Quantifying Hyperparameter Transfer and the Importance of Embedding Layer Learning Rate

Dayal Singh Kalra [2]

Maissam Barkeshli [1, 3,4]

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 [2, 5, 17, 18, 20, 23]. 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 [3, 5, 22]. 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 [28], keeping the training dynamics invariant across scales [4, 27]. Yang & Hu [28] 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*

1. Department of Physics, University of Maryland, College Park
2. Department of Computer Science, University of Maryland, College Park
3. Joint Quantum Institute, University of Maryland, College Park
4. Meta Superintelligence Labs, Fundamental AI Research

© D.S. Kalra & M. Barkeshli.

transfer, allows practitioners to utilize optimal learning rates from a smaller proxy model to train larger ones [29], reducing the need for expensive hyperparameter searches at scale.

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 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 make 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 [28], yet this alignment is low early in training and never reaches the full alignment even late in training [6]. It 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 J. 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) [9]. Surprisingly, despite these assumptions not holding, μP typically exhibits high-quality learning rate transfer—though it can break down in specific regimes, such as across the under-to-over-parameterization transition [12]. 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 (Section 2). We develop three key metrics. The first is the quality of the scaling law fit itself. The second is a robustness metric: whether the choice of optimal hyperparameter 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.

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 [26] trained with AdamW [19], μP has four key changes compared to SP (Section B). By examining all 16 ablations, we isolate the embedding layer learning rate as the key factor (Section 3). 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 (Section F). Therefore, one interesting source of training instabilities is that the learning can be bottlenecked by the embedding layer learning rate. We also study the effect of weight decay to show that (1) it can improve the scaling law fit quality, (2) in the fixed token per parameter setting, it hurts the robustness of the extrapolation (Section G).

2. Quantifying Hyperparameter Transfer

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

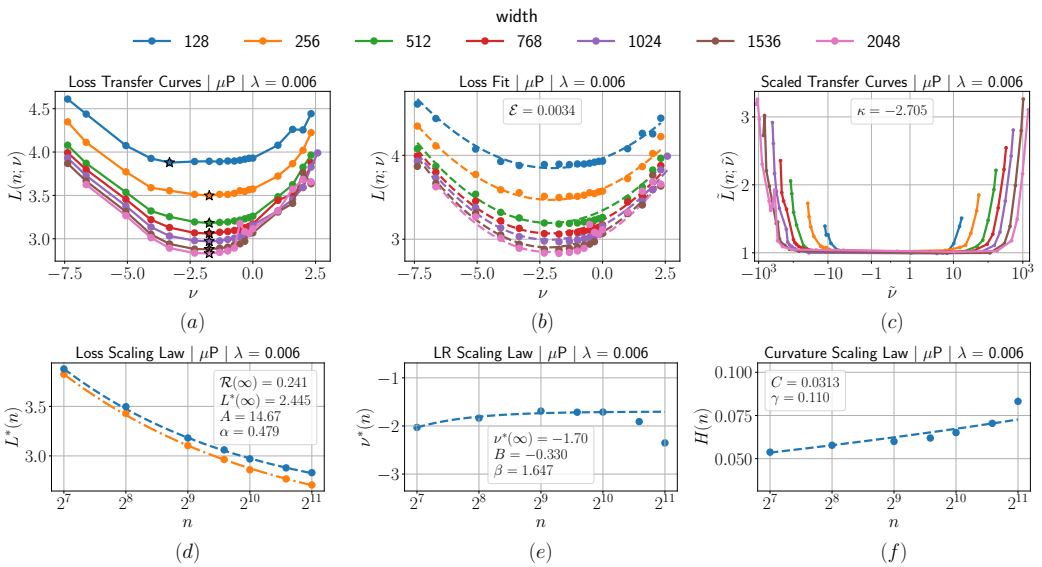


Figure 1: *Computing the three transfer metrics for μP.* (a) Loss vs. log learning rate ν, with star marking the optimum $\nu^*(n)$, (b) Joint fit of the loss model (Equation (4), dashed lines), with a low predictability error $\mathcal{E} = 0.0034$, (c) Loss curves in the normalized coordinates (Equation (6)), with $\kappa = -2.640$ indicating robust transfer. (d-f) Scaling laws for optimal loss $L^*(n)$, optimal log-learning-rate $\nu^*(n)$, and curvature $H(n)$.

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 [1, 9, 13], 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. We model the optimal log-learning rate [1] using a scaling law with an irreducible term as well:

$$\nu^*(n) = \nu^*(\infty) + Bn^{-\beta}, \tag{2}$$

where $0 < \nu^*(\infty) < \infty$ is its asymptotic value, B is the scaling coefficient and $\beta > 0$ controls the rate of convergence. 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 1 (Loss Degradation $\mathcal{R}(\infty)$) *For a parameterization, let $L^*(\infty)$ denote its irreducible loss, and let $L^*_{best}(\infty)$ be the best possible irreducible loss across hyperparameters and parameter-*

1. We consider $\log \eta$ rather than η itself because the loss curve is approximately symmetric around its minimum on a log scale, yielding better fits for quantifying reliability. See Section C for details.

izations. We define loss degradation as the asymptotic loss gap: $\mathcal{R}(\infty) = L^*(\infty) - L^*_{best}(\infty) \geq 0$. $\mathcal{R}(\infty) = 0$ *indicates best-in-class loss at scale, and* $\mathcal{R}(\infty) > 0$ *signals a performance gap.*

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

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

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[2] $H(n) = Cn^\gamma$, where C is the scaling coefficient and γ is the scaling exponent. Substituting the scaling laws into Equation (3), we get:

$$L(\nu; n) = L^*(\infty) + An^{-\alpha} + \frac{1}{2}Cn^\gamma \cdot \left(\nu - \nu^*(\infty) - Bn^{-\beta}\right)^2. \tag{4}$$

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

Definition 2 (Loss Predictability Error $\mathcal{E}$) *Given loss observations* $\{L(\nu_i; n_j)\}$ *at* N_ν *log-learning-rates* $\{\nu_i\}_{i=1}^{N_\nu}$ *and* N_n *widths* $\{n_j\}_{j=1}^{N_n}$*, we define loss predictability as the normalized mean squared error between observed and predicted loss* $\{\hat{L}(\nu_i; n_j)\}$ *from Equation (4) with fitted parameters:*

$$\mathcal{E} = \frac{1}{N_\nu N_n} \sum_{i,j}^{N_\nu, N_n} \left[L(\nu_i; n_j) - \hat{L}(\nu_i; n_j)\right]^2. \tag{5}$$

When $\mathcal{E} \approx 0$*, the loss is well captured by Equation (4), suggesting reliable extrapolation is possible. High* $\mathcal{E}$ *in contrast suggests a complex landscape arising from various sources, such as training instabilities, finite-size effects, or phase transitions, making an extrapolation unreliable.*

To analyze this sensitivity, we normalize both loss and log-learning-rate, focusing on the scale-invariant landscape defined by the normalized coordinates:

$$\tilde{L}(n) = \frac{L(\nu; n) - L^*(\infty)}{An^{-\alpha}}, \qquad \tilde{\nu}(n) = \frac{\nu - \nu^*(\infty)}{Bn^{-\beta}}. \tag{6}$$

In these normalized coordinates, Equation (4) becomes:

$$\tilde{L}(n) = 1 + \frac{C}{2AB^2} n^{\alpha - 2\beta + \gamma} \left(\tilde{\nu}(n) - 1\right)^2. \tag{7}$$

The normalized loss curvature scales as n^κ, $\kappa = \alpha - 2\beta + \gamma$ quantifying loss sensitivity with n.

