Warmstarting for Scaling Language Models

Neeratyoy Mallik[*]
University of Freiburg

Maciej Janowski[*]
University of Freiburg
University of Technology Nuremberg

Johannes Hog
University of Freiburg

Herilalaina Rakotoarison
University of Freiburg

Aaron Klein
ScaDS.AI Leipzig

Josif Grabocka
University of Technology Nuremberg

Frank Hutter
ELLIS Institute Tübingen
University of Freiburg

Abstract

Scaling model sizes to scale performance has worked remarkably well for the current large language models paradigm. The research and empirical findings of various scaling studies led to novel scaling results and laws that guides subsequent research. However, prohibitively high training costs at contemporary scales of data and models result in a lack of thorough understanding of how to tune and arrive at such training setups efficiently. One direction to ameliorate the cost of pretraining large models is to *warmstart* the large-scale training from smaller models that are cheaper to tune. In this work, we attempt to understand if the behavior of optimal hyperparameters can be retained under warmstarting for scaling. We explore simple operations that allow the application of theoretically motivated methods of zero-shot transfer of optimal hyperparameters using μTransfer. We investigate the aspects that contribute to the speedup in convergence and the preservation of stable training dynamics under warmstarting with μTransfer. We find that shrinking smaller model weights, zero-padding, and perturbing the resulting larger model with scaled initialization from μP enables effective warmstarting of μTransfer.

1 Introduction

Scaling of model size, dataset size and training compute together, lead to performance scaling in the large language model (LLMs) paradigm, as shown by the recent scaling law literature [Kaplan et al., 2020, Hoffmann et al., 2022, Caballero et al., 2023, Hägele et al., 2024, Porian et al., 2024]. Under different training setups, empirical data often fit predictable trends captured by an exponent of an assumed parametric relationship. Recent research trends point to an attempt at understanding and finetuning the nature of scaling law studies across various problem formulations [Sorscher et al., 2022, Alabdulmohsin et al., 2023, Wang et al., 2023a].

One commonality across such studies is the often non-overlapping choice of hyperparameters and sparse documentation of their selection process. This stems from the prohibitive costs of principled hyperparameter tuning at large scales. Reusing weights from tuned smaller models can offer a way to accelerate training of larger models. This is commonly known as *warmstarting* a model's training and has been approached previously through knowledge distillation [Chen et al., 2016, 2021], morphisms [Wei et al., 2016a, Elsken et al., 2019], learned transformations [Wang et al., 2023b,a],

[*]equal contribution, {mallik,janowski}@cs.uni-freiburg.de

shrink-and-perturb [Ash et al., 2020, Chebykin et al., 2023, Shin et al., 2024, Samragh et al., 2024], heuristics [Rae et al., 2021]. However, the change in optimality of hyperparameters when scaling up a model is not usually discussed alongside such sophisticated warmstarting techniques.

The growing literature on scaled parameterizations looks at the hyperparameters that directly affect a model's ability to learn features as a function of its growing model scale [Yang et al., 2021, Everett et al., 2024, Blake et al., 2024, Thérien et al., 2024, Yang et al., 2024]. μTransfer [Yang et al., 2021] offers analytical scaling relations that scale the parameters for a small model: learning rate and initialization standard deviation, so that it remains optimal even for the larger model. Our primary inquiry is to understand whether *warmstarting* μP for a large model using a *tuned* smaller model would offer any improvements over vanilla-μP. Our contributions through this work are as followed:

- We identify a simple method that warmstarts μP runs of larger models, improving convergence speed and in certain cases final performance (RQ1 and RQ2 in Section 3);
- Demonstrate that warmstarting retains the μP training stability guarantees in practice, with respect to model scaling along specific dimensions, such as width[2] (RQ3 in Section 3).

The importance of a working *warmstarting* method lies in its application to continually scale models with data, speeding up convergence. We believe that enabling warmstarted-μP can also substantially accelerate the tuning of non-μ-parameterizable hyperparameters for large models.

In the next section, we briefly highlight the method (Section 2) we find to be the most simple and yet in line with scaled parameterizations, allowing us to leverage algorithms such as μP. We then show our empirical findings that validate our approach (Section 3). Appendix A onwards contain various supporting information and details.

2 Method: Warmed-μP

In this section, we formalize and formulate warmstarting in the context of scaled parameterizations. We refer the readers to Appendix A for more on background concepts and related work.

Given a trained *base* model $\mathcal{M}_{\text{base}}$ with tuned hyperparameters (e.g. learning rate) and a *target* model $\mathcal{M}_{\text{target}}$ for transferring the hyperparameters, we define the *warmstarting* operation as the layer-wise initialization of $\mathcal{M}_{\text{target}}$ using $\mathcal{M}_{\text{base}}$. In this work, we initialize each layer of $\mathcal{M}_{\text{target}}$ using a combination of a shrunk version of $\mathcal{M}_{\text{base}}$ weights with the standard μP initialization. For a layer l in $\mathcal{M}_{\text{target}}$, let $\theta^l_{\text{target}} \in \mathbb{R}^{p\times q}$ be its weight and $\theta^l_{\text{base}} \in \mathbb{R}^{m\times n}$ be the corresponding weight from $\mathcal{M}_{\text{base}}$ (where $m \leq p$ and $n \leq q$). Formally, the warmstarting operation is given by:

