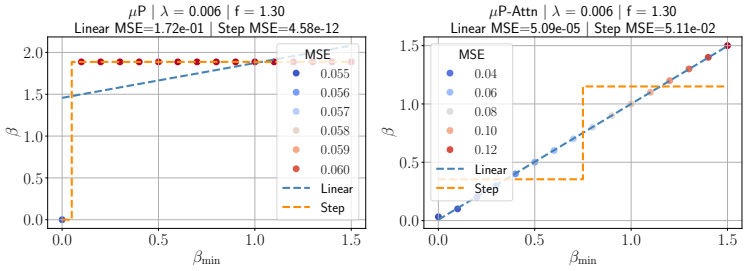


(a) (b)

Figure 6: (a) Interpolated loss curves for μP across widths with loss filtering threshold $f = 1.35$. Raw observed points (circles) and the fitted smoothing spline (solid lines) are shown for each width. (b) Per-width curvature fits for μP. For each width, we show the interpolated loss curve (solid line) and the fitted centered quadratic $L(\nu) = L_{\min} + \frac{1}{2}H(n)(\nu - \nu^*)^2$ (dashed line).

Filtering, Smoothing and Interpolation. For each width, we retain runs with loss within $f = 1.35\times$ the per-width optimal loss $L^*(n)$ to focus the analysis on the landscape around the minimum. The threshold f is chosen to avoid unstable and divergent runs while retaining as many data points as possible. We then fit a cubic spline (`UnivariateSpline`, degree $k = 3$) to the filtered per-width curves with smoothing parameter $S = s \cdot N \cdot \mathrm{Var}(L)$, where $s = 0.1$ is the base smoothing coefficient, N is the number of points, and $\mathrm{Var}(L)$ is the variance of the loss values. Each spline is evaluated on a uniform grid of 400 points spanning the observed ν range, resulting in smooth, dense curves for the subsequent analysis (see Figure 6a for an example). Out of 128 total combinations (16 parameterizations $\times$ 8 weight decay values), 4 failed this procedure either due to too few points after the filtering step or the interpolated loss curve exhibiting excessive noise resulting in negative loss values, and were therefore excluded from the analysis.

Estimating the Three Transfer Metrics. By fitting scaling laws to $L^*(n)$ and $\nu^*(n)$, we obtain the exponents α, β and the asymptotic loss $L^*(\infty)$, from which we compute $\mathcal{R}(\infty)$. The loss scaling law is fit in log space with constraints $L^*(\infty), A, \alpha \geq 0$, ensuring the fitted scaling law strictly decreases with width and converges to a finite irreducible value. The log-learning-rate scaling law is fit in linear space, since ν is already in log space, with $\beta \geq 0$ enforced to ensure convergence to the asymptotic value. Since $\mathcal{R}(\infty)$ is non-negative by definition (Definition 3.1), we clamp fitted



(a) (b)

Figure 7: Fitted β as a function of the lower bound $\beta_{\min}$ for two cases. Left: a degenerate case (μP) where the step function better fits the observed trend, and we adopt the optimal solution with $\beta > \beta^*_{\min}$. Right: a genuine case (μP-Attn) where β monotonically increases with $\beta_{\min}$, suggesting that the fitted small β can be trusted.

values to zero when they are negative due to finite-size fitting artifacts. For the curvature $H(n)$, we first fit a centered quadratic $L(\nu) = L^* + \frac{1}{2}H(n)(\nu - \nu^*(n))^2$ to the interpolated per-width curves (see Figure 6b), with the center fixed at $\nu^*(n)$ to prevent noise in the loss curve from shifting the fitted center away from the true optimum. We then fit the scaling law $H(n) = Cn^\gamma$ (Figure 8(f)) to obtain the exponent γ. Combined with α and β from above, this gives $\kappa = \alpha - 2\beta + \gamma$. Finally, we jointly fit all the parameters of Equation (6) using the interpolated curves, and evaluate the fit on the raw filtered data to obtain the loss predictability error $\mathcal{E}$.

We fit all scaling laws using a Huber loss objective with $\delta = 10^{-3}$ to improve robustness to outliers and 200 random initializations to avoid local minima in the non-convex scaling law fits. We additionally cap all scaling exponents at 2.0, which serves as a practical proxy for exponents running to infinity and prevents spuriously large values that can make comparisons across parameterization unreliable.

Degeneracy in ν^* Scaling. A subtle issue arises when $\nu^*(n)$ is nearly constant across widths. In this regime, the scaling law $\nu^*(n) = \nu^*(\infty) + Bn^{-\beta}$ admits two degenerate solutions: (1) $\beta \to 0$, reducing the model to a constant $\nu^*(\infty) + B$ and (2) $\beta \to \infty$, indicating rapid convergence to $\nu^*(\infty)$. We prefer the $\beta \to \infty$ solution, as $\beta \to 0$ reduces the scaling law to a constant and is inconsistent with the convergence-based interpretation of our framework. Since β may not be exactly zero numerically, distinguishing a genuinely small β from the degenerate $\beta \to 0$ solution is non-trivial. To resolve this, we repeatedly fit the scaling law with an increasing lower bound $\beta_{\min}$ on β and track how the fitted β changes. If the true solution is degenerate, the fitted β will exhibit a sudden jump at some $\beta^*_{\min}$, beyond which random initializations begin to prefer the $\beta \to \infty$ solution. In contrast, if the solution is not degenerate, the fitted β will linearly increase from $\beta_{\min}$ (see Figure 7). We then fit both a step function and a linear function to the resulting β vs. $\beta_{\min}$ trend. If the step function fits better, we have identified the small β as a degenerate solution and select the best solution with $\beta > \beta^*_{\min}$. Otherwise, we treat the small observed β as genuine. We apply this procedure only to the $\nu^*(n)$ scaling law fits to estimate β and hence κ.

Degeneracy in the Full Model. The full loss model (Equation (6)) introduces additional degeneracies beyond those in the individual ν^* scaling law. Specifically, the term $\frac{1}{2}Cn^\gamma(\nu - \nu^*(\infty) - Bn^{-\beta})^2$ couples the curvature scaling parameters C and γ with the ν scaling parameters B and β through the product $Cn^\gamma \cdot B^2 n^{-2\beta}$, making individual parameter estimates unreliable even when the overall fit is good. We therefore do not use the joint fit for estimating β, κ, and $\mathcal{R}(\infty)$, and instead rely on the individually fitted scaling laws. The parameters of the full model are constrained in the same way as the individual scaling laws, with $L^*(\infty), A, \alpha, C, \beta \geq 0$ and B unconstrained. Nevertheless, the joint fit consistently yields smaller fit errors than substituting the individually fitted parameters directly into Equation (6), and we therefore use it for reporting $\mathcal{E}$.

E TRANSFER METRICS FOR ADDITIONAL PARAMETERIZATIONS

In this section, we analyze the three transfer metrics for all parameterizations interpolating between SP and μP in further detail. Figure 8–17 show the loss and scaling law curves for each parameterization. We use '$+$' to denote incremental changes from SP towards μP and '$-$' to denote changes from μP towards SP. For each parameterization, we select the weight decay value that best represents its typical behavior.

E.1 EFFECT OF DIFFERENT LAYER TYPES ON TRANSFER METRICS

SP vs. μP. As described in Section 4, μP and SP are surprisingly similar across most metrics (Figures 8 and 9). Despite being more unstable (a $3\times$ larger loss predictability error), SP exhibits a large negative robustness exponent $\kappa = -3.505$. SP thus has the right ingredients for reliable transfer but is held back by training instability.

Embedding Layer Learning Rate. The embedding layer learning rate has the most pronounced effect on stability. Training SP with a $\Theta(1)$ embedding learning rate (SP+Embd) eliminates the instabilities, resulting in smooth loss curves (Figure 10). Conversely, training μP with a $\Theta(1/n)$ embedding learning rate (μP-Embd) destabilizes training entirely (Figure 11).

Attention Scaling. The effect of $1/d$ attention scaling is more subtle. Training SP with $1/d$ attention scaling (SP+Attn) does not improve stability (Figure 12): the loss predictability error $\mathcal{E}$ remains similar to SP, $\mathcal{R}(\infty)$ is slightly worse, and instabilities worsen at large widths. Most notably, κ becomes positive ($\kappa = 0.509$), indicating that transfer becomes brittle. A similar degradation is observed when training μP with $1/\sqrt{d}$ attention scaling (μP-Attn) (Figure 13): $\mathcal{E}$ is similar to SP, the optimal learning rate decreases with width, and $\kappa = 0.881$, making transfer unreliable. These results suggest that the attention scaling has a more subtle effect depending on the scaling of other layers.

LayerNorm Learning Rate. Increasing the LayerNorm learning rate to $\Theta(1)$ in SP (SP+LN) severely destabilizes training (Figure 14), with $\mathcal{E}$ roughly $10\times$ larger than SP. Interestingly, while the loss at small widths is poor, SP+LN achieves the best asymptotic loss across all parameterizations at large widths. This highlights a sharp tradeoff: SP+LN can in principle reach excellent performance at scale, but its unpredictable loss landscape makes transfer unreliable—a high-reward but high-risk parameterization. By comparison, decreasing the LayerNorm learning rate to $\Theta(1/n)$ in μP (μP-LN) has surprisingly almost no effect. The loss predictability error $\mathcal{E}$ is slightly better than μP, while the asymptotic loss is slightly worse. The consistent improvement in loss predictability from slowing down LayerNorm training observed in both SP+LN and μP-LN suggests that LayerNorm parameters are best trained slowly. This again reflects a tradeoff: slower LayerNorm training stabilizes the training dynamics at the cost of a small performance penalty.

Last Layer Initialization. Finally, reducing the variance of the last-layer initialization to $\Theta(1/n^2)$ in SP has a negligible effect, with all three metrics remaining comparable to SP. By comparison, increasing it to $1/n$ in μP (μP-Last) causes training instabilities at small widths, but transfer remains robust at larger widths. The minimal role of last layer initialization is consistent with the observation that a larger initialization variance leads to higher Hessian sharpness, but learning rate warmup gradually reduces this sharpness early in training (Kalra & Barkeshli, 2024), effectively mitigating any initialization differences.

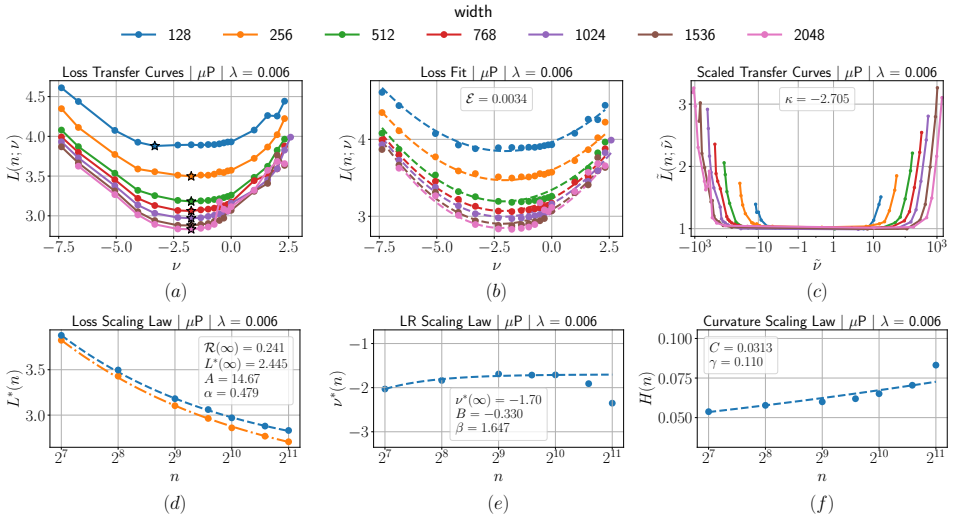


Figure 8: Transfer metrics for μP with weight decay $\lambda = 0.006$. Panels (a-c) repeat Figure 1; (d-f) show the scaling laws for optimal loss $L^*(n)$, optimal log-learning-rate $\nu^*(n)$, and curvature $H(n)$. In (d), the orange curve shows the best loss across parameterizations at each width, used for estimating the asymptotic loss gap $\mathcal{R}(\infty)$.