Definition 3 (Transfer Robustness Exponent κ) *We say that a parameterization under our ansatz (Equation (4)) exhibits* robust transfer *if*$\kappa = \alpha - 2\beta + \gamma \leq 0$*. The sign of* κ *controls how prediction errors propagate to loss at scale.* $\kappa < 0$ *means the landscape flattens, so errors in extrapolating* $\nu^*(n)$ *from small widths result in diminishing loss penalties at scale. By comparison,* $\kappa > 0$ *amplifies these errors and degrades transfer reliability, as the landscape sharpens with width.*

Taken together, these metrics capture three complementary axes of transfer: $\mathcal{R}(\infty)$ captures performance at scale, $\mathcal{E}$ quantifies loss predictability, and κ measures whether prediction errors amplify or dampen at large n. As we show later, these metrics can even be at odds with each other.

2. Empirically, we find this form well approximates the Hessian scaling for many parameterizations we considered.

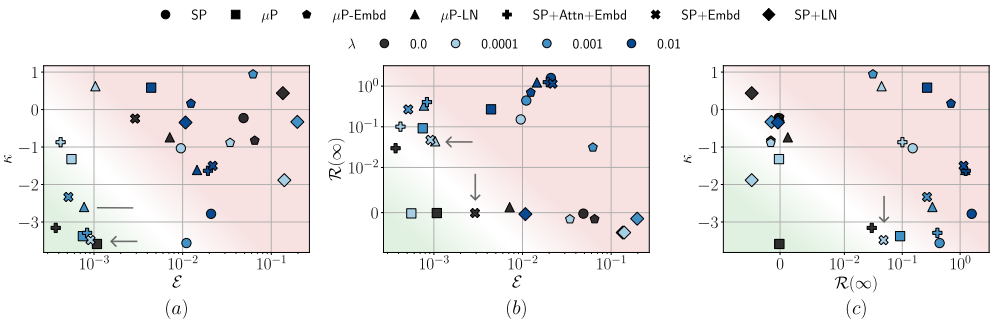


Figure 2: *Transfer metrics for parameterizations interpolating between SP and μP.* Parameterizations with '$+$' denote incremental changes from SP towards μP, while '$-$' denotes changes from μP towards SP. Green and red regions indicate desirable and undesirable regimes, respectively. The orange arrow highlights 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.

3. Examining μP and SP Through the Lens of the Three Transfer Metrics

We pre-train GPT-style Transformers on FineWeb-Edu [21] using AdamW for a fixed number of steps. We scale the embedding dimension (width) by increasing the number of heads, while keeping the head dimension fixed. For each width, we sweep the peak learning rate and weight decay strength λ. For full experimental details, see Section A. We detail the procedure for fitting the scaling laws and estimating the three transfer metrics from loss observations in Section D.

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 6 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 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.

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 B), 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 2 shows the three transfer metrics for selected ablations and weight decay values. We defer the full results to Figure 15 in Section E and summarize the key findings here.

The embedding layer learning rate emerges as the most critical modification: 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. 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.

In Section F, we run switch experiments that change the embedding layer learning rate at step t_{switch}, and find that the embedding layer learning rate is most critical early in training, and that training it too slowly not only slows down learning but also results in training instabilities.

Weight Decay and Compute-Optimal Training. In Section G, we study the effect of weight decay and find that it improves loss predictability $\mathcal{E}$ but worsens asymptotic loss $\mathcal{R}(\infty)$ in the fixed-step setting. In the compute-optimal (fixed tokens-per-parameter) regime, weight decay instead degrades transfer robustness κ, suggesting the standard $\eta \cdot \lambda = \Theta(1)$ convention is inadequate when training steps scale with width.

4. Related Works

Our work is closely related to several recent works on hyperparameter transfer [2, 6, 7, 15]. Ghosh et al. [7] analyze when transfer is computationally useful, formalizing *fast transfer* as the regime where the optimal hyperparameter converges faster with scale than the loss. Our condition of robust transfer $\kappa \leq 0$ is related to their *fast transfer*, with the two conditions coinciding when $\gamma = 0$. Kosson et al. [15] 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 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. [2] 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 may be needed to fully resolve the transfer robustness degradation we observe, leaving the appropriate weight decay scaling as an open question.

5. 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 given 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=sipwrPCrIS.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] Team Llama. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[26] 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, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

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

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

[29] 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, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Bx6qKuBM2AD.

Table 1: Comparison of Standard (SP) and Maximal Update Parameterization (μP) for AdamW.

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	$1/d$	–	–	–

Appendix A. Experimental Details

We pre-trained GPT-style Transformers on FineWeb-Edu [21], building on the nanoGPT codebase [14]. All experiments use a depth of 12 Transformer blocks without biases, a context length of 1024, and 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 [10] (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 [9]. To ensure that small models are trained for a sufficient number of steps, we reduce the batch size to 256 ($\approx$ 0.25M tokens).

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.

Appendix B. Preliminaries: Neural Network Parameterizations

We consider a neural network f_θ with parameters θ trained by minimizing a loss $L(\theta)$ using AdamW optimizer [19] with learning rate η and weight decay λ. The performance of a model can be improved by scaling along different dimensions, such as width, depth, context length and data. A *parameterization* is then a set of rules that specify how key network and optimizer hyperparameters must be adjusted as the model is scaled to maintain stable training dynamics. In this work, we focus on the width n as the scaling dimension and the learning rate η as the hyperparameter to be scaled.

Following [28], 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 $1/\sqrt{d}$ [25].

Given this general setup, specific parameterizations are obtained by imposing stability conditions on the training dynamics. Two canonical examples of $abcd$ parameterization are SP [24] and μP [28]. SP requires that activations at initialization do not blow up or vanish as the width grows, imposing one constraint per layer. 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) finite number of training steps as the width $n \to \infty$, (2) full alignment between weight updates and the activations they act on [28], and (3) a fixed dataset as n scales.