$$\theta^l_{\text{target}} \;=\; \lambda_{\text{shrink}} \cdot \texttt{Pad}_0(\theta^l_{\text{base}}, p, q) + \mathcal{N}(0,\ {\sigma^l_{\mu\text{P}}}^2), \tag{1}$$

where $\sigma^l_{\mu\text{P}}$ is the recommended per-layer standard deviation by Yang et al. [2021] for initialization, and $\texttt{Pad}_0$ is a function that expands the *base* model weight to the *target* shape ($p \times q$) with zero padding. Intuitively, the initialization term of Equation 1 can be replaced with any scaling parameterization technique, while the first term can represent any transformation of the base weight matrices onto the target shape. Additionally, setting $\lambda_{\text{shrink}} = 0$ in Equation 1 recovers vanilla-μP.

We choose zero-padding as the simplest method to study for scaling the model size with μP and avoid more sophisticated heuristics [Rae et al., 2021], learned transformations [Wang et al., 2023b] or distillations [Chen et al., 2016]. In order to retain μP's guarantees of training stability, $\lambda_{\text{shrink}} \in [0,\ 1]$ is multiplied to all the transformed weights. The μP initialization acts as a *perturbation* matrix for the *base* model weights, transformed and casted to the *target* shape. We choose a fixed value of $\lambda_{\text{shrink}} = 0.4$ in our experiments (Section 3) as per *shrink-and-perturb*-based *warmstarting* literature [Ash et al., 2020, Zaidi et al., 2022, Chebykin et al., 2023] from the continual learning setting without model size growth. We find empirically that this recommendation interestingly holds and is crucial for warmstarted-μP (see, Section 3, Appendix B.2).

From a different perspective, one can also see the *warmstarting* operation as equivalent to *abc*-parameterization, which defines the family of scaled parameterization methods [Yang and Hu, 2021, Blake et al., 2024, Everett et al., 2024] that derive fixed scaling rules for hyperparameter transfer

[2]in this work, we look at model-width as the only scaling dimension, see Table 1

[Yang et al., 2021]. Scaled parameterization refers to modifying the initialization ($\mathcal{B}_w$), weight parameter scaling ($\mathcal{A}_w$) and learning rate ($\mathcal{C}_w$) as a function of model scales (see, Appendix A). Designing Equation 1 in a lean manner opens up the direction of phrasing λ_{shrink} as $\mathcal{A}_w$, while the overall scaled initialization can now be, $\sim \mathcal{N}(w_0', \mathcal{B}_w^2)$, where $w_0' = \lambda_{\text{shrink}} \cdot \texttt{Pad}_0(\theta_{\text{base}}^l, p, q)$. The next section shows empirically the gains provided by such simple adjustments and why studying and understanding warmstarted-μP more is relevant.

3 Empirical evaluation

In this section, we report our empirical findings, with the intention of emphasizing the underline intuition behind the proposed approach (Section 2), in both convergence speed and training stability.

Experimental Setup : We train a decoder-only GPT2 model [Radford et al., 2019] as per the GPT-NeoX implementation [Andonian et al., 2023] provided by LitGPT [AI, 2023] on the SlimPajama [Soboleva et al., 2023]'s 6B version[3] dataset. We set the weight decay to 0 for Adam [Kingma and Ba, 2015] to avoid interaction effects and confounding factors [Lingle, 2024, Wang and Aitchison, 2024]. Fixed learning rate (LR) schedules were used following Hägele et al. [2024], simplifying checkpoint studies with no LR management overheads. We disable learning rate warmup to aid comparative analysis of *if* warmstarting provides gains. All models are trained for 20 tokens/parameter, similar to Hoffmann et al. [2022], keeping sub-epoch training, that is, no micro batch of the dataset is ever repeated. Our FLOPs calculation is also based on Hoffmann et al. [2022]. Validation loss is measured on a fixed held-out split of the dataset, consistent across all runs, with 3 different seeds per run, and Gaussian smoothing is used for loss comparison across runs. For *warmstarting* runs, we do not repeat tokens seen by the *base* model. A *warmstarting* training starts with a fresh optimizer state, and follows standard μP training procedures. All trainings used a single NVIDIA RTX 2080 GPU.

RQ 1: Does warmstarting improve a vanilla-μP run? The μP recipe recommends the following broadly for μTransfer: i) Perform a grid search over hyperparameters at a small model scale; ii) Transfer the optimal hyperparameters at this scale, to a larger model scale using μP rules. The exact choice of *base* and *target* scales, choice of hyperparameters for grid search, if the base scale itself should be a hyperparameter, etc. are all actively pursued investigations in the community [Everett et al., 2024, Blake et al., 2024, Qiu et al., 2024, Lingle, 2024]. Given our experimental setup of fixed model scales across width scaling (refer, Table 1), we choose 3 possible *base* scales: 5M, 10M and 22M. Figure 1 shows μTransfer from the smallest base scale with and without warm-starting. Figure 4 shows the same result as Figure 1 but under longer compute or more tokens than recommended by Hoffmann et al. [2022]. Figure 5 for *warmstarting* from other base scales and Figure 6 for the learning curves that include the base model training.