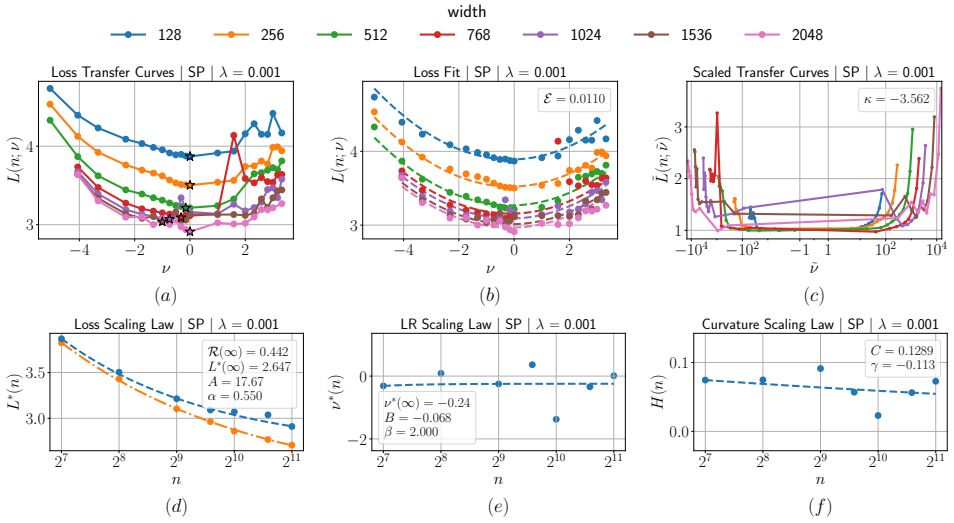


Figure 9: Transfer metrics for SP with weight decay $\lambda = 0.001$.

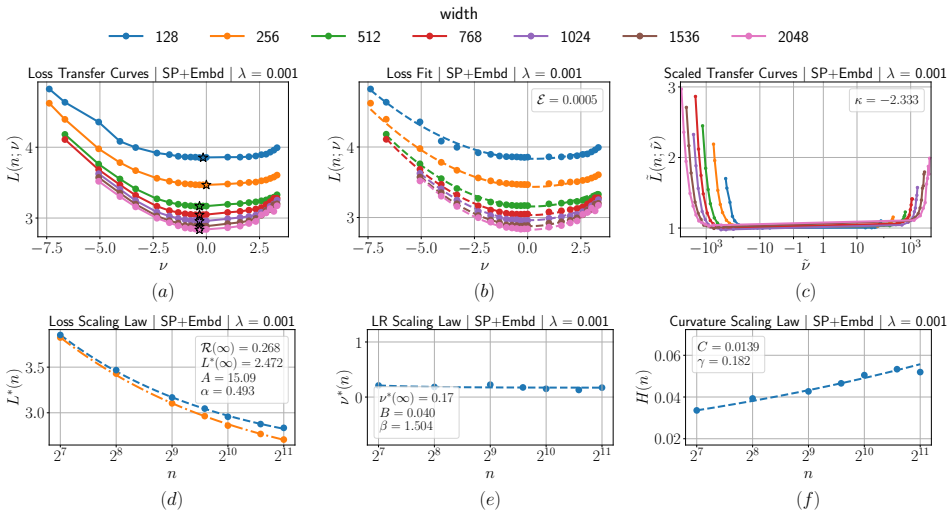


Figure 10: Transfer metrics for SP+Embd with weight decay $\lambda = 0.001$.

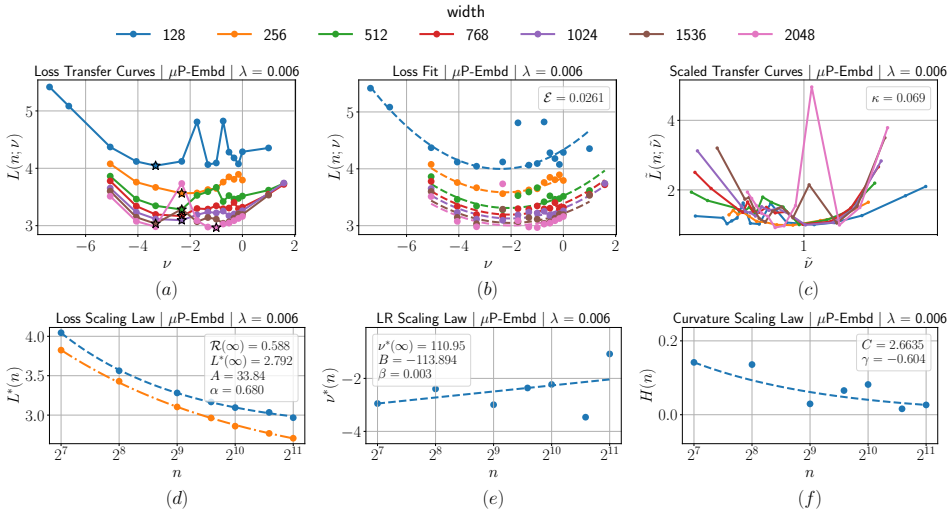


Figure 11: Transfer metrics for μP-Embd with weight decay $\lambda = 0.006$.

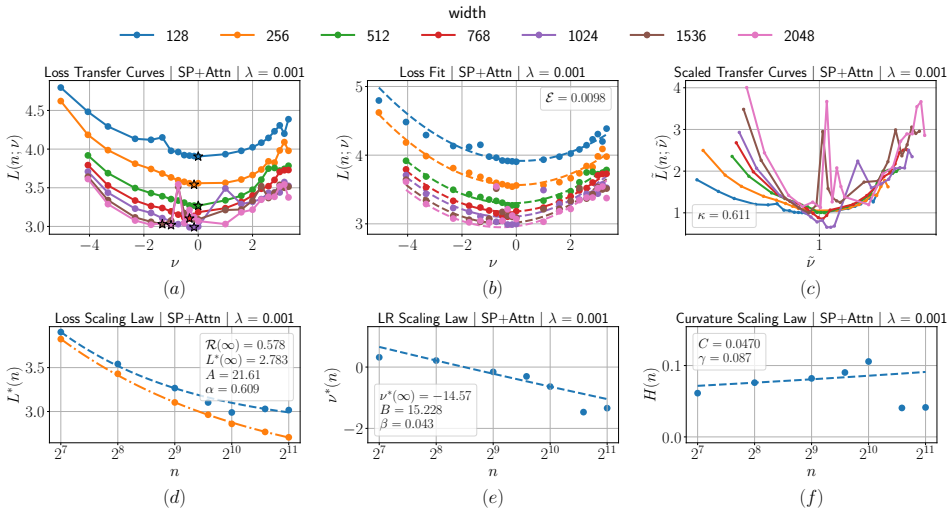


Figure 12: Transfer metrics for SP+Attn with weight decay $\lambda = 0.001$.

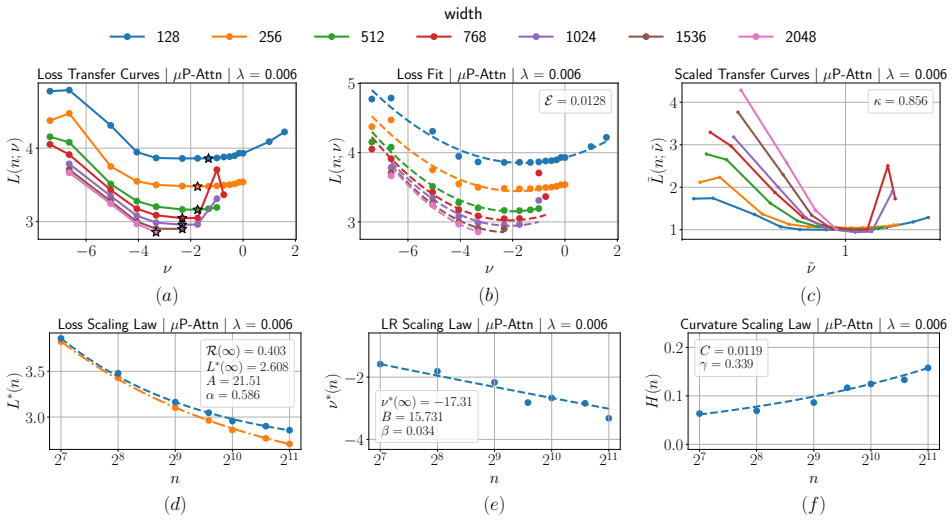


Figure 13: Transfer metrics for μP-Attn with weight decay $\lambda = 0.006$.

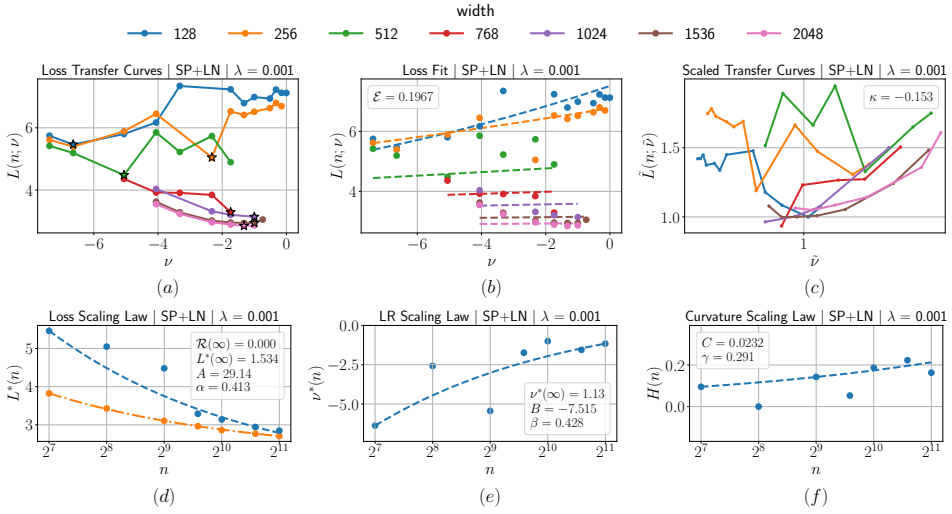


Figure 14: Transfer metrics for SP+LN with weight decay $\lambda = 0.001$.

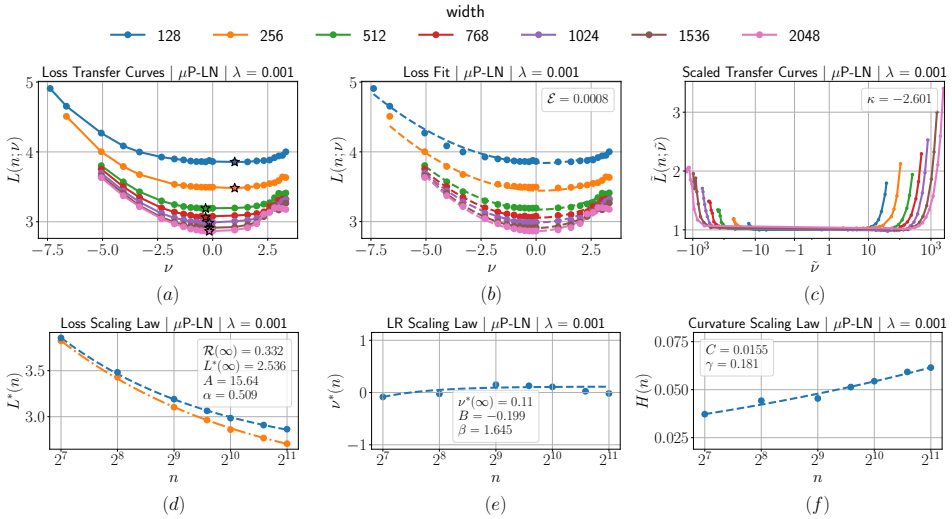


Figure 15: Transfer metrics for μP-LN with weight decay $\lambda = 0.001$.

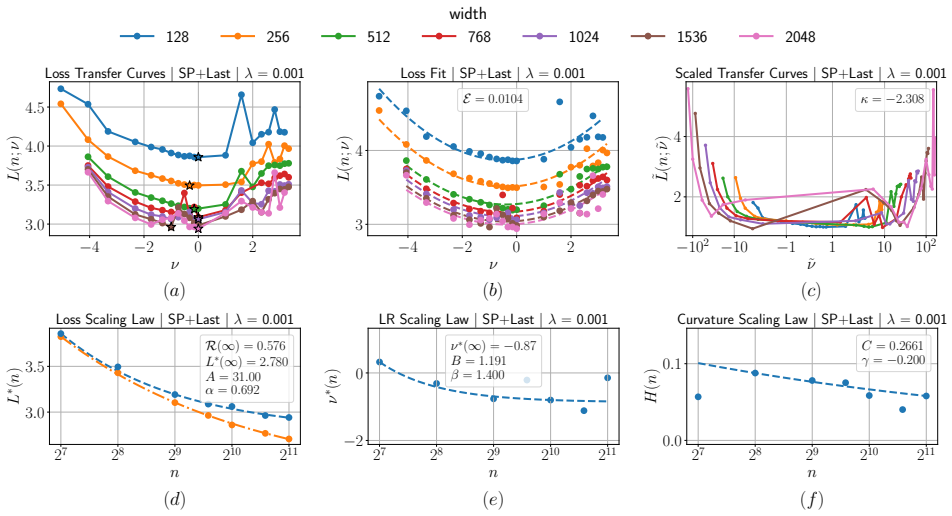


Figure 16: Transfer metrics for SP+Last with weight decay $\lambda = 0.001$.