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 [28] to set the multipliers $a = 0$ in our description of μP. As shown in Table 1, the two differ in four ways: (1) embedding layer learning rate, (2) last layer initialization variance, (3) LayerNorm learning rate, and (4) attention scaling.

The intuitive reason for these changes is as follows. SP applies a uniform $\Theta(1/n)$ learning rate to all layers, but the embedding and LayerNorm layers perform width-independent operations[3] and do not require this scaling, making $\Theta(1)$ the 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. However, in practice, learning rate warmup mitigates this by gradually reducing sharpness to a level determined by the peak learning rate [11], making the initialization difference between SP and μP negligible. 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.

Appendix C. Scaling Law Ansatz

To formalize the transfer metrics, we first need to model how the loss landscape changes with width and learning rate. Following standard practice in neural scaling laws [1, 9, 13], we model the optimal loss $L^*(n)$ as a power law in width:

$$L^*(n) = L^*(\infty) + An^{-\alpha}, \tag{8}$$

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-

3. Here, by width-independent operations we mean that these layers do not change the scale of activations and their updates.

optimal training. We model the optimal learning rate using a scaling law with an irreducible term as well:

$$\eta^*(n) = \eta^*(\infty) + B' n^{-\beta}, \tag{9}$$

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 [6, 17], which implicitly assumes that the optimal learning rate vanishes ($\beta' > 0$) or diverges ($\beta' < 0$) at large widths. In Equation (9), we assume the dominant scaling law has been peeled off, 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)). In log-learning-rate space $\nu := \log \eta$, Equation (9) takes the form:

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

Since Equation (9) captures the residual width dependence after the dominant scaling has been removed, the term $B' n^{-\beta}/\eta^*(\infty)$ is small, and to leading order:

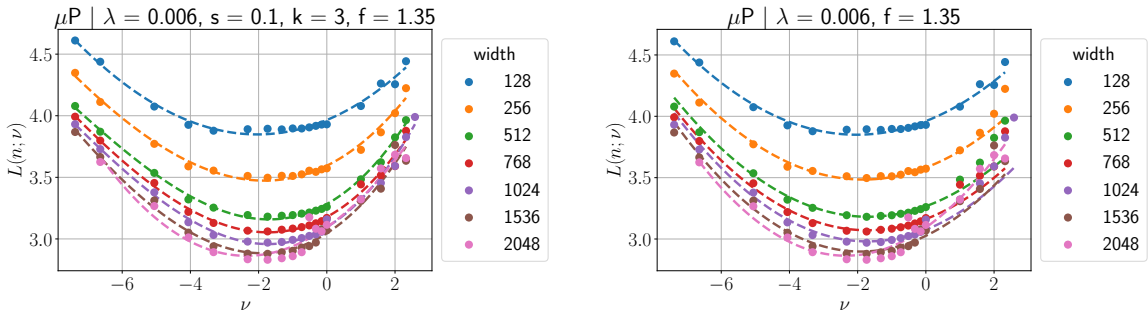
$$\nu^*(n) \approx \nu^*(\infty) + B n^{-\beta}, \tag{10}$$

where $B = B'/\eta^*(\infty)$. 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.

Appendix 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.

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 3(*a*)subfigure 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.



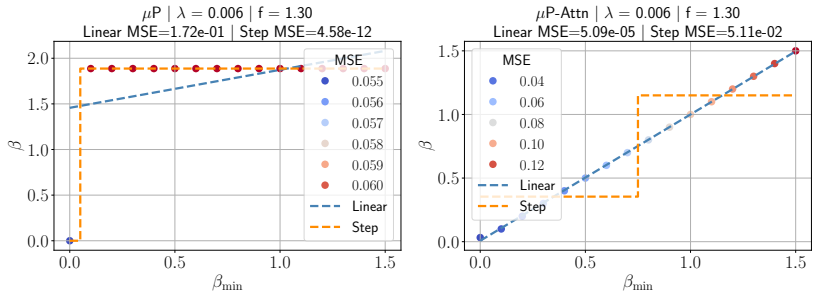
(a) (b)

Figure 3: (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).

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 (Theorem 1), we clamp fitted 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 3(*b*)subfigure), 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 1(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 (4) 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*) (*b*)

Figure 4: 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.

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 4). 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 (4)) 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 (4), and we therefore use it for reporting $\mathcal{E}$.

Appendix 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 5–14 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 3, μP and SP are surprisingly similar across most metrics (Figures 5 and 6). Despite being more unstable (a 3× 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 7). Conversely, training μP with a $\Theta(1/n)$ embedding learning rate (μP-Embd) destabilizes training entirely (Figure 8).

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 9): 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 10): $\mathcal{E}$ 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 11), 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 a 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 [11], effectively mitigating any initialization differences.

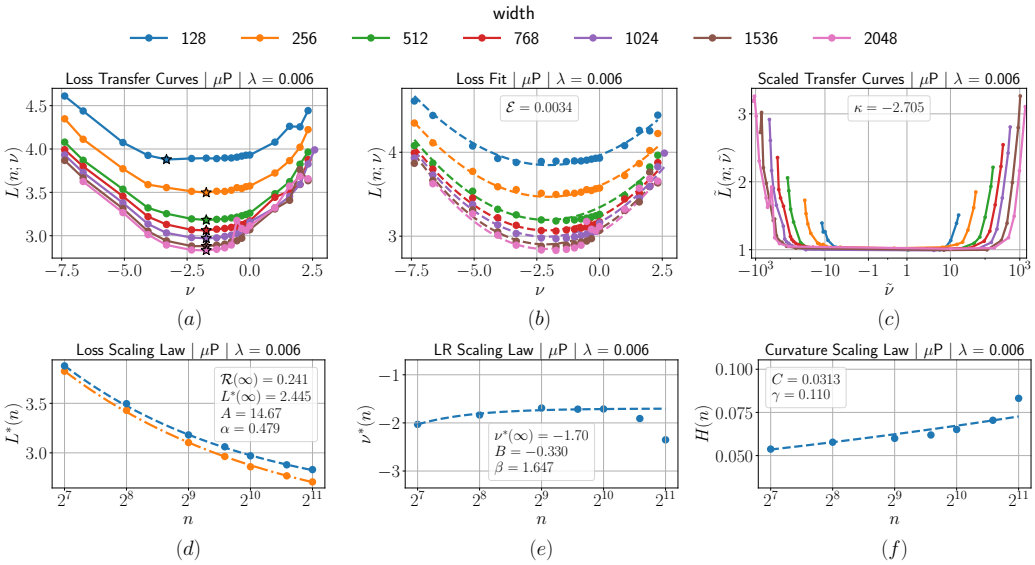


Figure 5: Transfer metrics for μP with weight decay $\lambda = 0.006$. Repeated Figure 1 for completeness.