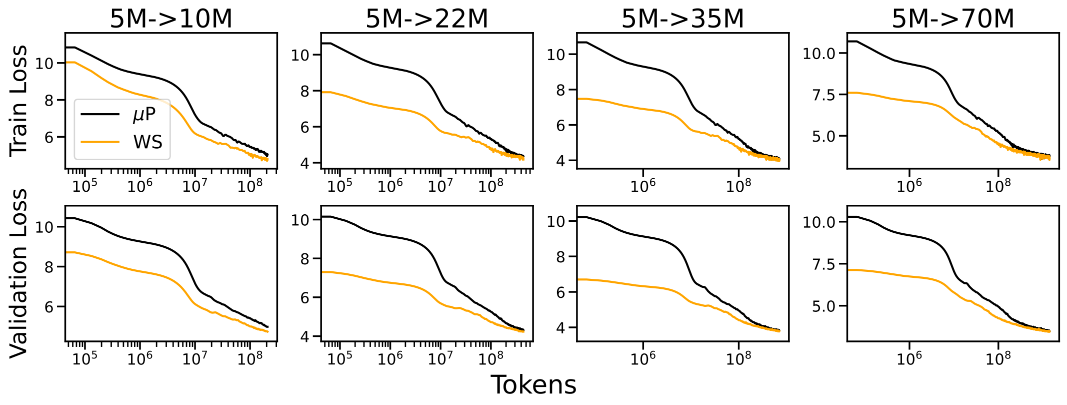

Figure 1: Transferring the best found learning rate at the base scale of 5M using μP. For *warmstarting* (WS) run, the model weights of the optimal 5M model is used to initialize the *target* model's training. Warmstarting appears to always improve μP convergence rates.

[3] https://huggingface.co/datasets/DKYoon/SlimPajama-6B

Warmstarting (WS), even across much larger model sizes yield a gain in initial performance, while either matching or improving the vanilla-μP performance under equivalent compute (as per Hoffmann et al. [2022]). Here, we compare the runs over the training compute as required by the target model. Both μP and warmstarting runs require a grid search from the *base* scale (5M) and hence are expected to share similar compute, and thus can be omitted from this analysis (Figure 1).

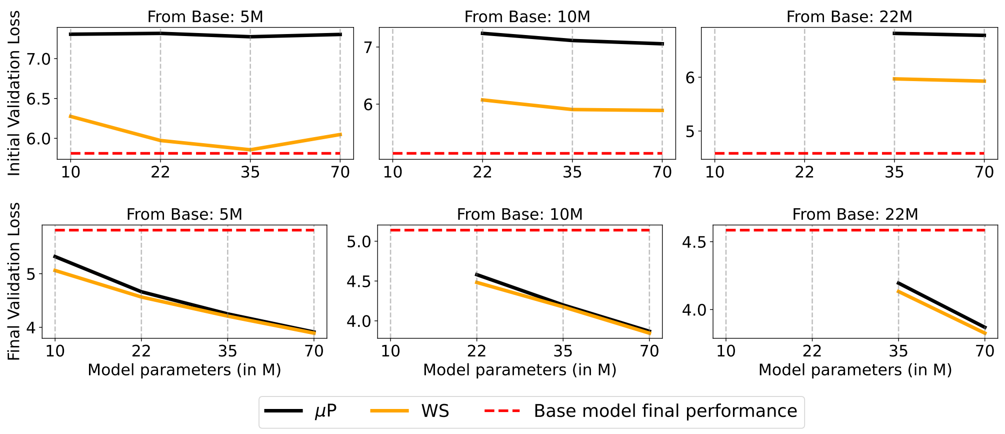

Figure 2: Comparing losses across model scales. (*Left to right*): given a larger *base* model, transfer to higher model scales; (*Top*): Shows the initial validation loss of the warmstarted model vs. vanilla-μP, where *warmstarting* always leads to improved initial loss; (*Bottom*): Shows the final validation loss of the warmstarted model vs. vanilla-μP run, which achieves better or equivalent loss.

RQ 2: How does the scale difference affect the quality of warmstarting? Given that the warmstarting operation (Section 2) chosen for our experiments pads the new connections to the *base* model with zeros to scale to the *target* model, it can be expected that the quality of warmstarting will be more pronounced when a *large enough* model is grown not *too large*. In Figure 2 we chart the initial and final validation loss comparison with μTransfer and warmstarting under width-scaling. We observe that warmstarting offers a much-improved initialization that accelerates loss convergence. It is also evident that the loss obtained by the *base* model is not transferred. This gap manifests as *spikes* if learning curves are concatenated over the *base* and warmstarted-*target* runs (see Figure 6). Figure 2 also highlights that the amount of data already seen by the base model, and the loss it can achieve given its scale, skews such an analysis as the resulting warmstarted model is likely to have a larger spike in loss. Given that we do not repeat the tokens seen by the *base* model when training the warmstarted *target* model, a larger spike would require more updates and thereby tokens to recover. We believe that though the *warmstarting* method shown here is adequate in providing speed ups, it is suboptimal in not providing consistent *improvement* in final loss, despite seeing more tokens in principle. Despite the obvious gains *warmstarting* can bring with μP, there are clearly much more design choices to study and consider for consistent, efficient warmstarting.