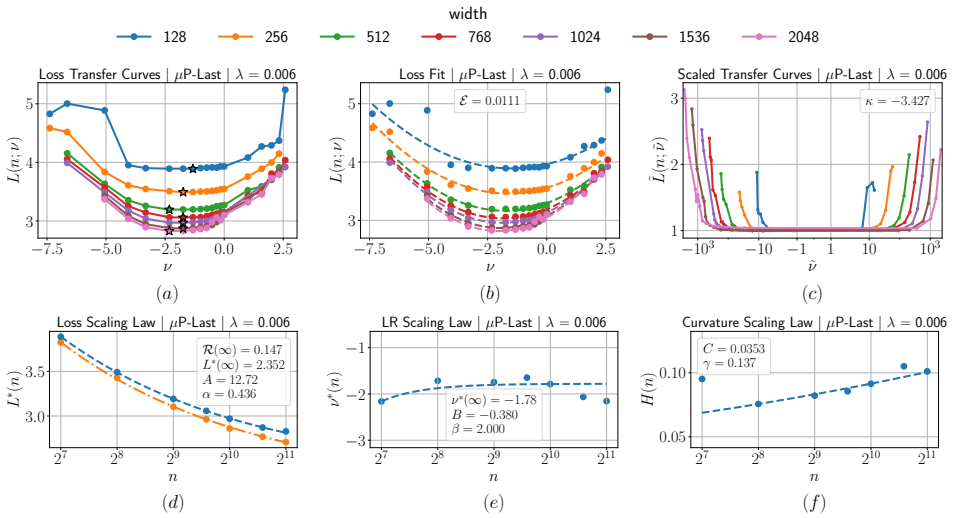


Figure 17: Transfer metrics for μP-Last with weight decay $\lambda = 0.006$.

E.2 Transfer Metric Phase Diagrams

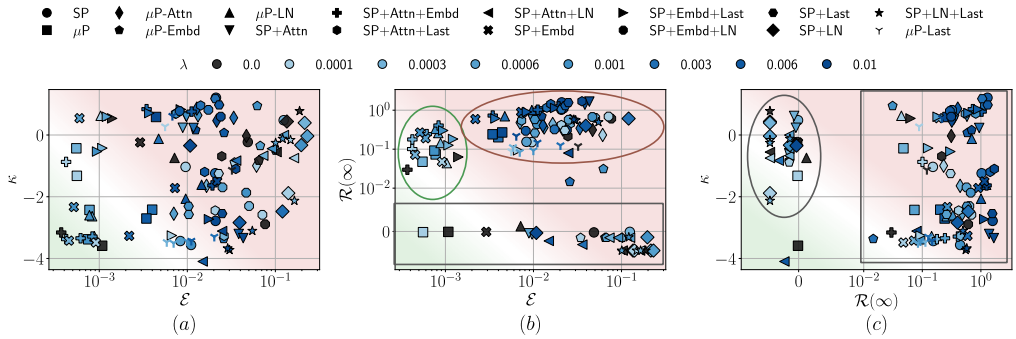


Figure 18: *Transfer metrics for parameterizations interpolating between SP and μP.* Parameterizations with '+' denote incremental changes from SP towards μP, while '−' denotes changes from μP towards SP. Green and red regions indicate desirable and undesirable regimes, respectively.

Figure 18 shows the three transfer metrics for all 16 parameterizations and 8 weight decay values. The metrics naturally separate parameterizations into clusters. For example, in panel (b), parameterizations that train the embedding well (μP, SP+Embd, μP-LN, SP+Attn+Embd) cluster in the desirable low $\mathcal{E}$, low $\mathcal{R}(\infty)$ region, while unstable parameterizations (SP, SP+LN, μP-Embd) achieve near-zero $\mathcal{R}(\infty)$ but at the cost of high $\mathcal{E}$. This illustrates the tradeoff between the metrics noted in Section 4: a parameterization can achieve excellent asymptotic performance while remaining unreliable for transfer. Large weight decay values are located in the top right, improving $\mathcal{E}$ relative to the unstable cluster but at an asymptotic performance cost, never reaching the stable cluster. Similar clustering patterns emerge in panels (a) and (c), with stable parameterizations consistently occupying the desirable regions across all three metrics.

F Importance of Embedding Layer Learning Rate in LLaMA-style Transformers with RoPE

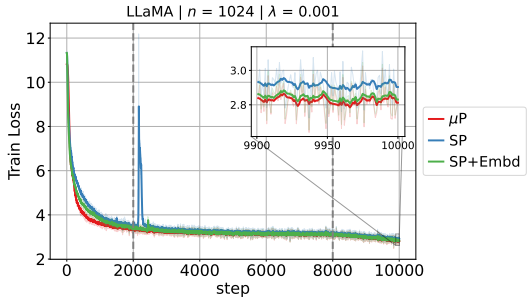


Figure 19: Train loss vs. step for a 12-layer LLaMA-style Transformer ($n = 1024$) with RoPE, RMSNorm and SwiGLU, trained for 10^4 steps. Each parameterization is shown at its own optimal learning rate, which is $\nu^* = 0$ for all three. Faint curves show the raw training loss and solid curves a 10-step moving average; dashed lines mark the end of warmup and the start of decay; the inset zooms the final 100 steps.

G Importance of the First Layer Learning Rate in CNN Image Classification

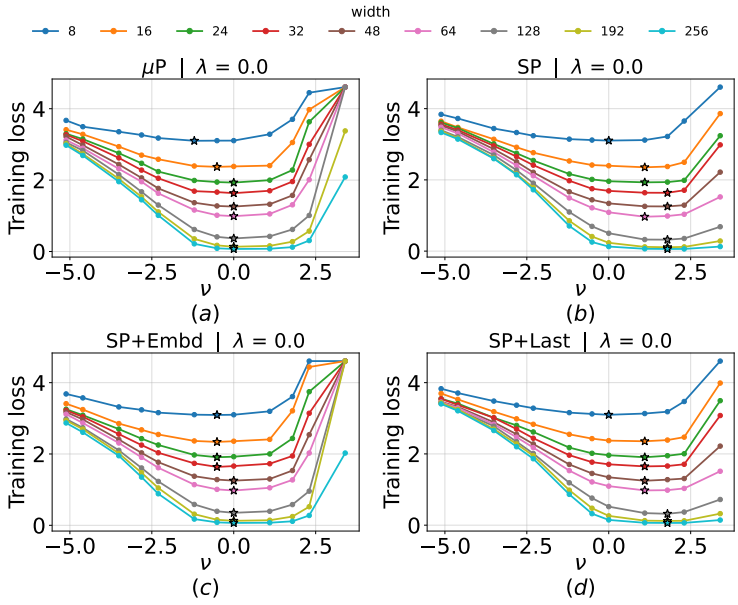


Figure 20: *CNN experiments on CIFAR with Adam.* Training loss as a function of log learning rate ν for four parameterizations across widths. In SP, the optimal learning rate drifts with width, while μP shows substantially less drift. Increasing the learning rate of the input-facing layer in SP (SP+Embd) largely removes this drift and places the optimum in a similar region to μP. By contrast, changing only the last-layer initialization (SP+Last) leaves both the learning-rate drift and the optimum location closer to SP. This suggests that the role of the embedding layer learning rate in Transformers reflects a more general importance of the input-facing layer learning rate.

In Section 5, we demonstrated that the embedding layer learning rate is the primary factor in explaining the difference between SP and μP in Transformers trained with Adam. A natural question is whether this result is specific to Transformers with an embedding layer or generalizes to other

architectures and tasks. To test this, we consider CNNs trained on CIFAR with Adam, where the first layer is a simple convolutional layer, and there is no LayerNorm or attention.

As there are only two differences in this setting (first and last layer), there are only four variants: μP, SP, SP+Embd, and SP+Last. Figure 20 shows the training loss as a function of log learning rate ν across widths. We observe that the optimal learning rate increases with width in SP, whereas in μP it remains fairly constant. Training SP with a $\Theta(1)$ first-layer learning rate largely removes this drift: SP+Embd behaves similarly to μP both in the location of the optimal learning rate and in its reduced drift. By contrast, changing only the last-layer initialization has little effect: SP+Last remains closer to SP, both in the location of the optimum and in the observed learning-rate drift.

These results suggest that the role of the embedding layer learning rate is not specific to Transformers, and that training the input layer sufficiently fast is important for learning-rate transfer under Adam.

H Weight Decay Scaling in Compute-Optimal Regime

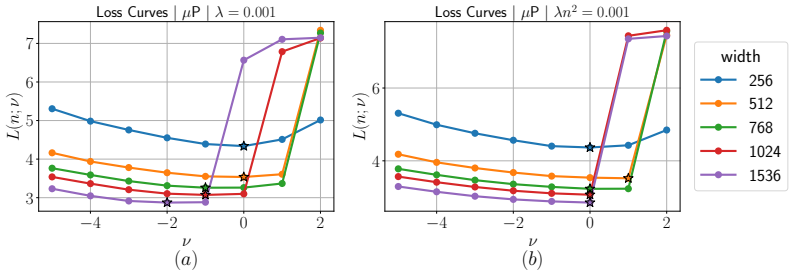


Figure 21: Loss curves for μP in the compute-optimal setting (20 TPP) under two weight decay scalings. (a) Standard μP convention $\eta \cdot \lambda = \Theta(1)$: $\nu^*(n)$ shifts to the left with increasing width. (b) Corrected convention $\eta \cdot \lambda = \Theta(1/n^2)$: $\nu^*(n)$ doesn't vary much, but the shape of the loss curves around the minimum changes across widths.

In the compute-optimal (fixed TPP) setting, the number of training steps scales as $T \propto n^2$, which violates μP's assumption of $\Theta(1)$ training steps compared to width. This makes the standard weight decay convention $\eta \cdot \lambda = \Theta(1)$ inadequate, as we observed in Section 6 where transfer robustness κ degrades with increasing weight decay. A natural scaling, motivated by Bergsma et al. (2025), is to scale weight decay strength as $\lambda = \Theta(1/n^2)$ so that the scale of weight decay's cumulative contribution over $T = \Theta(n^2)$ steps $\eta\lambda T$ remains $\Theta(1)$ across widths. That being said, we caution that this choice is not well motivated. There is no first-principle reason why the cumulative weight decay must be $\Theta(1)$, as different widths may benefit from different total contributions.

Figure 21 compares the loss curves for μP under two scaling conventions. Under $\eta \cdot \lambda = \Theta(1)$ (a), the optimal log learning rate $\nu^*(n)$ drifts noticeably to the left with increasing width, suggesting a slow convergence (small β). As a result, $\kappa = \alpha - 2\beta + \gamma$ becomes large, resulting in brittle transfer observed in Section 6. By comparison, under $\eta \cdot \lambda = \Theta(1/n^2)$ (panel b), this drift is visually reduced, suggesting an improvement in transfer robustness. However, the shape of the loss curves around the minimum changes across widths. We leave a systematic analysis of the appropriate weight decay scaling in the TPP regime to future work.

I The Effect of Learning Rate Warmup on Learning Rate Transfer

Learning rate warmup is standard practice in large-scale training (DeepSeek-AI et al., 2024; OLMo et al., 2025). Its primary effect is gradually reducing the sharpness of the loss Hessian (or preconditioned sharpness for Adam), effectively steering optimization towards well-conditioned regions of the loss landscape where the model can be trained at large learning rates (Gilmer et al., 2022; Kalra & Barkeshli, 2024). Despite its widespread use, its interaction with learning rate transfer has not been systematically studied. Figure 22 shows that short warmup durations (1-5%) result

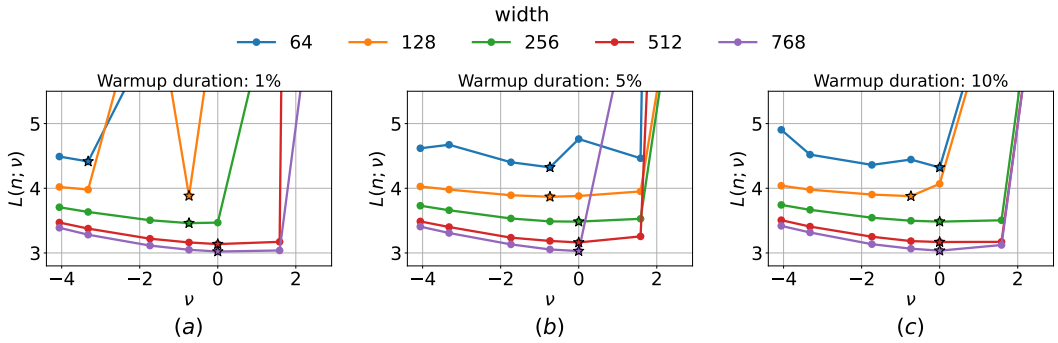


Figure 22: Learning rate warmup is essential for observing reliable learning rate transfer in μP.

in training instabilities that make transfer unreliable—the loss curves are noisy across widths, making extrapolation meaningless. At 10% warmup, the loss curves become noticeably smoother and $\nu^*(n)$ aligns more consistently across widths. This result confirms that warmup is crucial for observing reliable learning rate transfer, which is not accounted for in μP's theoretical derivation. We use a conservative warmup duration of 20% throughout experiments to observe reliable transfer.