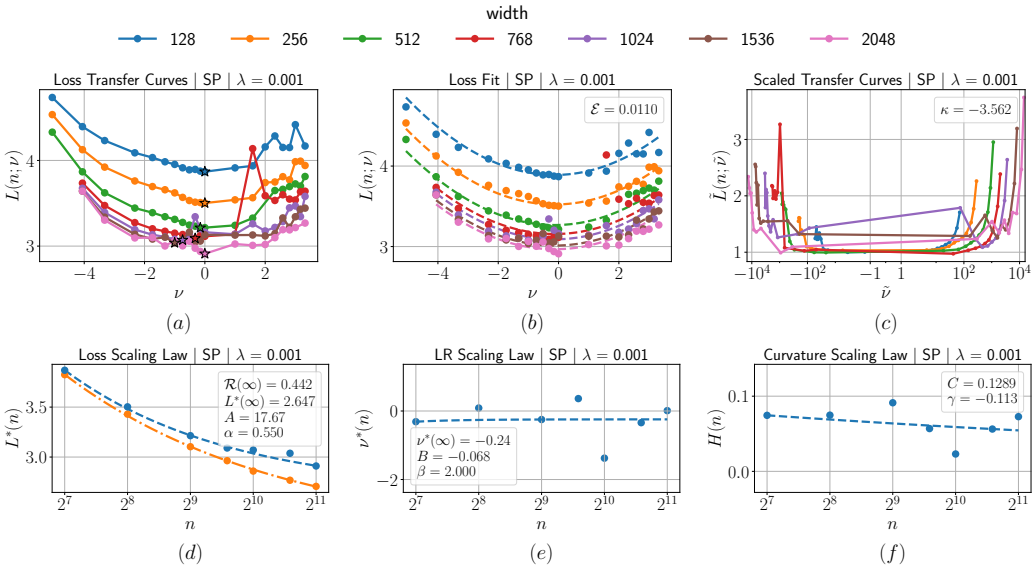


Figure 6: Transfer metrics for SP with weight decay $\lambda = 0.001$.

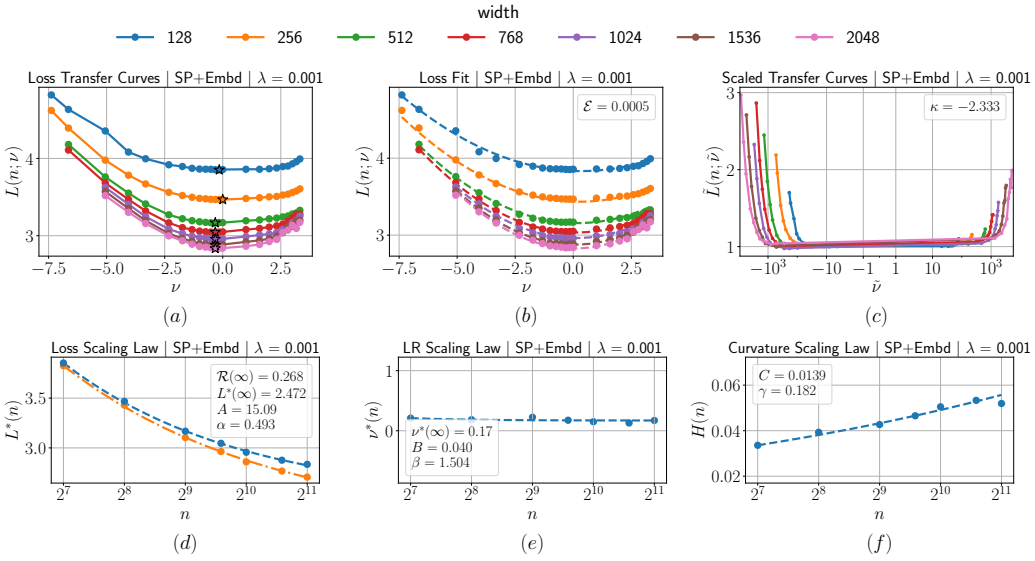


Figure 7: Transfer metrics for SP+Embd with weight decay $\lambda = 0.001$.

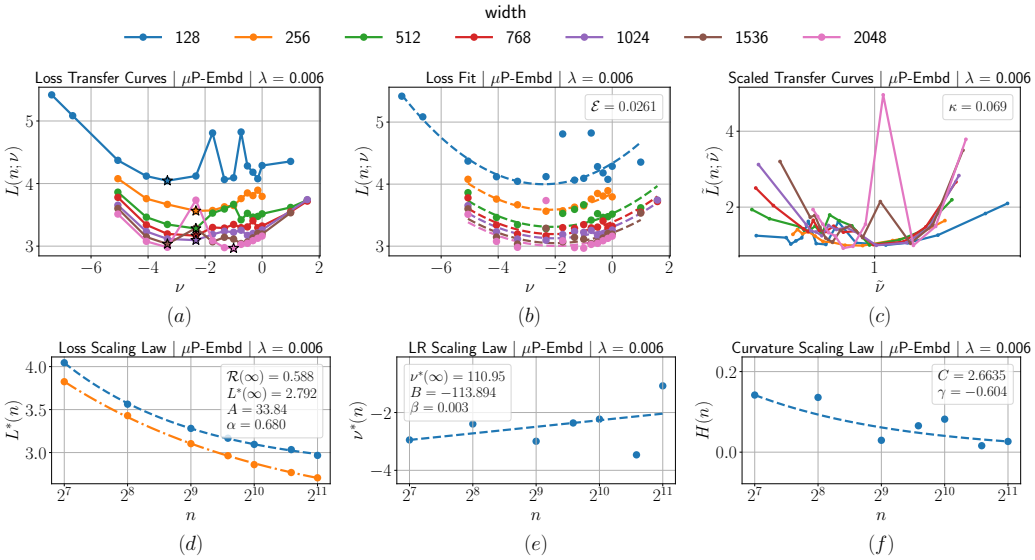


Figure 8: Transfer metrics for μP-Embd with weight decay $\lambda = 0.006$.