RQ 3: Does warmstarting with μP retain training stability guarantees? We empirically investigate the impact of the proposed *warmstarting* technique on μP training behavior [Yang et al., 2021]. We conduct coordinate checks as per the μP library[4] to verify the consistency of the L_1-norm of layer activations across width-scaling. Within the same experimental setting, we also ablate the effect of the shrinking value, the main hyperparameter of our approach.

Figure 3 shows L1-norm of activations for warmstarted μP with $\lambda_{\text{shrink}} \leq 0.6$ trend similar to vanilla μP across the considered width-scaling. However, we observe instability for $\lambda_{\text{shrink}} > 0.6$ (Figure 7, 10). This result also backs the $\lambda_{\text{shrink}} = 0.4$ value reported for *shrink-and-perturb* in the literature [Ash et al., 2020, Zaidi et al., 2022, Chebykin et al., 2023]. Additionally, we monitor the L1 norm of layer activations during the entire training in Figure 8. Warmstarted μP maintains activation values within a comparable range and trends consistent with vanilla-μP, suggesting that our demonstrated warmstarting method does not introduce instability to μP.

[4]`https://github.com/microsoft/mup?tab=readme-ov-file%23coord-check`

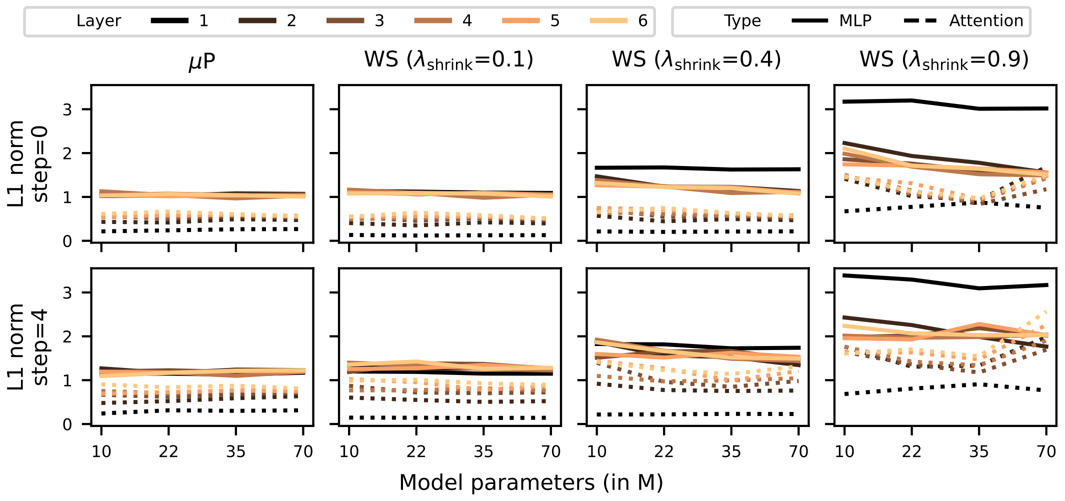

Figure 3: L1 norm of the layers activation across scales. Any warmstarted μP, having $\lambda_{\text{shrink}} \leq 0.6$, behaves well in scale as in μP (detailed results in Figure 7 of Appendix B).

4 Conclusion

In this work, we explore the feasibility of enabling continued pretraining of language models as more data is provided and model size is scaled up. We demonstrate that a simple technique, even when warmstarting from a much smaller model, is adequate in improving convergence speed and potentially final performance (Figures 1, 5). Our focus was to identify a warmstarting method that is simple enough to be implemented alongside μP (Section 2) while not being in conflict with μTransfer's expectations of stable training in practice (Figures 3, 7, 8, 10).

Limitations. We note that the empirical study shown here has an outcome of interest in that simple shrinking and perturbation with μP works already. However, our setting (Section 3) is carefully chosen to minimize effects induced by various choices of training setups, in order to understand the effects of *warmstarting* with μP. For more general practicality, more ablations and experiments must be performed over different learning rate schedules, weight decay, activations, and for much larger model sizes and context windows. More recent works on scaled parameterizations [Everett et al., 2024, Blake et al., 2024] which study and improve over μP recommendations should also be tested with our modular approach to further assert the potential of *warmstarting* under *abc*-parameterization.

Future directions. More diverse experiments aside, exploring what makes *warmstarting* work, as we demonstrate here, is important for a lean, general, efficient method that truly speeds up pretraining runs. Figures 7 and 10 show the effect of different shrinking factors in maintaining training stability of optimal hyperparameters, while Figure 2 shows that choice of scales directly affect gains provided by *warmstarting*. Exploiting the definition of Equation 1, we would like to explore the role of *layer-wise* shrinking of base model weights and investigate theoretical relations with *abc*-parameterization. Exploiting structures, ranks, and representations learned by the base model can also lead to better warmstarting and improvement over simple zero padding [Rae et al., 2021, Wang et al., 2023a, Qiu et al., 2024, Wei et al., 2024]. Moreover, warmstarting using our setup here leads to the consumption of more tokens, when comparing against the vanilla-μP training of the *target* model. Exploring if models can be warmstarted and scaled up progressively will be a clear direction to pursue (see, Appendix B.3). Such a paradigm of training can massively lower hyperparameter tuning costs overall and also potentially change the practice of scaling models up, with tuned hyperparameters.