J Effect of Freezing the Embedding Layer

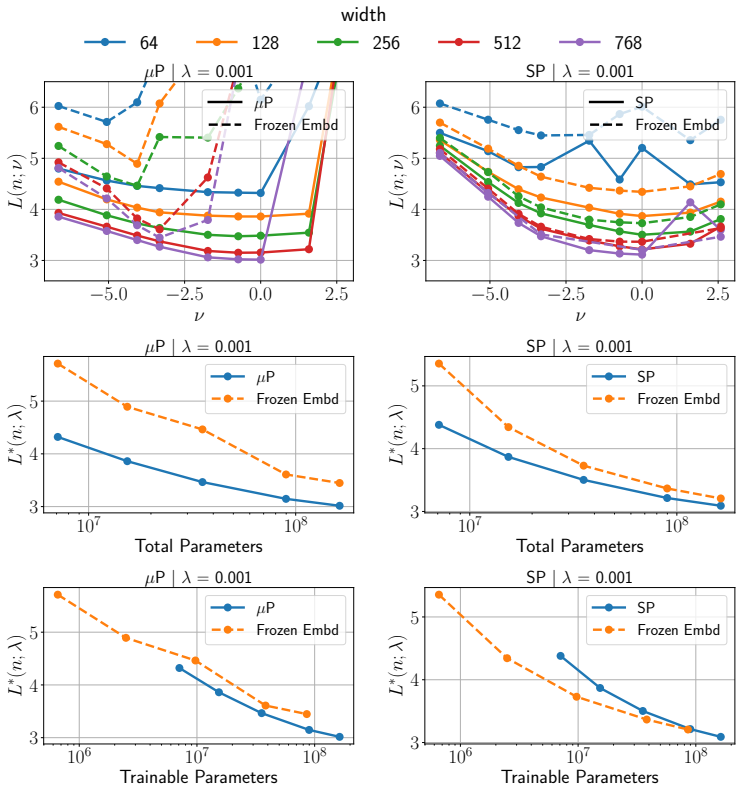


Figure 23: Effect of freezing the embedding layer in μP and SP. (top) Loss vs. log learning rate curves. (middle) Optimal loss vs. total parameters. (bottom) Optimal loss vs. trainable parameters, where trainable parameters exclude the embedding when frozen.

The switch experiments in Section 5 show that training the embedding layer too slowly can both slow down learning and cause training instabilities. Here, we take this further by studying the effect of completely freezing the embedding layer in both μP and SP.

Figure 23 (top) shows the loss vs. log learning rate curves. Freezing the embedding layer significantly affects μP, causing pronounced training instabilities and a shift in the optimal learning rate. SP is comparatively more robust: freezing causes instabilities at small widths, but the curves match the unfrozen case at large widths.

Figure 23 (middle) shows the optimal loss scaling laws against total parameters. For μP, freezing the embedding causes a large performance gap that narrows with width. For SP, the effect is more modest, with frozen and unfrozen performance converging at large widths.

However, comparing by total parameters is unfair, since at small widths the embedding dominates the parameter count (vocabulary size $\times$ width, with vocabulary size 50,304). We therefore also compare against trainable parameters, which excludes frozen embedding parameters (Figure 23, bottom). Even after this correction, freezing μP's embedding remains worse across all widths, though the gap narrows considerably. For SP, the frozen variant achieves lower loss at small widths, suggesting that non-embedding parameters might be more parameter-efficient at small scales, though the unfrozen case catches up at large widths. We leave a detailed understanding of this phenomenon to future work.

K MINIMAL DERIVATION OF μP

In this section, we present a self-contained derivation of scaling conditions for neural networks trained at large widths and show that μP arises as one particular solution to these conditions. We consider a minimal three-layer linear network. The choice is minimal for two reasons. First, three layers suffice to capture all distinct layer types in a neural network: input-to-hidden, hidden-to-hidden, and hidden-to-output. Second, standard activation functions used in practice (ReLU, GeLU, SiLU, tanh) act element-wise, so they contribute only $\Theta_n(1)$ factors to both the forward and backward passes and leave the width-scaling exponents unchanged. We build upon the analysis and notation of Everett et al. (2024), and extend the derivation to cover weight decay, LayerNorm, and attention scaling. Our goal here is to provide a simple, transparent derivation of the scaling rules, making every assumption explicit at the step where it is used.

K.1 NOTATION AND NORMS

In this derivation, we examine how every quantity (weights, activations, gradients, and updates) scales with the width n, using its RMS norm:

Definition K.1 (RMS norm)**.** For a vector $\mathbf{v} \in \mathbb{R}^n$, the RMS norm is defined as:

$$\|\mathbf{v}\|_{\mathrm{RMS}} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} v_i^2}.$$

For a matrix $M \in \mathbb{R}^{m\times n}$, we use the entrywise extension:

$$\|M\|_{\mathrm{RMS}} := \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} M_{ij}^2}.$$

Unless we state otherwise, we use $\|\cdot\|$ to denote the RMS norm throughout this derivation.

Definition K.2 (Width-scaling)**.** Let $\{\mathbf{v}_n\}$ be a family of vectors with $\mathbf{v}_n \in \mathbb{R}^n$. We say $\mathbf{v}_n = \Theta_n(n^\alpha)$ if there exist $C_1, C_2 > 0$ and N such that $C_1 n^\alpha \le \|\mathbf{v}_n\| \le C_2 n^\alpha$ for all $n \ge N$. The one-sided versions $\mathcal{O}_n$ and Ω_n keep only the upper or lower bound. Furthermore, for two families $\{\mathbf{u}_n\}$ and $\{\mathbf{v}_n\}$, we write $\mathbf{v}_n \sim \mathbf{u}_n$ to indicate that they share the same width-scaling, i.e., $\|\mathbf{v}_n\| = \Theta_n(\|\mathbf{u}_n\|)$. The subscript n specifies width as the scaling variable, and we will drop it whenever it's clear from context.

K.2 MODEL ARCHITECTURE

Consider a three-layer linear network with trainable parameters $\boldsymbol{\theta} := \{U, W, V\}$, width n, input dimension d_{in}, and output dimension d_{out}. For a training example $(\mathbf{x}, \mathbf{y})$, the forward pass is:

$$\begin{aligned} h(\mathbf{x}) &= n^{-a_u} U\mathbf{x}, \\ z(\mathbf{x}) &= n^{-a_w} W h(\mathbf{x}), \\ f(\mathbf{x}) &= n^{-a_v} V z(\mathbf{x}), \end{aligned} \tag{11}$$

with $U \in \mathbb{R}^{n\times d_{\mathrm{in}}}$, $W \in \mathbb{R}^{n\times n}$, and $V \in \mathbb{R}^{d_{\mathrm{out}}\times n}$. Here, the exponents $a := (a_u, a_w, a_v)$ scale the forward pass in each layer. The initialization variances are controlled by the exponents $b := (b_u, b_w, b_v)$:

$$U_{ij} \sim \mathcal{N}(0, n^{-2b_u}), \qquad W_{ij} \sim \mathcal{N}(0, n^{-2b_w}), \qquad V_{ij} \sim \mathcal{N}(0, n^{-2b_v}). \tag{12}$$

The per-layer learning rates are scaled by the exponents $c := (c_u, c_w, c_v)$:

$$\eta_u = \eta\, n^{-c_u}, \qquad \eta_w = \eta\, n^{-c_w}, \qquad \eta_v = \eta\, n^{-c_v}. \tag{13}$$

where η is the global learning rate. Finally, the per-layer weight decay strengths are scaled by the exponents $d := (d_u, d_w, d_v)$:

$$\lambda_u = \lambda\, n^{-d_u}, \qquad \lambda_w = \lambda\, n^{-d_w}, \qquad \lambda_v = \lambda\, n^{-d_v}, \tag{14}$$

where λ is the weight decay strength. The exponents $\{a, b, c, d\}$ collectively define our $abcd$ parameterization, which extends the abc parameterization of Yang & Hu (2021) with weight decay exponents d.

We use the subscript t to denote the training step. Thus, U_t, W_t, and V_t denote the parameters after t training steps, and $h_t(\mathbf{x})$, $z_t(\mathbf{x})$, and $f_t(\mathbf{x})$ denote the corresponding activations and network output. Initialization corresponds to $t = 0$. For any quantity q_t, we write $\Delta q_t := q_t - q_{t-1}$ to denote its change at step t.

K.3 Design Principles for Scaling Analysis

For training to remain *stable*, the activations and their updates must not vanish or explode as the width grows. If the activation updates vanish, the corresponding layer remains frozen at initialization; on the other hand, if they explode, training diverges. We formalize stability through the following two conditions (Yang & Hu, 2021).

Desideratum 1 (Stable Initialization)**.** The hidden activations and the network output remain stable with width at initialization:

$$\|h_0\| = \Theta_n(1), \qquad \|z_0\| = \Theta_n(1), \qquad \|f_0\| = \mathcal{O}_n(1). \tag{15}$$

The hidden activations h_0 and z_0 are required to be $\Theta_n(1)$ for stable signal propagation across layers (Roberts et al., 2022). The network output f_0 in contrast has a looser $\mathcal{O}_n(1)$ constraint because initializing the last layer with small weights does not affect signal propagation through the network (Yaida, 2022).

Desideratum 2 (Stable Updates at the First step)**.** At the first gradient step $t = 1$, the hidden activation and network output updates remain stable with width:

$$\|\Delta h_1\| = \Theta_n(1), \qquad \|\Delta z_1\| = \Theta_n(1), \qquad \|\Delta f_1\| = \Theta_n(1). \tag{16}$$

Here, we impose this condition at $t = 1$ so that the relevant computations are tractable. The same condition is desirable at every training step; however, extending it to general training step t requires further assumptions on weights, activations, and their alignment, which we introduce in Section K.13.

It is widely observed that parameterizations satisfying Desiderata 1 and 2 (e.g., μP) exhibit *learning rate transfer*: optimal learning rates at small width remain optimal at large widths. The implication, however, is non-trivial: ensuring that activations and their updates remain stable does not by itself imply that the optimal hyperparameters are width-independent, particularly in the limit where number of training steps is much larger than width. Several works support transfer under μP, though in restricted settings. The simplest setting is of Kalra et al. (2025), who showed that, for a two-layer linear network trained on a single example, the dynamical equations under μP have no width dependence whatsoever, directly implying hyperparameter transfer without any assumptions. A recent work by Hayou (2026) provided a formal proof of learning-rate transfer under μP in restricted settings of linear networks with input and output layers fixed at initialization. At the first gradient step, they provide an explicit limit and convergence rate; by comparison, at any fixed step t, they establish transfer under the additional assumption that the loss has a unique minimizer in η, but do not provide convergence rates as in the first step case. Beyond linear networks, Yaida (2022) showed that the leading-order terms in the function-space update equations are width-independent under μP, suggesting that the dynamics, and by extension the optimal hyperparameters, may also be width-stable in general neural networks. The subleading term, however, can accumulate over long training horizons and contribute to the optimal learning rate. Finally, Noci et al. (2024) empirically observed that the sharpness (largest Hessian eigenvalue) does not vary with width under μP, providing complementary evidence that the dominant loss landscape features remain invariant in μP. Proving learning rate transfer in generic settings remains an open question.

K.4 Conditions for Stable Initialization

In this section, we derive the conditions on $\{a, b\}$ required to satisfy Desideratum 1; the conditions on exponents $\{c, d\}$ will be imposed by the training dynamics in the later sections. At initialization, the weights and their inputs are independent, so the scale of each layer's output can be easily

computed by averaging over the weight distribution while treating the input as fixed. Since all the quantities in this section are at $t = 0$, we suppress the time index for brevity.

First layer. The forward pass of the first layer is:

$$h(\mathbf{x}) = n^{-a_u} U\mathbf{x}.$$

Let $\langle\cdot\rangle$ denote expectation over the weight distribution. The expected squared norm of $h(\mathbf{x})$ is:

$$\begin{aligned}\left\langle \|h(\mathbf{x})\|^2 \right\rangle = \left\langle \frac{1}{n}\sum_{i=1}^{n} h_i^2(\mathbf{x}) \right\rangle &= \frac{1}{n}\sum_{i=1}^{n} n^{-2a_u} \sum_{j,k=1}^{d_{\text{in}}} \langle U_{ij}U_{ik}\rangle\, x_j x_k \\ &= n^{-2a_u-1}\sum_{i=1}^{n}\sum_{j=1}^{d_{\text{in}}} n^{-2b_u}\, x_j^2 \qquad \left(\text{using } \langle U_{ij}U_{ik}\rangle = n^{-2b_u}\delta_{jk}\right) \\ &= n^{-2a_u-2b_u}\sum_{j=1}^{d_{\text{in}}} x_j^2. \end{aligned} \tag{17}$$

To handle the input term, we assume:

Assumption K.1 (Input scaling)**.** The inputs are normalized so that $\|\mathbf{x}\| = \Theta_n(1)$.