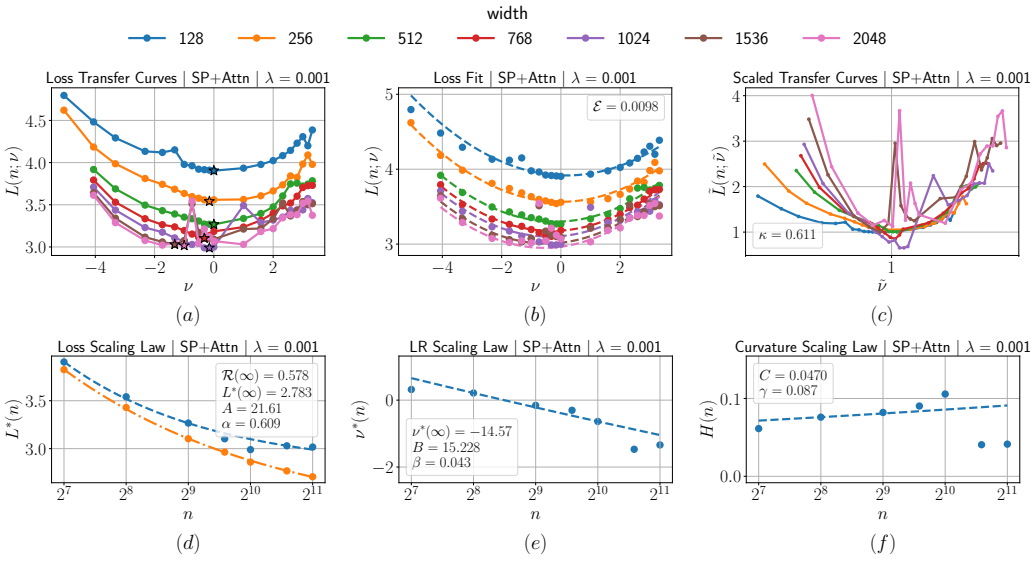


Figure 9: Transfer metrics for SP+Attn with weight decay $\lambda = 0.001$.

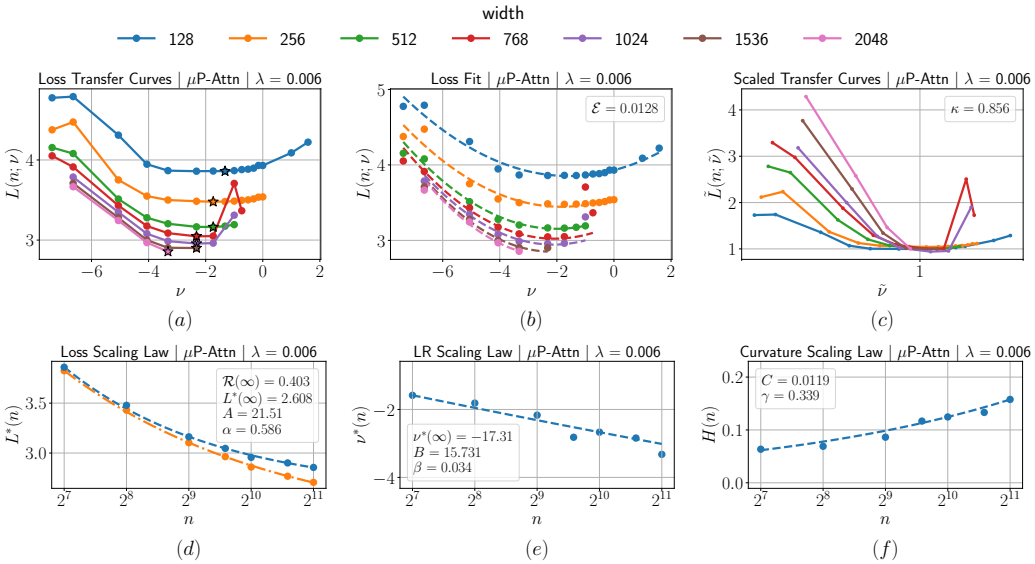


Figure 10: Transfer metrics for μP-Attn with weight decay $\lambda = 0.006$.

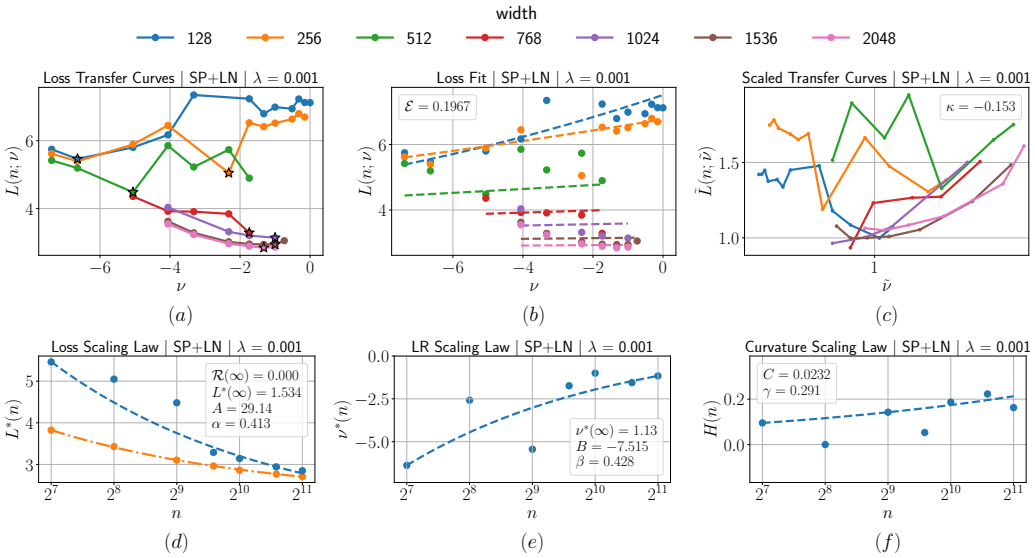


Figure 11: Transfer metrics for SP+LN with weight decay $\lambda = 0.001$.

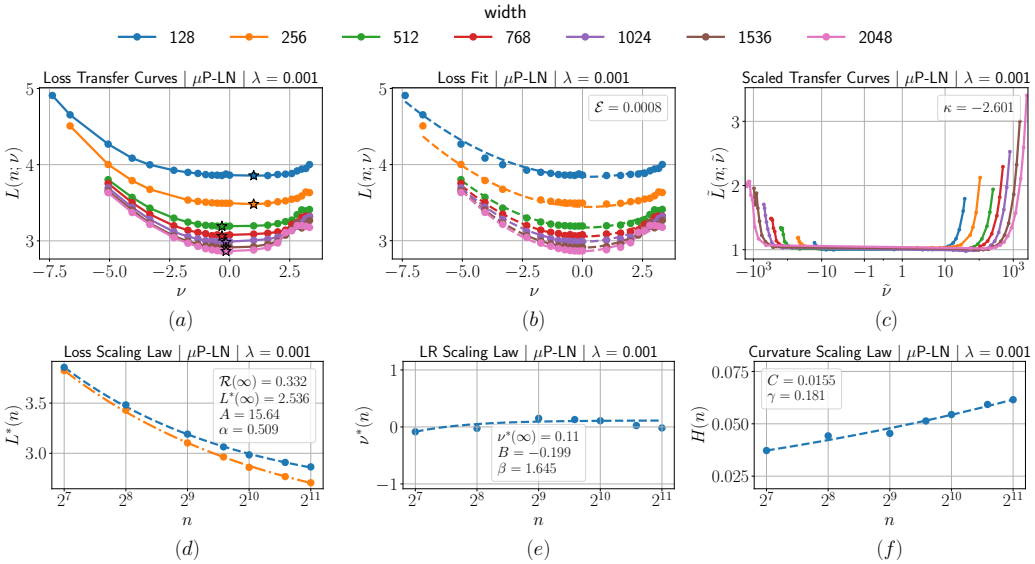


Figure 12: Transfer metrics for μP-LN with weight decay $\lambda = 0.001$.