Acknowledgements. **FH** acknowledges the financial support of the Hector Foundation. **NM**, **JH**, **HR**, **FH** acknowledge funding by the state of Baden-Württemberg through bwHPC, the German Research Foundation (DFG) through grant numbers INST 39/963-1 FUGG and 417962828, and the European Union (via ERC Consolidator Grant Deep Learning 2.0, grant no. 101045765), TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. **JH** and **FH** acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under SFB 1597 (SmallData), grant number 499552394. **MJ** and **JG** acknowledge funding by The Carl Zeiss Foundation through the research network "Responsive and Scalable Learning for Robots Assisting Humans" (ReScaLe) of the University of Freiburg. **AK** acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

Funded by
the European Union

References

J. Kaplan, S. McCandlish, T. Henighan, T. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv:2001.08361 [cs.LG]*, 2020.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models. 2022.

E. Caballero, K. Gupta, I. Rish, and D. Krueger. Broken neural scaling laws. In *International Conference on Learning Representations (ICLR'23)* icl [2023]. Published online: `iclr.cc`.

A. Hägele, E. Bakouch, A. Kosson, L. Ben allal, L. Von Werra, and M. Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024.

R. Porian, M. Wortsman, J. Jitsev, L. Schmidt, and Y. Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024.

B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli, and A. S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*, 2022.

I. Alabdulmohsin, X. Zhai, A. Kolesnikov, and L. Beyer. Getting Vit in Shape: Scaling laws for compute-optimal model design. In *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*, 2023.

P. Wang, R. Panda, and Z. Wang. Data efficient neural scaling law via model reusing. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, volume 202 of *Proceedings of Machine Learning Research*, pages 36193–36204. PMLR, 2023a.

T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *Proceedings of the International Conference on Learning Representations (ICLR'16)* icl [2016]. Published online: `iclr.cc`.

C. Chen, Y. Yin, L. Shang, X. Jiang, Y. Qin, F. Wang, Z. Wang, X. Chen, Z. Liu, and Q. Liu. bert2bert: Towards reusable pretrained language models. *arXiv preprint arXiv:2110.07143*, 2021.

T. Wei, C. Wang, Y. Rui, and C. W. Chen. Network morphism. In *International conference on machine learning* icl [2016], pages 564–572. Published online: `iclr.cc`.

T. Elsken, J. Metzen, and F. Hutter. Efficient multi-objective Neural Architecture Search via lamarckian evolution. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*, 2019. Published online: `iclr.cc`.

P. Wang, R. Panda, L. T. Hennigen, P. Greengard, L. Karlinsky, R. Feris, D. D. Cox, Z. Wang, and Y. Kim. Learning to grow pretrained models for efficient transformer training. In *International Conference on Learning Representations (ICLR'23)* icl [2023]. Published online: `iclr.cc`.

J. Ash, R. P. Adams, and D. Author. On warm-starting neural network training. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors, *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020.

A. Chebykin, A. Dushatskiy, T. Alderliesten, and P. A. N. Bosman. Shrink-perturb improves architecture mixing during population based training for neural architecture search. In *ECAI 2023 - 26th European Conference on Artificial Intelligence, including 12th Conference on Prestigious Applications of Intelligent Systems, PAIS 2023 - Proceedings*, pages 381–388, 2023.

B. Shin, J. Oh, H. Cho, and C. Yun. Dash: Warm-starting neural network training without loss of plasticity under stationarity. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ ICML 2024)* icm [2024].

M. Samragh, I. Mirzadeh, K. A. Vahid, F. Faghri, M. Cho, M. Nabi, D. Naik, and M. Farajtabar. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*, 2024.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

G. Yang, E. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In Ranzato et al. [2021].

K. E. Everett, L. Xiao, M. Wortsman, A. A. Alemi, R. Novak, P. J. Liu, I. Gur, J. Sohl-Dickstein, L. P. Kaelbling, J. Lee, and J. Pennington. Scaling exponents across parameterizations and optimizers. In *Forty-first International Conference on Machine Learning*, 2024.

C. Blake, C. Eichenberg, J. Dean, L. Balles, L. Y. Prince, B. Deiseroth, A. F. Cruz-Salinas, C. Luschi, S. Weinbach, and D. Orr. u-μp: The unit-scaled maximal update parametrization. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024.

B. Thérien, C. Joseph, B. Knyazev, E. Oyallon, I. Rish, and E. Belilovsky. μlo: Compute-efficient meta-generalization of learned optimizers. *arXiv preprint arXiv:2406.00153*, 2024.

G. Yang, D. Yu, C. Zhu, and S. Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *International Conference on Learning Representations (ICLR'24)* icl [2024]. Published online: `iclr.cc`.

S. Zaidi, T. Berariu, H. Kim, J. Bornschein, C. Clopath, Y. W. Teh, and R. Pascanu. When does re-initialization work? In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification@NeurIPS'22*, 2022.