Under Assumption K.1, $\sum_{j=1}^{d_{\text{in}}} x_j^2 = \Theta_n(1)$, so

$$\left\langle \|h(\mathbf{x})\|^2 \right\rangle = \Theta_n(n^{-2a_u-2b_u}). \tag{18}$$

Requiring $\|h(\mathbf{x})\| = \Theta_n(1)$ yields:

$$\boxed{a_u + b_u = 0.} \tag{19}$$

Middle layer. The forward pass of the middle layer is:

$$z(\mathbf{x}) = n^{-a_w} W h(\mathbf{x}).$$

The expected squared norm of $z(\mathbf{x})$, with the expectation taken over W at fixed $h(\mathbf{x})$, is:

$$\begin{aligned}\left\langle \|z(\mathbf{x})\|^2 \right\rangle = \left\langle \frac{1}{n}\sum_{i=1}^{n} z_i^2(\mathbf{x}) \right\rangle &= \frac{1}{n}\sum_{i=1}^{n} n^{-2a_w} \sum_{j,k=1}^{n} \langle W_{ij}W_{ik}\rangle\, h_j(\mathbf{x})h_k(\mathbf{x}) \\ &= n^{-2a_w-1}\sum_{i=1}^{n}\sum_{j=1}^{n} n^{-2b_w}\, h_j^2(\mathbf{x}) \qquad \left(\text{using } \langle W_{ij}W_{ik}\rangle = n^{-2b_w}\delta_{jk}\right) \\ &= n^{-2a_w-2b_w+1}\,\|h(\mathbf{x})\|^2. \end{aligned} \tag{20}$$

Using $\|h(\mathbf{x})\| = \Theta_n(1)$ from the first layer, we have:

$$\boxed{a_w + b_w = \tfrac{1}{2}.} \tag{21}$$

Last layer. The forward pass of the last layer is:

$$f(\mathbf{x}) = n^{-a_v} V z(\mathbf{x}).$$

The calculation follows as in the middle layer, with V in place of W and the output dimension d_{out} replacing n in the output index.

$$\left\langle \|f(\mathbf{x})\|^2 \right\rangle = n^{-2a_v-2b_v+1}\,\|z(\mathbf{x})\|^2. \tag{22}$$

Using $\|z(\mathbf{x})\| = \Theta_n(1)$ and the one-sided requirement $\|f(\mathbf{x})\| = \mathcal{O}_n(1)$:

$$\boxed{a_v + b_v \geq \tfrac{1}{2}.} \tag{23}$$

Unlike the previous two layers, the last-layer initialization scale is not fixed by stability alone, as any $a_v + b_v \geq 1/2$ satisfies the constraint.

The conditions above fix the weight and activation scales only at initialization, and the weights and activations can evolve in a width-dependent manner during training. We will discuss the implications in the later sections.

K.5 Gradient Scaling at Initialization

Given a dataset $\mathcal{D} = \{(\mathbf{x}^\mu, \mathbf{y}^\mu)\}_{\mu=1}^D$ and a per-example loss function $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathbb{R}^{d_{\text{out}}} \to \mathbb{R}$, we define the training loss as the empirical average:

$$L(\boldsymbol{\theta}) = \frac{1}{D} \sum_{\mu=1}^{D} \ell(f(\mathbf{x}^\mu), \mathbf{y}^\mu) \,. \tag{24}$$

The per-layer gradients are:

$$\begin{aligned} g_v &:= \nabla_v L(\boldsymbol{\theta}) = \frac{1}{D} \sum_{\mu=1}^{D} n^{-a_v} \nabla_f \ell^\mu \cdot z(\mathbf{x}^\mu)^\top, \\ g_w &:= \nabla_w L(\boldsymbol{\theta}) = \frac{1}{D} \sum_{\mu=1}^{D} n^{-a_w - a_v} V^\top \nabla_f \ell^\mu \cdot h(\mathbf{x}^\mu)^\top, \\ g_u &:= \nabla_u L(\boldsymbol{\theta}) = \frac{1}{D} \sum_{\mu=1}^{D} n^{-a_u - a_w - a_v} W^\top V^\top \nabla_f \ell^\mu \cdot \mathbf{x}^{\mu\top}, \end{aligned} \tag{25}$$

where $\nabla_f \ell^\mu := \nabla_f \ell(f(\mathbf{x}^\mu), \mathbf{y}^\mu)$. Next, we assume that the loss derivative does not scale with width:

Assumption K.2 (Loss derivative scaling)**.** For every example $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in \mathcal{D}$, the per-example loss derivative satisfies $\|\nabla_f \ell^\mu\| = \Theta_n(1)$.

We first analyze the per-example terms. For g_v, the per-example term $\nabla_f \ell^\mu \cdot z(\mathbf{x}^\mu)^\top$ is an outer product of two vectors with $\Theta_n(1)$ entries, so:

$$\|n^{-a_v} \nabla_f \ell^\mu z(\mathbf{x}^\mu)^\top\| = n^{-a_v} \|\nabla_f \ell^\mu\| \|z(\mathbf{x}^\mu)\| = \Theta_n(n^{-a_v}). \tag{26}$$

For g_w, the per-example term is the outer product $(V^\top \nabla_f \ell^\mu) h(\mathbf{x}^\mu)^\top$. The left factor is a matrix-vector product along the output dimension d_{out} and does not pick up a width scaling factor. Therefore, we can write:

$$\langle \|V^\top \nabla_f \ell^\mu\|^2 \rangle \sim n^{-2b_v} \, \|\nabla_f \ell^\mu\|^2 = \Theta_n(n^{-2b_v}),$$

This gives $\|V^\top \nabla_f \ell^\mu\| = \Theta_n(n^{-b_v})$. The right factor satisfies $\|h(\mathbf{x}^\mu)\| = \Theta_n(1)$, so:

$$\|n^{-a_w - a_v} V^\top \nabla_f \ell^\mu h(\mathbf{x}^\mu)^\top\| = n^{-a_w - a_v} \|V^\top \nabla_f \ell^\mu\| \|h(\mathbf{x}^\mu)\| = \Theta_n(n^{-a_w - a_v - b_v}). \tag{27}$$

For g_u, the per-example term is the outer product $(W^\top V^\top \nabla_f \ell^\mu) \mathbf{x}^{\mu\top}$. The left factor is a matrix–vector product, whose expected squared RMS norm at initialization is:

$$\langle \|W^\top V^\top \nabla_f \ell^\mu\|^2 \rangle \sim n^{-2b_w + 1} \, \|V^\top \nabla_f \ell^\mu\|^2 = \Theta_n(n^{-2b_w - 2b_v + 1}),$$

Thus, $\|W^\top V^\top \nabla_f \ell^\mu\| = \Theta_n(n^{-b_w - b_v + \frac{1}{2}})$. Using $a_w + b_w = \frac{1}{2}$, this scaling becomes $\Theta_n(n^{a_w - b_v})$. The right factor satisfies $\|\mathbf{x}^\mu\| = \Theta_n(1)$, so:

$$\|n^{-a_u - a_w - a_v} W^\top V^\top \nabla_f \ell^\mu \mathbf{x}^{\mu\top}\| = n^{-a_u - a_w - a_v} \|W^\top V^\top \nabla_f \ell^\mu\| \|\mathbf{x}^\mu\| = \Theta_n(n^{-a_u - a_v - b_v}). \tag{28}$$

In summary, the per-example contributions scale as:

$$\|g_v^\mu\| = \Theta_n(n^{-a_v}), \qquad \|g_w^\mu\| = \Theta_n(n^{-a_w - a_v - b_v}), \qquad \|g_u^\mu\| = \Theta_n(n^{-a_u - a_v - b_v}). \tag{29}$$

The full gradient g_l is obtained by averaging the per-example contributions:

$$g_l = \frac{1}{D} \sum_{\mu=1}^{D} g_l^\mu. \tag{30}$$

We make the following assumption on this aggregation:

Assumption K.3 (Aggregation does not induce width-scaling)**.** The average $\frac{1}{D}\sum_{\mu=1}^{D} g_l^\mu$ inherits the width-scaling of the per-example terms.

Under Assumption K.3, the layerwise gradients scale as:

$$\|g_v\| = \Theta_n(n^{-a_v}), \qquad \|g_w\| = \Theta_n(n^{-a_w-a_v-b_v}), \qquad \|g_u\| = \Theta_n(n^{-a_u-a_v-b_v}). \tag{31}$$

The assumption rules out the gradient acquiring extra width scaling from aggregation. This is a benign assumption, as we do not expect a width dependence from aggregating. Without it, the triangle inequality provides only $\mathcal{O}_n$ upper bounds.

Similar to the activations, the gradient scales can also evolve in a width-dependent manner during training. We will discuss the implications in the later sections.

K.6 Alignment and Feature Learning Exponents

Following Everett et al. (2024), we define three alignment exponents $\{\rho, \omega, \sigma\} \in [0, 1]$ per layer that quantify the correlations between weight matrices, their updates, and activation vectors at the first gradient step.

Definition K.3 (Alignment exponents)**.** For each layer, we define:

$$\begin{aligned}
\rho_{v,1} &:= \lim_{n\to\infty} \log_n \frac{\|\Delta V_1\, z_0\|}{\|\Delta V_1\|\|z_0\|}, & \rho_{w,1} &:= \lim_{n\to\infty} \log_n \frac{\|\Delta W_1\, h_0\|}{\|\Delta W_1\|\|h_0\|},\\
\omega_{v,1} &:= \lim_{n\to\infty} \log_n \frac{\|V_0\, \Delta z_1\|}{\|V_0\|\|\Delta z_1\|}, & \omega_{w,1} &:= \lim_{n\to\infty} \log_n \frac{\|W_0\, \Delta h_1\|}{\|W_0\|\|\Delta h_1\|},\\
\sigma_{v,1} &:= \lim_{n\to\infty} \log_n \frac{\|\Delta V_1\, \Delta z_1\|}{\|\Delta V_1\|\|\Delta z_1\|}, & \sigma_{w,1} &:= \lim_{n\to\infty} \log_n \frac{\|\Delta W_1\, \Delta h_1\|}{\|\Delta W_1\|\|\Delta h_1\|}.
\end{aligned} \tag{32}$$

We do not define alignment exponents for the first layer because the matrix-vector product $U\mathbf{x}$ does not involve the width, and thus no alignment exponent is needed.

Definition K.4 (Activation Update exponents)**.** Let Δh_1, Δz_1, and Δf_1 denote the activation updates at the first gradient step. We define:

$$r_{u,1} := \lim_{n\to\infty} \log_n \|\Delta h_1\|, \quad r_{w,1} := \lim_{n\to\infty} \log_n \|\Delta z_1\|, \quad r_{v,1} := \lim_{n\to\infty} \log_n \|\Delta f_1\|. \tag{33}$$

These definitions generalize to any step t by replacing the subscript 1 with t throughout. We work at $t = 1$ in the following sections because the scaling of weights, updates, and activations is tractable at the first step.

K.7 Stable Update Conditions for SGD

We now derive the conditions on $\{a, b, c\}$ that satisfy Desideratum 2 for SGD, assuming Desideratum 1 holds. We analyze the weight decay scaling in Section K.9 for SGD and Adam together, as it follows the same logic for both optimizers.

SGD update. For SGD with per-layer learning rate $\eta_l = \eta\, n^{-c_l}$, the parameter update at step t is:

$$U_{t+1} = U_t - \eta_u\, g_u(\boldsymbol{\theta}_t), \qquad W_{t+1} = W_t - \eta_w\, g_w(\boldsymbol{\theta}_t), \qquad V_{t+1} = V_t - \eta_v\, g_v(\boldsymbol{\theta}_t). \tag{34}$$

At the first step, $\|\Delta U_1\| = \eta_u\|g_u\|$, $\|\Delta W_1\| = \eta_w\|g_w\|$, and $\|\Delta V_1\| = \eta_v\|g_v\|$. Using the gradient scalings at initialization from Section K.5:

$$\|\Delta U_1\| = \Theta_n(n^{-c_u-a_u-a_v-b_v}), \quad \|\Delta W_1\| = \Theta_n(n^{-c_w-a_w-a_v-b_v}), \quad \|\Delta V_1\| = \Theta_n(n^{-c_v-a_v}). \tag{35}$$

Desideratum 2 requires $\|\Delta h_1\|, \|\Delta z_1\|, \|\Delta f_1\| = \Theta_n(1)$, which we enforce by setting the activation update exponents to zero: $r_{u,1} = r_{w,1} = r_{v,1} = 0$.