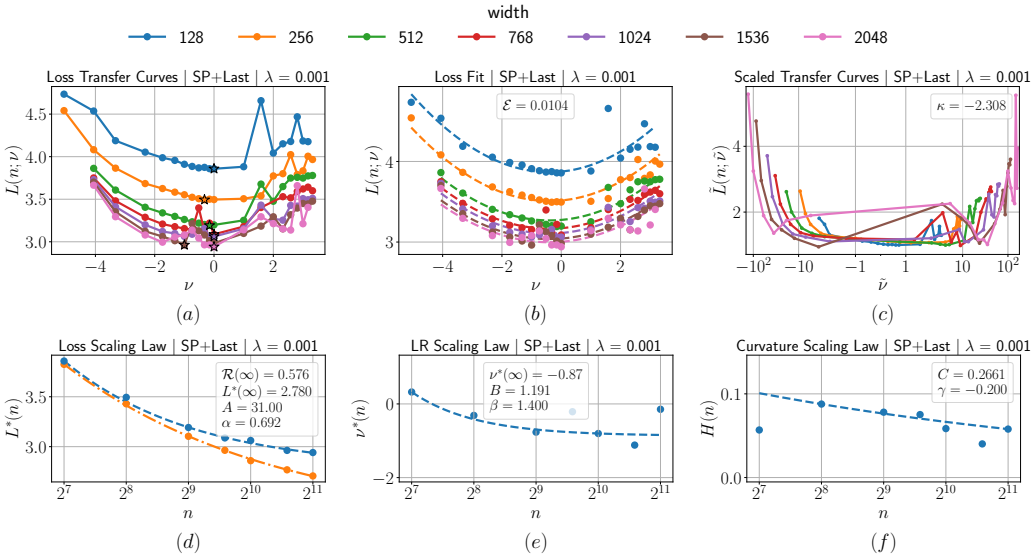


Figure 13: Transfer metrics for SP+Last with weight decay $\lambda = 0.001$.

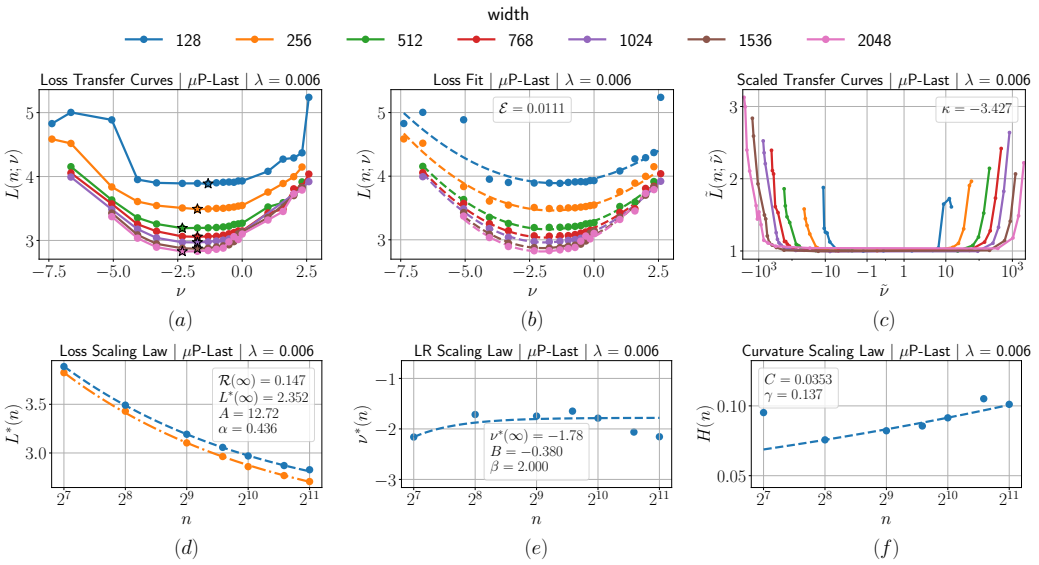


Figure 14: Transfer metrics for μP-Last with weight decay $\lambda = 0.006$.

E.2. Transfer Metric Phase Diagrams

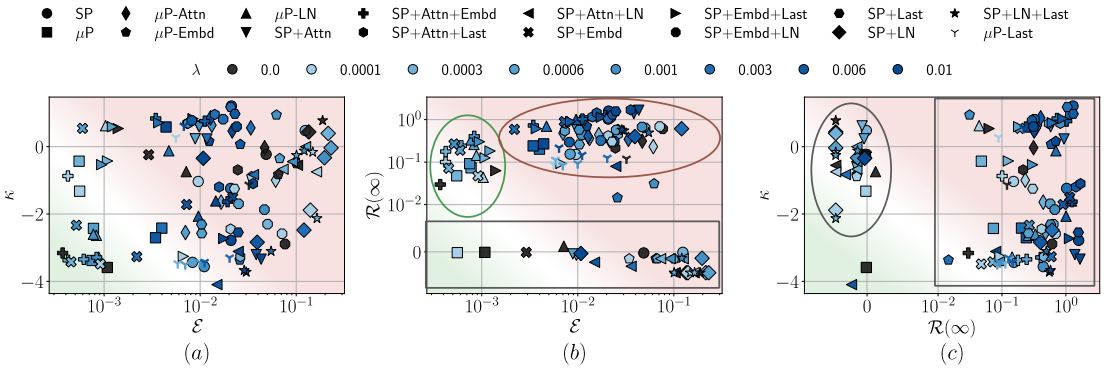


Figure 15: *Transfer metrics for parameterizations interpolating between SP and μP.* Parameterizations with '+' denote incremental changes from SP towards μP, while '−' denotes changes from μP towards SP. Green and red regions indicate desirable and undesirable regimes, respectively.