G. Yang and E. J Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A. Andonian, Q. Anthony, S. Biderman, S. Black, P. Gali, L. Gao, E. Hallahan, J. Levy-Kramer, C. Leahy, L. Nestler, K. Parker, M. Pieler, J. Phang, S. Purohit, H. Schoelkopf, D. Stander, T. Songz, C. Tigges, B. Thérien, P. Wang, and S. Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch, 9 2023. URL `https://www.github.com/eleutherai/gpt-neox`.

Lightning AI. Litgpt. `https://github.com/Lightning-AI/litgpt`, 2023.

D. Soboleva, F. Al-Khateeb, R. Myers, J. R. Steeves, J. Hestness, and N. Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, June 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*, 2015. Published online: `iclr.cc`.

L. Lingle. A large-scale exploration of μ-transfer. *arXiv preprint arXiv:2404.05728*, 2024.

Xi. Wang and L. Aitchison. How to set adamw's weight decay as you scale model and dataset size. *arXiv preprint arXiv:2405.13698*, 2024.

S. Qiu, A. Potapczynski, M. A. Finzi, M. Goldblum, and A. G. Wilson. Compute better spent: Replacing dense layers with structured matrices. In *Forty-first International Conference on Machine Learning* icm [2024].

X. Wei, S. Moalla, R. Pascanu, and C. Gulcehre. Investigating low-rank training in transformer language models: Efficiency and scaling analysis. *arXiv preprint arXiv:2407.09835*, 2024.

International Conference on Learning Representations (ICLR'23), 2023. Published online: `iclr.cc`.

Proceedings of the International Conference on Learning Representations (ICLR'16), 2016. Published online: `iclr.cc`.

Forty-first International Conference on Machine Learning, 2024.

M. Ranzato, A. Beygelzimer, K. Nguyen, P. Liang, J. Vaughan, and Y. Dauphin, editors. *Proceedings of the 35th International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*, 2021.

International Conference on Learning Representations (ICLR'24), 2024. Published online: `iclr.cc`.

S. Zaidi, A. Zela, T. Elsken, C. Holmes, F. Hutter, and Y. Teh. Neural ensemble search for uncertainty estimation and dataset shift. In Ranzato et al. [2021], pages 7898–7911.

T. Wei, C. Wang, Y. Rui, and C. W. Chen. Network morphism. In M. Balcan and K. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning (ICML'17)*, volume 48, pages 564–572. Proceedings of Machine Learning Research, 2016b.

X. Deng, H. Shi, R. Huang, C. Li, H. Xu, J. Han, J. Kwok, S. Zhao, W. Zhang, and X. Liang. Growclip: Data-aware automatic model growing for large-scale contrastive language-image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22178–22189, 2023.

Y. Yao, Z. Zhang, J. Li, and Y. Wang. Masked structural growth for 2x faster language model pre-training. In *The Twelfth International Conference on Learning Representations* icl [2024]. URL `https://openreview.net/forum?id=rL7xsg1aRn`. Published online: `iclr.cc`.

W. Du, T. Luo, Z. Qiu, Z. Huang, Y. Shen, R. Cheng, Y. Guo, and J. Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. 2024.

S. Schug, S. Kobayashi, Y. Akram, J. Sacramento, and R. Pascanu. Attention as a hypernetwork. *arXiv preprint arXiv:2406.05816*, 2024.

A Background and Related Work

Scaled parameterizations refers to a set of rules that determine how certain parameters can be scaled with respect to one or more scaling dimensions [Everett et al., 2024]. Such prescriptions attempt to ensure stable feature learning under stable optimal hyperparameters to ensure maximal feature learning in the infinte-width limit [Yang and Hu, 2021]. *abc*-parameterizations [Yang and Hu, 2021] is a formulation that subsumes such parameterizations and broadly follows the assumption where model weights are so defined,

$$w_0 \sim \mathcal{N}(0, \boldsymbol{\mathcal{B}}_{\boldsymbol{w}}^{\mathbf{2}}), \tag{2}$$

$$W_t = \boldsymbol{\mathcal{A}}_{\boldsymbol{w}} \cdot w_t, \tag{3}$$

$$w_{t+1} = w_t + \boldsymbol{\mathcal{C}}_{\boldsymbol{w}} \cdot \Phi_t(\nabla L_0, \ldots, \nabla L_t), \tag{4}$$

where t defines the training step and $\Phi_t(\nabla L_0, \ldots, \nabla L_t)$ is the gradient-based weight update step [Blake et al., 2024]. $\mathcal{A}_w,\ \mathcal{B},\ \mathcal{C}_w$ are scalars that change with model width and determine the required scaling of parameterized entities. The specific prescription of these scalars is governed by the method and its assumptions which lead to the analytical rules of scaling. We refer the reader to Tables 1, 2, 3 from Yang et al. [2021] for a succinct summary of the scaling rules for the scalars $\mathcal{A}_w,\ \mathcal{B},\ \mathcal{C}_w$, which realizes μParameterization or μP. This class of methods is seeing growing interest in the community as model sizes increase and prior knowledge of good hyperparameters are not available on such scales [Yang et al., 2024, Everett et al., 2024, Blake et al., 2024, Thérien et al., 2024, Qiu et al., 2024, Lingle, 2024]. Recent studies find improvements in the μP recommendation and scenarios or setups where μP can be bettered. Our work is along these lines where we aim to show that *warmstarting* a larger model training from a smaller model is one more such scenario where vanilla-μP can be improved upon.