First layer. The activation update is $\Delta h_1(\mathbf{x}) = n^{-a_u}\Delta U_1\,\mathbf{x}$. Since the matrix-vector product $U\mathbf{x}$ does not scale with width, we can write $\|\Delta U_1\mathbf{x}\| = C\|\Delta U_1\|\|\mathbf{x}\|$ for some width-independent constant C. As a result, the activation update scales as:

$$\|\Delta h_1(\mathbf{x})\| = Cn^{-a_u}\|\Delta U_1\|\|\mathbf{x}\| = \Theta_n(n^{-2a_u-c_u-a_v-b_v}). \tag{36}$$

Setting $r_{u,1} = 0$:

$$\boxed{2a_u + c_u + a_v + b_v = 0.} \tag{37}$$

Middle layer. The activation $z(\mathbf{x})$ at step $t = 1$ is:

$$z_1(\mathbf{x}) = n^{-a_w}(W_0 + \Delta W_1)(h_0(\mathbf{x}) + \Delta h_1(\mathbf{x})). \tag{38}$$

After expanding and subtracting $z_0(\mathbf{x}) = n^{-a_w}W_0h_0(\mathbf{x})$, the activation update decomposes into three terms:

$$\Delta z_1(\mathbf{x}) = \underbrace{n^{-a_w}\Delta W_1\,h_0(\mathbf{x})}_{\text{weight-update term}} + \underbrace{n^{-a_w}W_0\,\Delta h_1(\mathbf{x})}_{\text{activation-update term}} + \underbrace{n^{-a_w}\Delta W_1\,\Delta h_1(\mathbf{x})}_{\text{second-order term}}. \tag{39}$$

By the triangle inequality:

$$\|\Delta z_1(\mathbf{x})\| \;\le\; n^{-a_w}(\|\Delta W_1h_0(\mathbf{x})\| + \|W_0\Delta h_1(\mathbf{x})\| + \|\Delta W_1\Delta h_1(\mathbf{x})\|)\,. \tag{40}$$

We make the following assumption on the superposition of the three terms:

Assumption K.4 (Superposition does not induce width-scaling)**.** The superposition of the three terms in $\Delta z_1(\mathbf{x})$ does not introduce additional width-scaling: if each term is $\Theta_n(n^\alpha)$, then their sum is $\Theta_n(n^\alpha)$.

Under Assumption K.4, it suffices to analyze the width-scaling of each term individually. Throughout this derivation, we impose the condition $r_{u,1} = 0$, i.e., $\|\Delta h_1\| = \Theta_n(1)$.

Weight-update term. By the definition of the alignment exponent (Definition K.3), $\|\Delta W_1\,h_0\| = n^{\rho_{w,1}}\,\|\Delta W_1\|\,\|h_0\|$. Using $\|\Delta W_1\| = \Theta_n(n^{-c_w-a_w-a_v-b_v})$ from the SGD update and $\|h_0\| = \Theta_n(1)$:

$$n^{-a_w}\|\Delta W_1\,h_0(\mathbf{x})\| = \Theta_n(n^{\rho_{w,1}-2a_w-c_w-a_v-b_v}). \tag{41}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\rho_{w,1} - 2a_w - c_w - a_v - b_v = 0.} \tag{42}$$

Activation-update term. Similarly, $\|W_0\,\Delta h_1\| = n^{\omega_{w,1}}\,\|W_0\|\,\|\Delta h_1\|$. Using $\|W_0\| = \Theta_n(n^{-b_w})$ at initialization, $\|\Delta h_1\| = \Theta_n(1)$, and $a_w + b_w = 1/2$:

$$n^{-a_w}\|W_0\,\Delta h_1(\mathbf{x})\| = \Theta_n(n^{\omega_{w,1}-a_w-b_w}) = \Theta_n(n^{\omega_{w,1}-1/2}). \tag{43}$$

Requiring this to be $\Theta_n(1)$ fixes the alignment exponent itself:

$$\boxed{\omega_{w,1} - \tfrac{1}{2} = 0.} \tag{44}$$

Unlike $\rho_{w,1}$ (and $\sigma_{w,1}$, as we will see below), $\omega_{w,1}$ is not a free variable if we demand stability and non-vanishing updates $r_{w,1} = 0$ and its value is pinned to $1/2$. This scaling $\omega = 1/2$ is expected to hold at the first gradient step, but as training progresses, $\omega_{w,t}$ is free to evolve, and the activation-update term contribution may vanish or diverge as width is scaled.

Second-order term. By the definition of the alignment exponent (Definition K.3), $\|\Delta W_1\,\Delta h_1\| = n^{\sigma_{w,1}}\,\|\Delta W_1\|\,\|\Delta h_1\|$. Using $\|\Delta W_1\| = \Theta_n(n^{-c_w-a_w-a_v-b_v})$ and $\|\Delta h_1\| = \Theta_n(1)$:

$$n^{-a_w}\|\Delta W_1\,\Delta h_1(\mathbf{x})\| = \Theta_n(n^{\sigma_{w,1}-2a_w-c_w-a_v-b_v}). \tag{45}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\sigma_{w,1} - 2a_w - c_w - a_v - b_v = 0.} \tag{46}$$

The boxed conditions Equations (42), (44) and (46) correspond to requiring *all three* terms in Δz_1 to be $\Theta_n(1)$. The minimal requirement for $\|\Delta z_1\| = \Theta_n(1)$ is weaker: each term must be $\mathcal{O}_n(1)$ and at least one must be $\Theta_n(1)$, i.e.,

$$\boxed{\max\big(\rho_{w,1} - 2a_w - c_w - a_v - b_v,\;\; \omega_{w,1} - \tfrac{1}{2},\;\; \sigma_{w,1} - 2a_w - c_w - a_v - b_v\big) = 0.} \tag{47}$$

Last layer. The output update decomposes similarly:

$$\Delta f_1(\mathbf{x}) = \underbrace{n^{-a_v}\Delta V_1\, z_0(\mathbf{x})}_{\text{weight-update term}} + \underbrace{n^{-a_v} V_0\, \Delta z_1(\mathbf{x})}_{\text{activation-update term}} + \underbrace{n^{-a_v}\Delta V_1\, \Delta z_1(\mathbf{x})}_{\text{second-order term}}. \tag{48}$$

Under Assumption K.4, it suffices to analyze each term individually. We impose $r_{w,1} = 0$, i.e., $\|\Delta z_1\| = \Theta_n(1)$.

Weight-update term. By the definition of the alignment exponent (Definition K.3), $\|\Delta V_1\, z_0\| = n^{\rho_{v,1}}\, \|\Delta V_1\|\, \|z_0\|$. Using $\|\Delta V_1\| = \Theta_n(n^{-c_v - a_v})$ from the SGD update and $\|z_0\| = \Theta_n(1)$:

$$n^{-a_v}\|\Delta V_1\, z_0(\mathbf{x})\| = \Theta_n(n^{\rho_{v,1} - 2a_v - c_v}). \tag{49}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\rho_{v,1} - 2a_v - c_v = 0.} \tag{50}$$

Activation-update term. Similarly, $\|V_0\, \Delta z_1\| = n^{\omega_{v,1}}\, \|V_0\|\, \|\Delta z_1\|$. Using $\|V_0\| = \Theta_n(n^{-b_v})$ at initialization and $\|\Delta z_1\| = \Theta_n(1)$:

$$n^{-a_v}\|V_0\, \Delta z_1(\mathbf{x})\| = \Theta_n(n^{\omega_{v,1} - a_v - b_v}). \tag{51}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\omega_{v,1} - a_v - b_v = 0.} \tag{52}$$

Unlike the middle layer, demanding $r_{v,1} = 0$ does not pin $\omega_{v,1}$ to a fixed value, and it is constrained only through the relation $\omega_{v,1} = a_v + b_v$, where $a_v + b_v$ is itself underdetermined due to the weaker initialization constraint (Desideratum 1).

Second-order term. By the definition of the alignment exponent (Definition K.3), $\|\Delta V_1\, \Delta z_1\| = n^{\sigma_{v,1}}\, \|\Delta V_1\|\, \|\Delta z_1\|$. Using the same scalings:

$$n^{-a_v}\|\Delta V_1\, \Delta z_1(\mathbf{x})\| = \Theta_n(n^{\sigma_{v,1} - 2a_v - c_v}). \tag{53}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\sigma_{v,1} - 2a_v - c_v = 0.} \tag{54}$$

As in the middle layer, the boxed conditions Equations (50), (52) and (54) correspond to requiring all three terms in Δf_1 to be $\Theta_n(1)$. The minimal requirement for $\|\Delta f_1\| = \Theta_n(1)$ is weaker:

$$\boxed{\max\big(\rho_{v,1} - 2a_v - c_v,\ \ \omega_{v,1} - a_v - b_v,\ \ \sigma_{v,1} - 2a_v - c_v\big) = 0.} \tag{55}$$

Gauge symmetry for SGD. The boxed conditions Equations (47) and (55) are invariant under the per-layer transformation:

$$a_l \to a_l + \Delta_l, \qquad b_l \to b_l - \Delta_l, \qquad c_l \to c_l - 2\Delta_l,$$

for any $\Delta_l \in \mathbb{R}$ and each layer $l \in \{u, w, v\}$. The forward-pass exponent $a_l + b_l$ is unchanged, and the shift in c_l compensates so that update equations have the same width scalings. As a consequence, different choices of (a, b, c) describe the same training dynamics. We will later use this freedom later to write down several equivalent forms of μP (Table 5).

K.8 Stable Update Conditions for Adam

We now derive the conditions on $\{a, b, c\}$ that satisfy Desideratum 2 for Adam, assuming Desideratum 1 holds. The only change from SGD is in the scaling of the parameter updates; the activation decompositions and alignment-exponent calculations are otherwise identical. Throughout this section, we use Assumption K.4 on the superposition of three-term decomposition, and apply the alignment-exponent identities from Definition K.3 (e.g., $\|\Delta W h\| = n^{\rho_w}\|\Delta W\|\|h\|$) without further citation. Weight decay is analyzed in Section K.9 for SGD and Adam together.

Adam update. Adam computes the update as $\Delta\theta_l = -\eta_l \cdot m_l/(\sqrt{v_l} + \epsilon)$, where m_l and v_l are the first and second moment estimates of g_l. We assume:

Assumption K.5 (Adam normalized update)**.** The normalized update $m_l/(\sqrt{v_l} + \epsilon)$ has $\Theta_n(1)$ entries. This is a reasonable assumption as m_l scales as g_l and v_l scales as g_l^2 entrywise, so $m_l/\sqrt{v_l}$ behaves like a sign-magnitude normalization independent of the gradient width scaling.

Under Assumption K.5, the parameter update norms at the first step depend only on the layerwise learning rate:

$$\|\Delta U_1\| = \Theta_n(n^{-c_u}), \qquad \|\Delta W_1\| = \Theta_n(n^{-c_w}), \qquad \|\Delta V_1\| = \Theta_n(n^{-c_v}). \tag{56}$$

Desideratum 2 requires $\|\Delta h_1\|, \|\Delta z_1\|, \|\Delta f_1\| = \Theta_n(1)$, which we enforce by setting the activation update exponents to zero: $r_{u,1} = r_{w,1} = r_{v,1} = 0$.

First layer. The activation update is $\Delta h_1(\mathbf{x}) = n^{-a_u}\Delta U_1\,\mathbf{x}$. Since the matrix-vector product $U\mathbf{x}$ does not scale with width, we can write $\|\Delta U_1\mathbf{x}\| = C\|\Delta U_1\|\|\mathbf{x}\|$ for some width-independent constant C. As a result:

$$\|\Delta h_1(\mathbf{x})\| = Cn^{-a_u}\|\Delta U_1\|\|\mathbf{x}\| = \Theta_n(n^{-a_u-c_u}). \tag{57}$$

Setting $r_{u,1} = 0$:

$$\boxed{a_u + c_u = 0.} \tag{58}$$

Middle layer. As in the SGD case, the activation update decomposes into three terms:

$$\Delta z_1(\mathbf{x}) = \underbrace{n^{-a_w}\Delta W_1\, h_0(\mathbf{x})}_{\text{weight-update term}} + \underbrace{n^{-a_w} W_0\, \Delta h_1(\mathbf{x})}_{\text{activation-update term}} + \underbrace{n^{-a_w}\Delta W_1\, \Delta h_1(\mathbf{x})}_{\text{second-order term}}. \tag{59}$$

Weight-update term. Using $\|\Delta W_1\| = \Theta_n(n^{-c_w})$ and $\|h_0\| = \Theta_n(1)$:

$$n^{-a_w}\|\Delta W_1\, h_0(\mathbf{x})\| = n^{-a_w}\, n^{\rho_{w,1}}\, \|\Delta W_1\|\, \|h_0\| = \Theta_n(n^{\rho_{w,1}-a_w-c_w}). \tag{60}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\rho_{w,1} - a_w - c_w = 0.} \tag{61}$$

Activation-update term. Using $\|W_0\| = \Theta_n(n^{-b_w})$ at initialization, $\|\Delta h_1\| = \Theta_n(1)$, and $a_w + b_w = 1/2$:

$$n^{-a_w}\|W_0\, \Delta h_1(\mathbf{x})\| = n^{-a_w}\, n^{\omega_{w,1}}\, \|W_0\|\, \|\Delta h_1\| = \Theta_n(n^{\omega_{w,1}-1/2}). \tag{62}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\omega_{w,1} - \tfrac{1}{2} = 0.} \tag{63}$$

As in the SGD case, $\omega_{w,1}$ is not a free variable if we demand stability and non-vanishing updates $r_{w,1} = 0$, and its value gets pinned to $1/2$.