Figure 15 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 3: a parameterization can achieve excellent asymptotic performance while

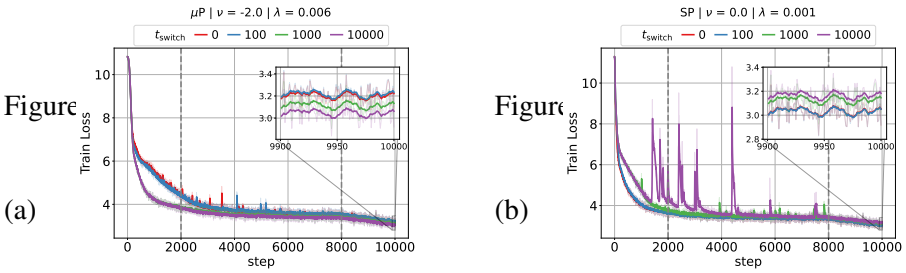


Figure 18: (a) Reducing the embedding learning rate from $\Theta(1)$ to $\Theta(1/n)$ in μP at step t_{switch} causes a persistent loss gap that grows larger for earlier switches (inset). (b) Increasing it from $\Theta(1/n)$ to $\Theta(1)$ in SP at step t_{switch} eliminates training instabilities and improves performance over SP ($t_{\text{switch}} = 10^4$).

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.

Appendix F. 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, as our results in Section 3 show, is that training it too slowly destabilizes the training, 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 learning rate not only slows down training but also causes instabilities, with early training being the most critical. In the μP case (Figure 18a), 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 18b), switching to $\Theta(1)$ embedding learning rate at an early stage simultaneously eliminates the training instabilities and improves performance. We additionally study the effect of completely freezing the embedding layer in Section K, finding that freezing hurts both parameterizations, but μP much more than SP. This indicates that later layers do not compensate for an untrained embedding.

One possible explanation for the outsized role of the embedding layer is that a slowly trained embedding puts excessive pressure on the first attention layer, which must extract useful features from nearly random token representations and, unlike later layers, has no upstream processing to rely on, making it sensitive to embedding quality. We also test whether the need to train the first layer sufficiently fast is specific to Transformers. To this end, in Section H, we show that for CNNs trained on CIFAR-100 [16], SP with a $\Theta(1)$ input-layer learning rate matches μP's transfer quality.

Appendix G. 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 [6, 29]. While recent work has begun examining the effect of weight decay [15] and compute-optimal scaling regime [2], 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 [9]), 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 19a). The effect on loss predictability error $\mathcal{E}$ is more nuanced and depends on the baseline stability of the parameterization without weight decay (Figure 19b). 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 (4)). The transfer robustness exponent κ remains largely negative and shows no clear dependence on λ (Figure 19c).

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 I, 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. We leave a detailed analysis of the appropriate weight decay scaling in this regime to future work.

Appendix H. Importance of the First Layer Learning Rate in CNN Image Classification

In Section F, 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

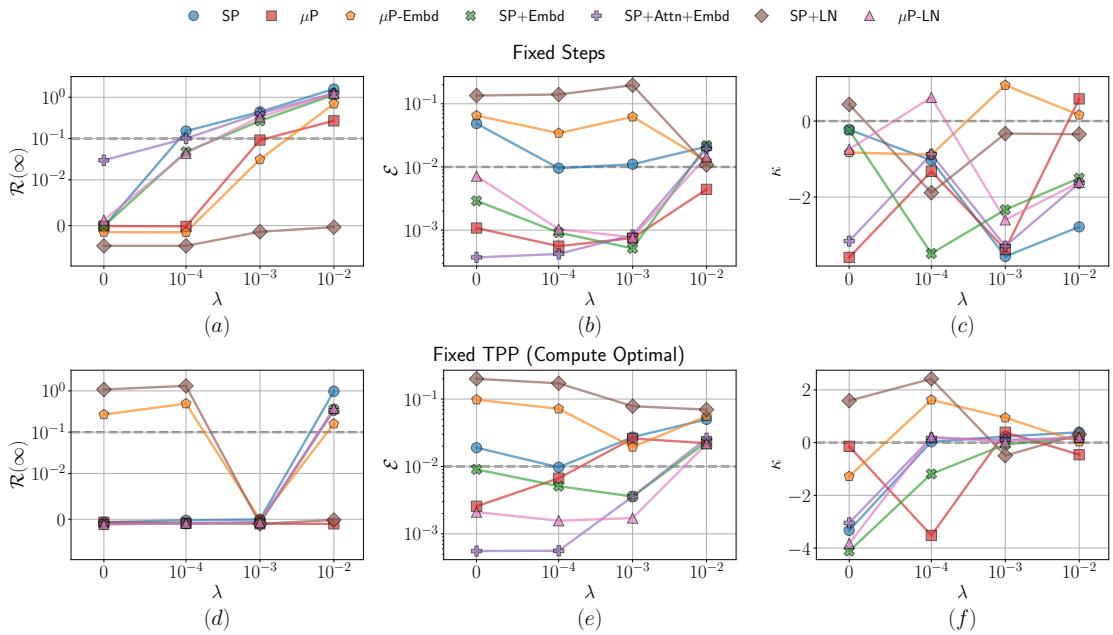


Figure 19: *Effect of weight decay on the three transfer metrics in the fixed-step (top) and fixed TPP (bottom) settings.* In the fixed-step setting, weight decay improves predictability error $\mathcal{E}$ at the cost of asymptotic performance $\mathcal{R}(\infty)$. In the TPP setting, stable parameterizations achieve near-zero $\mathcal{R}(\infty)$, and landscape predictability trends are similar, but robustness κ degrades as weight decay increases.

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.

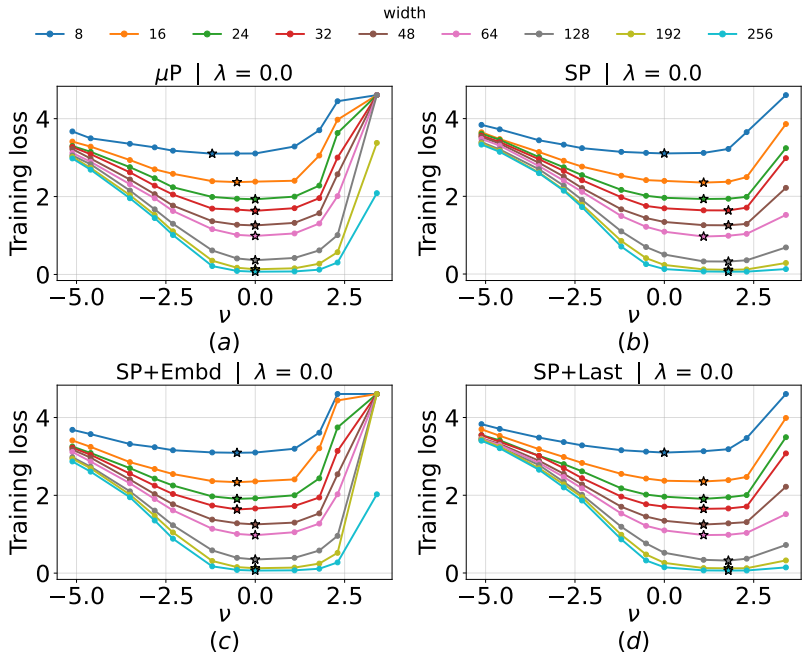


Figure 20: *CNN experiments on CIFAR with Adam.* Training loss as a function of log learning rate ν for four parameterizations across widths. In SP, the optimal learning rate drifts with width, while μP shows substantially less drift. Increasing the learning rate of the input-facing layer in SP (SP+Embd) 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.