Shrink-and-perturb is a technique that showed that models of the same size can be warmstarted from a previous training run successfully such that it converges faster without hurting generalization [Ash et al., 2020]. Under *shrink-and-perturb*, every learnable parameter w_t^i is initialized as, $w_t^i \leftarrow \lambda\theta_{t-1}^i + p_t$, where $p_t \sim \mathcal{N}(0,\ \sigma^2)$ and $0 < \lambda < 1$. Ash et al. [2020] posits that the shrinking of the weights act as a regularization but not similar to weight decay. In effect, shrinking weights can act as increasing the entropy of the output distribution and preserving the relative activation at each layer. While perturbation seems to have an effect on balancing gradient contribution from each learned parameter. Crucially, this version of *shrink-and-perturb* was specifically shown for continual learning under stationary data distributions [Chebykin et al., 2023, Shin et al., 2024]. However, to our knowledge, no such *shrink-and-perturb* warmstarting method directly talks about warmstarting across model scales.

Model growth literature, or warmstarting literature across model sizes is rich and encompasses multiple methodological paradigms, including initialization techniques [Zaidi et al., 2021], network morphisms [Wei et al., 2016b], knowledge transfer and distillation approaches [Chen et al., 2016, 2021, Wang et al., 2023a], learned transformations [Deng et al., 2023, Wang et al., 2023b], and empirical heuristics [Rae et al., 2021]. More recent works show that increasing transformer model size can improve pretraining efficiency and potentially the compute-loss scaling coefficient [Yao et al., 2024, Du et al., 2024]. Each such method represents a well-designed and sophisticated approach that, in principle, achieves the required warmstarting effect. These approaches conflict with our requirement for a simple and practical implementation, require changes to training routines, loss functions, and usually keep the hyperparameters fixed across scales. Samragh et al. [2024], a concurrent work, echoes our requirements for a warmstarting method and proposes a *shrink-and-perturb*-like method for growing a language model, and is the closest work to us, altering only the initialization method given a smaller *base* model checkpoint. Our method and approach differ from all approaches mentioned above since we study and demonstrate that it is possible to warmstart *while* suitably scaling hyperparameters.

Our work aims to develop a theoretically motivated modular framework that can, given any model checkpoint with its tuned hyperparameters, systematically scale this model along arbitrary dimensions, initialize through principled warmstarting from smaller model weights, and train using theoretically derived hyperparameter scaling rules.

B Experiments

This section covers additional details on the experimental setup and more supporting results.

Model scales. Table 1 covers the model scales reported in the experiments in this work.

n_layers	d_model	n_head	head_size	# params [M]
6	48	2	24	5
6	96	6	16	10
6	192	8	24	22
6	288	12	24	35
6	512	16	32	70

Table 1: Overview of model scaling parameters with the number of layers fixed at 6 across all configurations. The d_model parameter is defined as the product of the embedding size and the number of attention heads. The *block size* is set to 1024. The effective batch size is as determined by the grid search on a base scale used for μTransfer.

Grid search results for *optimal* hyperparameters. For the chosen 3 base scales, we perform a grid search over the learning rate (LR) and batch size. The grid search results are summarized below:

- 5M: `learning rate` $= 0.03$; `batch size` $= 64$
- 10M: `learning rate` $= 0.01$; `batch size` $= 64$
- 22M: `learning rate` $= 0.003$; `batch size` $= 256$

When performing μTransfer, the batch size found for the base scale is kept constant while LR is scaled as per μP.

B.1 Additional results

Figure 4 shows Figure 1 when run for longer, i.e., trained with more tokens (here, 30 tokens/parameter). We observe that the feature learning under μTransfer is retained under our proposed *warmstarting* method. A similar converging loss compared to vanilla-μP may be the limit imposed on the loss by the model scale. Figure 5 adds more *base* scale transfer to our primary result, complementing Figure 1. The general trend of our *warmstarting* approach improving μP continues across all pairs of model scales we tried. However, as expected, the extent of gains provided by *warmstarting* varies depending on the model scales. For all of these results, we append the base model learning curves to the target model learning curves of our *warmstarting* approach in Figures 6. Although this shifts the learning curves, the impact is minimal, particularly when there is a significant difference in size between the base and target models.

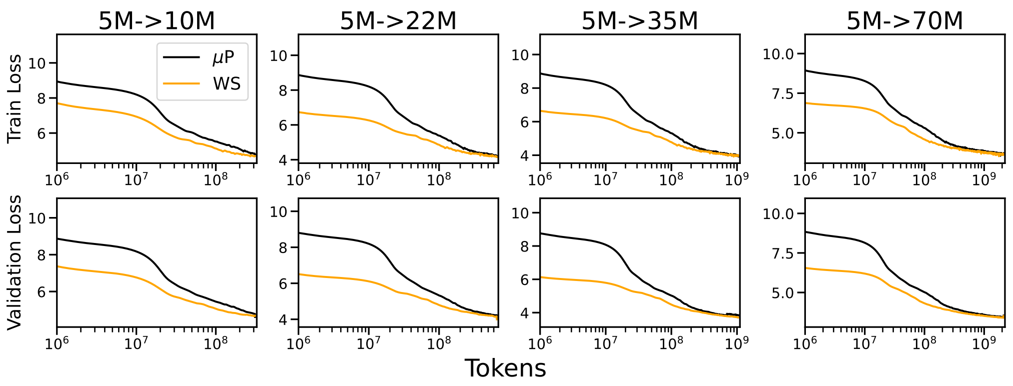

Figure 4: Models in Figure 1 trained for more tokens and thus compute. Here, we train each model for 30 tokens/parameter instead of the 20 recommended by Hoffmann et al. [2022].