Second-order term. Using $\|\Delta W_1\| = \Theta_n(n^{-c_w})$ and $\|\Delta h_1\| = \Theta_n(1)$:

$$n^{-a_w}\|\Delta W_1\, \Delta h_1(\mathbf{x})\| = n^{-a_w}\, n^{\sigma_{w,1}}\, \|\Delta W_1\|\, \|\Delta h_1\| = \Theta_n(n^{\sigma_{w,1}-a_w-c_w}). \tag{64}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\sigma_{w,1} - a_w - c_w = 0.} \tag{65}$$

The boxed conditions Equations (61), (63) and (65) correspond to requiring all three terms to be $\Theta_n(1)$. The minimal requirement for $\|\Delta z_1\| = \Theta_n(1)$ is weaker:

$$\boxed{\max\big(\rho_{w,1} - a_w - c_w,\ \ \omega_{w,1} - \tfrac{1}{2},\ \ \sigma_{w,1} - a_w - c_w\big) = 0.} \tag{66}$$

Last layer. The output update decomposes similarly:

$$\Delta f_1(\mathbf{x}) = \underbrace{n^{-a_v}\Delta V_1\, z_0(\mathbf{x})}_{\text{weight-update term}} + \underbrace{n^{-a_v} V_0\, \Delta z_1(\mathbf{x})}_{\text{activation-update term}} + \underbrace{n^{-a_v}\Delta V_1\, \Delta z_1(\mathbf{x})}_{\text{second-order term}}. \tag{67}$$

Weight-update term. Using $\|\Delta V_1\| = \Theta_n(n^{-c_v})$ and $\|z_0\| = \Theta_n(1)$:

$$n^{-a_v}\|\Delta V_1\, z_0(\mathbf{x})\| = n^{-a_v}\, n^{\rho_{v,1}}\, \|\Delta V_1\|\, \|z_0\| = \Theta_n(n^{\rho_{v,1}-a_v-c_v}). \tag{68}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\rho_{v,1} - a_v - c_v = 0.} \tag{69}$$

Activation-update term. Using $\|V_0\| = \Theta_n(n^{-b_v})$ at initialization and $\|\Delta z_1\| = \Theta_n(1)$:

$$n^{-a_v}\|V_0\, \Delta z_1(\mathbf{x})\| = n^{-a_v}\, n^{\omega_{v,1}}\, \|V_0\|\, \|\Delta z_1\| = \Theta_n(n^{\omega_{v,1}-a_v-b_v}). \tag{70}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\omega_{v,1} - a_v - b_v = 0.} \tag{71}$$

As in the SGD case, demanding $r_{v,1} = 0$ does not pin $\omega_{v,1}$ to a fixed value; it is constrained only through the relation $\omega_{v,1} = a_v + b_v$, where $a_v + b_v$ is itself underdetermined due to the weaker initialization constraint (Desideratum 1).

Second-order term. Using $\|\Delta V_1\| = \Theta_n(n^{-c_v})$ and $\|\Delta z_1\| = \Theta_n(1)$:

$$n^{-a_v}\|\Delta V_1\, \Delta z_1(\mathbf{x})\| = n^{-a_v}\, n^{\sigma_{v,1}}\, \|\Delta V_1\|\, \|\Delta z_1\| = \Theta_n(n^{\sigma_{v,1}-a_v-c_v}). \tag{72}$$

Requiring this to be $\Theta_n(1)$:

$$\boxed{\sigma_{v,1} - a_v - c_v = 0.} \tag{73}$$

As in the middle layer, the boxed conditions Equations (69), (71) and (73) correspond to requiring all three terms in Δf_1 to be $\Theta_n(1)$. The minimal requirement for $\|\Delta f_1\| = \Theta_n(1)$ is weaker:

$$\boxed{\max\big(\rho_{v,1} - a_v - c_v,\ \ \omega_{v,1} - a_v - b_v,\ \ \sigma_{v,1} - a_v - c_v\big) = 0.} \tag{74}$$

Gauge symmetry for Adam. The boxed conditions Equations (66) and (74) are invariant under the per-layer transformation:

$$a_l \to a_l + \Delta_l, \qquad b_l \to b_l - \Delta_l, \qquad c_l \to c_l - \Delta_l,$$

for any $\Delta_l \in \mathbb{R}$ and each layer $l \in \{u, w, v\}$. The shift in c_l differs from the SGD case ($-2\Delta_l$ there) because Adam's update norm $\Theta_n(n^{-c_l})$ depends on c_l only once, whereas SGD's $\eta_l\|g_l\|$ inherits c_l explicitly and additional a, b dependence implicitly through the gradient.

K.9 Weight Decay Scaling

With weight decay, the parameter update at the first step becomes:

$$\Delta\theta_l = -\eta_l\, \phi(g_l) - \eta_l \lambda_l\, \theta_l, \tag{75}$$

where $\phi(g_l)$ is the optimizer's transformation of the gradient: $\phi(g_l) = g_l$ for SGD and $\phi(g_l) = m_l/(\sqrt{v_l}+\epsilon)$ for Adam. For the weight decay term to contribute at the same scale as the parameters, we require $\eta_l\lambda_l = \Theta_n(1)$. Using $\eta_l = \eta\, n^{-c_l}$ and $\lambda_l = \lambda\, n^{-d_l}$:

$$\boxed{c_l + d_l = 0.} \tag{76}$$

This holds identically for SGD and Adam, since the weight decay term does not depend on ϕ.

Table 3: Summary of stability conditions for SGD and Adam. Each $r_{l,1} = 0$ condition is a max over the three terms (weight-update, activation-update, second-order) in the activation decomposition.

	SGD	Adam
Stable Init	$a_u + b_u = 0$ $a_w + b_w = 1/2$ $a_v + b_v \geq 1/2$	
Stable Updates	$r_{u,1} = 2a_u + c_u + a_v + b_v = 0$ $r_{w,1} = \max\begin{Bmatrix} \rho_{w,1} - 2a_w - c_w - a_v - b_v \\ \omega_{w,1} - \frac{1}{2} \\ \sigma_{w,1} - 2a_w - c_w - a_v - b_v \end{Bmatrix} = 0$ $r_{v,1} = \max\begin{Bmatrix} \rho_{v,1} - 2a_v - c_v \\ \omega_{v,1} - a_v - b_v \\ \sigma_{v,1} - 2a_v - c_v \end{Bmatrix} = 0$	$r_{u,1} = a_u + c_u = 0$ $r_{w,1} = \max\begin{Bmatrix} \rho_{w,1} - a_w - c_w \\ \omega_{w,1} - \frac{1}{2} \\ \sigma_{w,1} - a_w - c_w \end{Bmatrix} = 0$ $r_{v,1} = \max\begin{Bmatrix} \rho_{v,1} - a_v - c_v \\ \omega_{v,1} - a_v - b_v \\ \sigma_{v,1} - a_v - c_v \end{Bmatrix} = 0$
Weight Decay	$c_l + d_l = 0$ for all layers l	

Extended gauge symmetry. Including weight decay extends the abc gauge symmetry to $abcd$. The condition $c_l + d_l = 0$ is preserved under

$$d_l \to d_l + \zeta \Delta_l, \tag{77}$$

where $\zeta = 2$ for SGD and $\zeta = 1$ for Adam.

K.10 Summary of Stability Conditions

Table 3 summarizes the conditions on the $\{a, b, c, d\}$ exponents that satisfy Desiderata 1 and 2. The activation update conditions $r_{l,1} = 0$ at each layer are expressed as a max over the three decomposition terms (weight-update, activation-update, second-order), reflecting the minimal requirement for $\|\Delta h_1\|, \|\Delta z_1\|, \|\Delta f_1\| = \Theta_n(1)$. In the next section, we discuss the conditions on the alignment exponents.

K.11 Alignment Exponent Values

The stability conditions constrain the alignment exponents, however, the alignment exponents are not free design choices, and are empirical properties that quantify the correlations between weights, activations and their updates.

At initialization, the weights and activations are all random and their products behave like products of independent Gaussian vectors (same logic as in Desideratum 1). As a result, all alignment exponents have the value $1/2$:

$$\rho_{l,1} = \omega_{l,1} = \sigma_{l,1} = \tfrac{1}{2} \qquad \text{at initialization.} \tag{78}$$

As training progresses, the weights, activations, and their updates become correlated. As a result, the alignment exponents increase above their random-alignment value of $1/2$ during training, although they typically remain far below the fully aligned value of 1 (Everett et al., 2024).

To derive μP, Yang & Hu (2021) first impose $r_{u,1} = r_{w,1} = r_{v,1} = 0$, so that all activation updates are $\Theta_n(1)$. This constrains the hidden layer exponent $\omega_w = 1/2$. Since weights, activations, and updates become correlated during training, Yang & Hu (2021) assume that the remaining free exponents attain their maximum value:

Assumption K.6 (μP Full Alignment Assumption)**.** After imposing $r_{u,1} = r_{w,1} = r_{v,1} = 0$, all remaining free alignment exponents are set to 1:

$$\rho_{w,1} = \rho_{v,1} = \sigma_{w,1} = \sigma_{v,1} = \omega_{v,1} = 1. \tag{79}$$

Table 4 shows the conditions on $abcd$ after imposing the above assumption. Note that the assumption $\omega_v = 1$ tightens the condition from $a_v + b_v \geq 1/2$ to $a_v + b_v = 1$. Everett et al. (2024) relax this assumption to $\omega_{v,1} = 1/2$, and observe that hyperparameter transfer still holds empirically, which suggests that μP's full alignment assumption may be excessive.

Table 4: Conditions on exponents (a, b, c, d) for SGD and Adam obtained after imposing Assumption K.6.

Layer	**SGD**	**Adam**
First	$a_u + b_u = 0$	$a_u + b_u = 0$
	$2a_u + c_u + a_v + b_v = 0$	$a_u + c_u = 0$
	$c_u + d_u = 0$	$c_u + d_u = 0$
Middle	$a_w + b_w = 1/2$	$a_w + b_w = 1/2$
	$2a_w + c_w + a_v + b_v = 1$	$a_w + c_w = 1$
	$c_w + d_w = 0$	$c_w + d_w = 0$
Last	$a_v + b_v = 1$	$a_v + b_v = 1$
	$2a_v + c_v = 1$	$a_v + c_v = 1$
	$c_v + d_v = 0$	$c_v + d_v = 0$

K.12 Gauge Symmetry and Parameterization Choice

Even with these constraints, the system remains underdetermined due to the gauge symmetry of the $abcd$ parameterization. One exponent per layer must be fixed to obtain a specific parameterization. These choices are typically made based on implementation convenience. For instance, setting $a = 0$ removes the explicit width factors in the forward pass, while setting all learning rate exponents $c = 0$ allows using a single global learning rate. The most common choice is the canonical μP proposed in Yang & Hu (2021). Table 5 summarizes several such implementations, all of which are gauge-equivalent and yield identical training dynamics under SGD and Adam. This symmetry, however, can be broken by operations that modify the gradients, such as gradient clipping or Adam's ϵ.

K.13 Multi-Step Scaling

So far we have analyzed the scaling of activation updates at the first training step. At later steps, the weights, activations, gradients, and alignment exponents can evolve in a width-dependent way. To extend the first-step scaling calculation to multiple steps, we assume that the weight and activation scalings persist throughout training.

Assumption K.7 (Persistent weight and activation scaling)**.** For any training step t, we assume:

$$\|U_t\| = \Theta_n(n^{-b_u}), \qquad \|W_t\| = \Theta_n(n^{-b_w}), \qquad \|V_t\| = \Theta_n(n^{-b_v}), \tag{80}$$

and

$$\|h_t\| = \Theta_n(1), \qquad \|z_t\| = \Theta_n(1), \qquad \|f_t\| = \mathcal{O}_n(1). \tag{81}$$

For SGD, we additionally assume that the gradient scalings remain the same as at initialization.