Appendix I. Weight Decay scaling in Compute-Optimal Regime

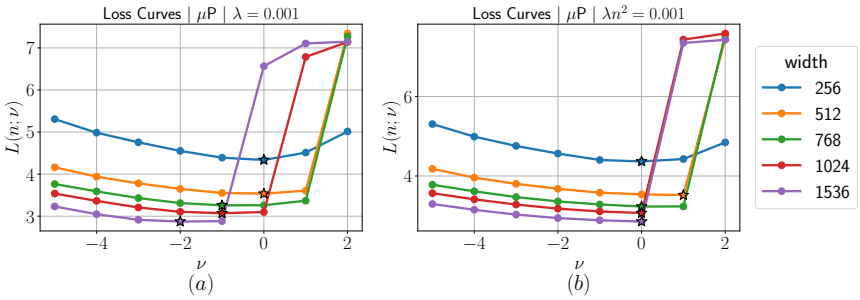


Figure 21: Loss curves for μP in the compute-optimal setting (20 TPP) under two weight decay scalings. (a) Standard μP convention $\eta \cdot \lambda = \Theta(1)$: $\nu^*(n)$ shifts to the left with increasing width. (b) Corrected convention $\eta \cdot \lambda = \Theta(1/n^2)$: $\nu^*(n)$ doesn't vary 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 G where transfer robustness κ degrades with increasing weight decay. A natural scaling, motivated by [2], is to scale weight decay as $\eta \cdot \lambda = \Theta(1/n^2)$, so that the effective regularization strength remains the same across different training steps.

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 G. 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.

Appendix J. The Effect of Learning Rate Warmup on Learning Rate Transfer

Learning rate warmup is standard practice in large-scale training [5, 20]. Its primary effect is gradually reducing the sharpness of the loss Hessian (or pre-conditioned sharpness for Adam), effectively steering optimization towards well-conditioned regions of the loss landscape where the model can be trained at large learning rates [8, 11]. 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 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.

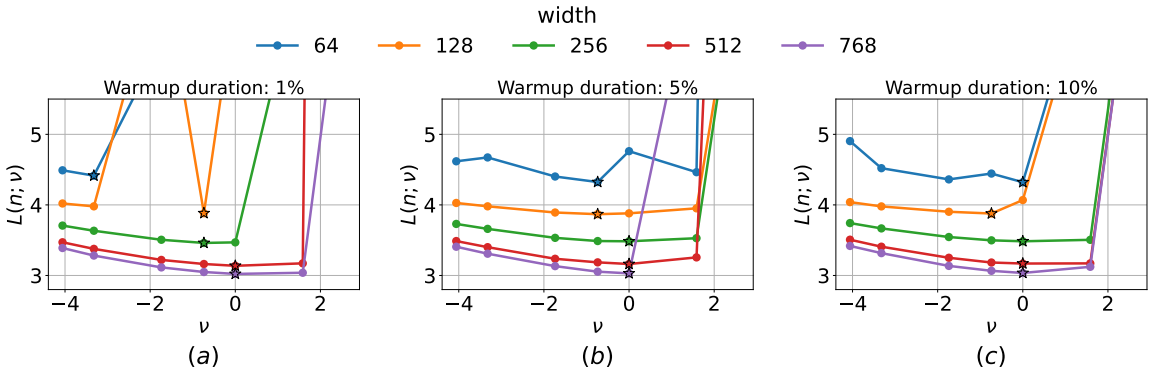


Figure 22: Learning rate warmup is essential for observing reliable learning rate transfer in μP.

Appendix K. Effect of Freezing the Embedding Layer

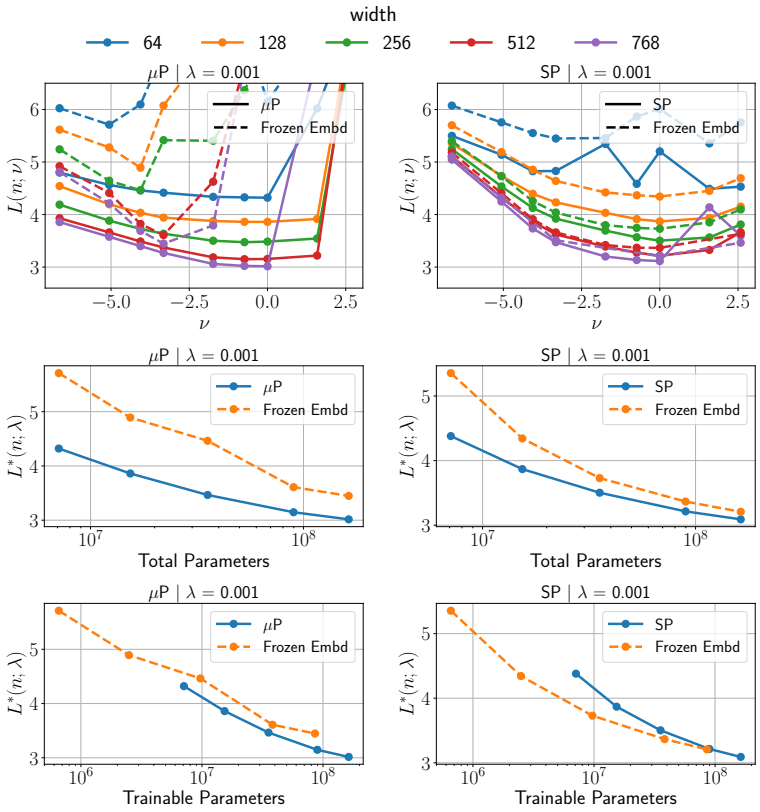


Figure 23: Effect of freezing the embedding layer in μP and SP. (top) Loss vs. log learning rate curves. (middle) Optimal loss vs. total parameters. (bottom) Optimal loss vs. trainable parameters, where trainable parameters exclude the embedding when frozen.

The switch experiments in Section F 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.

Appendix L. Effect of Training other Layers Slowly

In Section 3, we showed that training the embedding layer slowly not only can slow down training, but can also result in training instabilities. In this section, we analyze the effect of training other layers slowly. Specifically, we train the hidden and last layer with a learning rate of $\Theta(1/n^2)$ instead of $\Theta(1/n)$.