Assumption K.8 (Persistent gradient scaling for SGD)**.** For all training steps $t \leq T$,

$$\|g_{v,t}\| = \Theta_n(n^{-a_v}), \qquad \|g_{w,t}\| = \Theta_n(n^{-a_w - a_v - b_v}), \qquad \|g_{u,t}\| = \Theta_n(n^{-a_u - a_v - b_v}). \tag{82}$$

Adam does not require this gradient-scaling assumption, because under Assumption K.5 the normalized update has $\Theta_n(1)$ entries.

Under these assumptions, the first-step scaling analysis can be repeated at any fixed step t. However, this still does not control accumulation over a number of steps that grows with width. We therefore also assume a width-independent training horizon.

Assumption K.9 (Width-independent training horizon)**.** The number of training steps does not scale with width:

$$T = \Theta_n(1). \tag{83}$$

This assumption is quite restrictive and does not hold in realistic training regimes where the number of optimization steps is much larger than the width, or where the number of steps itself scales with width. This occurs, for example, in compute-optimal training. For a width-n model with $\Theta(n^2)$ parameters, the token budget scales with the parameter count, and as a result the number of optimization steps scales as $T(n) = \Theta_n(n^2)$. In such regimes, different-width models are trained for different numbers of steps, and therefore accumulate different total update magnitudes. Therefore, fixed-step scaling arguments do not directly guarantee that training would remain consistent across widths.

Table 5: Comparison of different μP implementations for SGD and Adam. All implementations are gauge-equivalent and yield identical training dynamics. Weight decay exponents satisfy $c_l + d_l = 0$.

Implementation	Layer	SGD				Adam			
		Mult. (n^{-a})	Var. (n^{-2b})	LR (n^{-c})	WD (n^{-d})	Mult. (n^{-a})	Var. (n^{-2b})	LR (n^{-c})	WD (n^{-d})
No multipliers (Hayou & Liu, 2025)	First	1	1	n	$1/n$	1	1	1	1
	Middle	1	$1/n$	1	1	1	$1/n$	$1/n$	n
	Last	1	$1/n^2$	$1/n$	n	1	$1/n^2$	$1/n$	n
No LR scaling	First	$\sqrt{n}$	$1/n$	1	1	1	1	1	1
	Middle	1	$1/n$	1	1	$1/n$	n	1	1
	Last	$1/\sqrt{n}$	$1/n$	1	1	$1/n$	1	1	1
Canonical (Yang & Hu, 2021)	First	$\sqrt{n}$	$1/n$	1	1	$\sqrt{n}$	$1/n$	$1/\sqrt{n}$	$\sqrt{n}$
	Middle	1	$1/n$	1	1	1	$1/n$	$1/n$	n
	Last	$1/\sqrt{n}$	$1/n$	1	1	$1/\sqrt{n}$	$1/n$	$1/\sqrt{n}$	$\sqrt{n}$
CompleteP (Dey et al., 2025)	First	1	1	n	$1/n$	1	1	1	1
	Middle	1	$1/n$	1	1	1	$1/n$	$1/n$	n
	Last	$1/n$	1	n	$1/n$	$1/n$	1	1	1

K.14 Attention and LayerNorm scaling

In standard transformers, two additional layer types are commonly used beyond the linear layers analyzed so far: attention and LayerNorm. These two have been analyzed in Yang & Hu (2021) and Dey et al. (2025). We treat them in turn, starting with attention.

Attention Scaling. For queries, keys, and values $Q, K, V \in \mathbb{R}^{T\times d}$, where T is the context length and d is the head dimension, the attention mechanism computes:

$$A = \mathrm{softmax}\left(\frac{QK^\top}{d^\alpha}\right)V, \tag{84}$$

where d^α is the attention scaling factor. If the logits are too large, the softmax concentrates all mass on a single entry. To avoid it, we require the logits norm to be $\|QK^\top/d^\alpha\| = \Theta_d(1)$. Assuming $\|Q\|, \|K\| = \Theta_d(1)$, the dot product norm scales according to the alignment exponent:

$$\rho_a := \lim_{d\to\infty} \log_d \frac{\|QK^\top\|}{\|Q\|\|K\|}, \tag{85}$$

giving $\|QK^\top\| = \Theta_d(d^{\rho_a})$. At initialization, W_Q and W_K are independent Gaussian matrices, so Q and K behave as independent random matrices and $\|QK^\top\| = \Theta_d(\sqrt{d})$, giving $\alpha = 1/2$, which is the scaling used in the original Transformer paper (Vaswani et al., 2017a). During training, Q and K should develop correlations between them. Under the full alignment assumption (analogous to Assumption K.6), $\|QK^\top\| = \Theta_d(d)$, giving $\alpha = 1$, which is the μP prescription (Yang & Hu, 2021).

The alignment exponent ρ_a itself can depend on width n and evolve during training, so neither prescription is *correct* in general. That said, $\alpha = 1$ acts as an upper bound, since $\rho_a \leq 1$.

LayerNorm. For $\mathbf{x} \in \mathbb{R}^n$ with trainable gain $\boldsymbol{\gamma} \in \mathbb{R}^n$ and bias $\beta \in \mathbb{R}^n$, the output of LayerNorm is given by:

$$\mathrm{LN}(\mathbf{x}) = \hat{\mathbf{x}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}, \qquad \hat{\mathbf{x}} := \frac{\mathbf{x} - \mu_x \mathbf{1}}{\sqrt{\sigma_x^2 + \epsilon}}, \tag{86}$$

where $\|\hat{\mathbf{x}}\| = \Theta_n(1)$.

The LayerNorm parameters are deterministically initialized as $\gamma_i = 1, \beta_i = 0$, which gives $\|\mathrm{LN}(\mathbf{x})\|_0 = \Theta_n(1)$ at initialization. There is no forward-pass multiplier a_{LN}, and weight decay is typically not applied to LayerNorm parameters, so only the learning rate exponent c_{LN} remains to be determined.

The gradients with respect to $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are:

$$(g_{\boldsymbol{\gamma}})_i = \tfrac{1}{D} \sum_\mu (\nabla_{\mathrm{LN}} \ell^\mu)_i \, \hat{x}_i^\mu, \qquad (g_{\boldsymbol{\beta}})_i = \tfrac{1}{D} \sum_\mu (\nabla_{\mathrm{LN}} \ell^\mu)_i, \tag{87}$$

where $\nabla_{\mathrm{LN}} \ell^\mu := \nabla_{\mathrm{LN}(\mathbf{x}^\mu)} \ell(f(\mathbf{x}^\mu), \mathbf{y}^\mu)$ denotes the gradient of the per-example loss with respect to the LayerNorm output. Assuming $\|\nabla_{\mathrm{LN}} \ell^\mu\|, \|\hat{\mathbf{x}}^\mu\| = \Theta_n(1)$ entrywise, we get $\|g_{\boldsymbol{\gamma}}\|, \|g_{\boldsymbol{\beta}}\| = \Theta_n(1)$.

Since the gradients are $\Theta_n(1)$, for both SGD and Adam, the parameter updates at the first step satisfy $\|\Delta\boldsymbol{\gamma}_1\| = \|\Delta\boldsymbol{\beta}_1\| = \Theta_n(n^{-c_{\mathrm{LN}}})$. The resulting output update $\Delta\mathrm{LN}_1(\mathbf{x}) = \hat{\mathbf{x}} \odot \Delta\boldsymbol{\gamma}_1 + \Delta\boldsymbol{\beta}_1$ inherits this scaling. Requiring $\|\Delta\mathrm{LN}_1(\mathbf{x})\| = \Theta_n(1)$ yields $c_{\mathrm{LN}} = 0$.

K.15 Weight Tying

In language models, the input embedding and output unembedding matrices are often tied, sharing parameters to reduce the parameter count. In our three-layer setup, this corresponds to setting $V = U^\top$. The forward pass in this case becomes:

$$h(\mathbf{x}) = n^{-a_u} U \mathbf{x}, \qquad z(\mathbf{x}) = n^{-a_w} W h(\mathbf{x}), \qquad f(\mathbf{x}) = n^{-a_v} U^\top z(\mathbf{x}), \tag{88}$$

with $U \in \mathbb{R}^{n \times d_{\mathrm{in}}}$ and $W \in \mathbb{R}^{n \times n}$. The tied parameter U appears with two different forward-pass exponents a_u and a_v. The initialization variance is controlled by the exponents b_u, b_w:

$$U_{ij} \sim \mathcal{N}(0, n^{-2b_u}), \qquad W_{ij} \sim \mathcal{N}(0, n^{-2b_w}). \tag{89}$$

The per-layer learning rates are scaled by c_u and c_w:

$$\eta_u = \eta \, n^{-c_u}, \qquad \eta_w = \eta \, n^{-c_w}. \tag{90}$$

In contrast to the untied case, the last layer no longer has its own b_v and c_v exponents, since these are inherited from U as b_u and c_u. Only the forward-pass exponent a_v remains as a free parameter for the last layer. Performing a similar analysis to the untied case gives the stability conditions summarized in Table 6.

Next, applying Assumption K.6 yields Table 7, which compares the untied and tied exponents side by side for both SGD and Adam. For Adam, the first-layer and middle-layer conditions are identical in the tied and untied cases, since Adam's normalized update is independent of the gradient magnitude.

Table 6: Stability conditions under weight tying ($U = V^\top$).

	SGD	Adam
Stable Init	$a_u + b_u = 0$ $a_w + b_w = 1/2$ $a_v + b_u \geq 1/2$	
Stable Updates	$r_{u,1} = a_u + c_u + a_v = 0$ $r_{w,1} = \max\begin{Bmatrix} \rho_{w,1} - 2a_w - c_w - a_v - b_u \\ \omega_{w,1} - \frac{1}{2} \\ \sigma_{w,1} - 2a_w - c_w - a_v - b_u \end{Bmatrix} = 0$ $r_{v,1} = \max\begin{Bmatrix} \rho_{v,1} - 2a_v - c_u \\ \omega_{v,1} - a_v - b_u \\ \sigma_{v,1} - 2a_v - c_u \end{Bmatrix} = 0$	$r_{u,1} = a_u + c_u = 0$ $r_{w,1} = \max\begin{Bmatrix} \rho_{w,1} - a_w - c_w \\ \omega_{w,1} - \frac{1}{2} \\ \sigma_{w,1} - a_w - c_w \end{Bmatrix} = 0$ $r_{v,1} = \max\begin{Bmatrix} \rho_{v,1} - a_v - c_u \\ \omega_{v,1} - a_v - b_u \\ \sigma_{v,1} - a_v - c_u \end{Bmatrix} = 0$

Table 7: Conditions on exponents (a, b, c) under μP, with and without weight tying.

Layer	SGD (untied)	SGD (tied)	Adam (untied)	Adam (tied)
First	$a_u + b_u = 0$ $2a_u + c_u + a_v + b_v = 0$	$a_u + b_u = 0$ $a_u + c_u + a_v = 0$	$a_u + b_u = 0$ $a_u + c_u = 0$	$a_u + b_u = 0$ $a_u + c_u = 0$
Middle	$a_w + b_w = 1/2$ $2a_w + c_w + a_v + b_v = 1$	$a_w + b_w = 1/2$ $2a_w + c_w + a_v + b_u = 1$	$a_w + b_w = 1/2$ $a_w + c_w = 1$	$a_w + b_w = 1/2$ $a_w + c_w = 1$
Last	$a_v + b_v = 1$ $2a_v + c_v = 1$	$a_v + b_u = 1$ $2a_v + c_u = 1$	$a_v + b_v = 1$ $a_v + c_v = 1$	$a_v + b_u = 1$ $a_v + c_u = 1$

Why a_u and a_v must remain separate. Collapsing them to a single exponent ($a_v = a_u$) would force $a_u + b_u = 0$ from the first-layer init and $a_u + b_u = 1$ from the last-layer condition, resulting in a contradiction. The two forward-pass exponents thus provide the only remaining degree of freedom to satisfy both layer constraints simultaneously.

A weight-tied μP example. Unlike the untied case, we cannot set $a_u = a_v = 0$ simultaneously: substituting into the conditions yields $b_u = 0$ from the first layer and $b_u = 1$ from the last layer. We must therefore fix one of a_u, a_v to zero and let the other absorb the layer mismatch. Choosing $a_u = 0$ for Adam yields the table in Table 8. In this case, a naive SP implementation with $1/\text{fan-in}$ initialization and a global learning rate of $1/n$ would not only train the embedding layer slowly, but also have $\Theta(\sqrt{n})$ logits at initialization.

Table 8: A weight-tied μP example.

Layer	Mult. (n^{-a})	Var. (n^{-2b})	LR (n^{-c})	WD (n^{-d})
First (U)	1	1	1	1
Middle (W)	1	$1/n$	$1/n$	n
Last ($U^\top$)	$1/n$	1	1	1