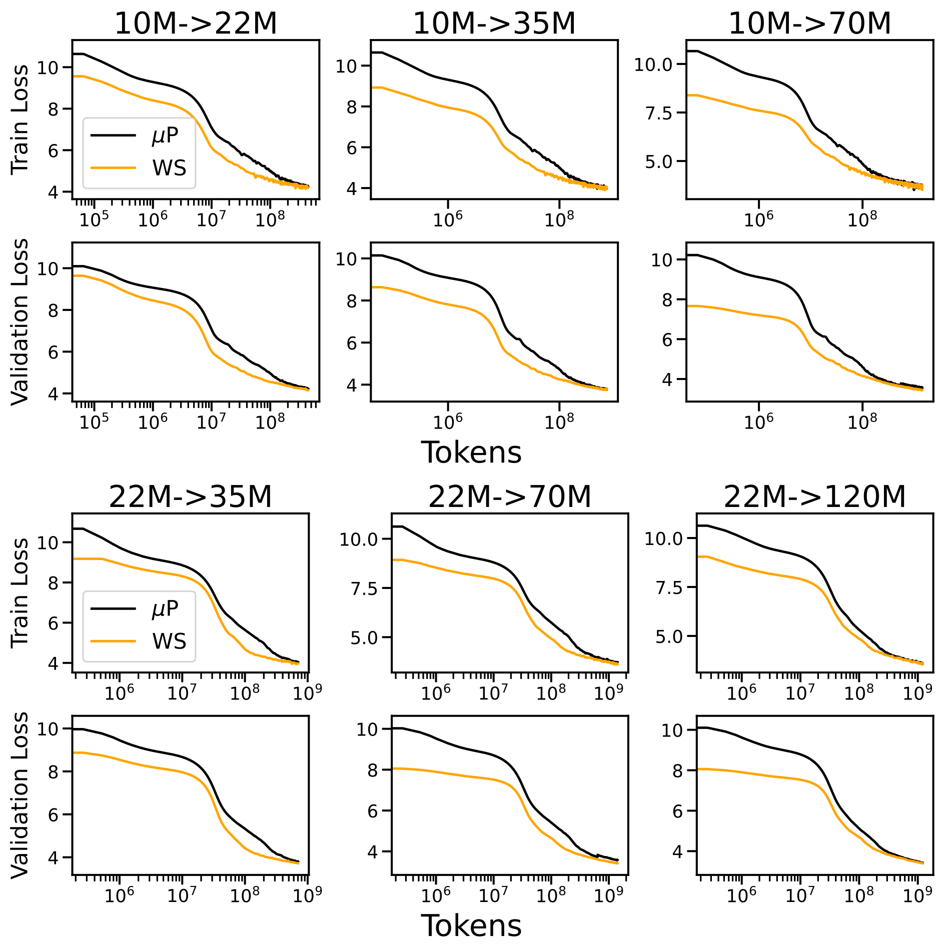

Figure 5: Transferring the best found learning rate at the base scale of 10M (*top set of* 2×3) and 22M (*bottom set of* 2×2) using μP. For *warmstarting* (WS) run, the model weights of the optimal *base* model is used to initialize the *target* model's training. Warmstarting improves μP convergence, though the quality speedup and gains depend heavily on the choice of base and target scales.

B.2 Analyzing training dynamics

When applying *warmstarting* with μP, it is important to also guarantee the optimality of the *base* scale hyperparameters when scaled and transferred to the larger *target* model. This beahviour manifests in metrics that evolve over training steps, such as the L1 norm of activations, L1/L2 norms of the weights, etc. Figures 7, 8, 9, 10 together highlight that the loss perspective alone is inadequate in knowing what will work best across scales. However, it appears that there is a *sweet-spot* for λ_{shrink} such that it is not too low ($= 0$ is equivalent to μP) or too high ($= 1$ is equivalent to not shrinking the *base* model). The value of 0.4 suggested in the literature appears to work the best in our setup.

B.3 Successive warmstarting

The entire goal of warmstarting a model from a smaller model is to speed up convergence of the larger model training. For pretraining LLMs, typically done in a sub-epoch manner by not repeating a data, we have a unique setup where if we want to train a model at a particular target scale, say 100M, for compute-optimality [Hoffmann et al., 2022], a warmstarted training run at 100M will consume *more tokens* than a standard run from scratch at 100M. This is illustrated in Figure 11 (left) where we see that for the same amount of compute (in FLOPs) allocated to a vanilla-μP and warmstarted-μP, the latter consumes more tokens.

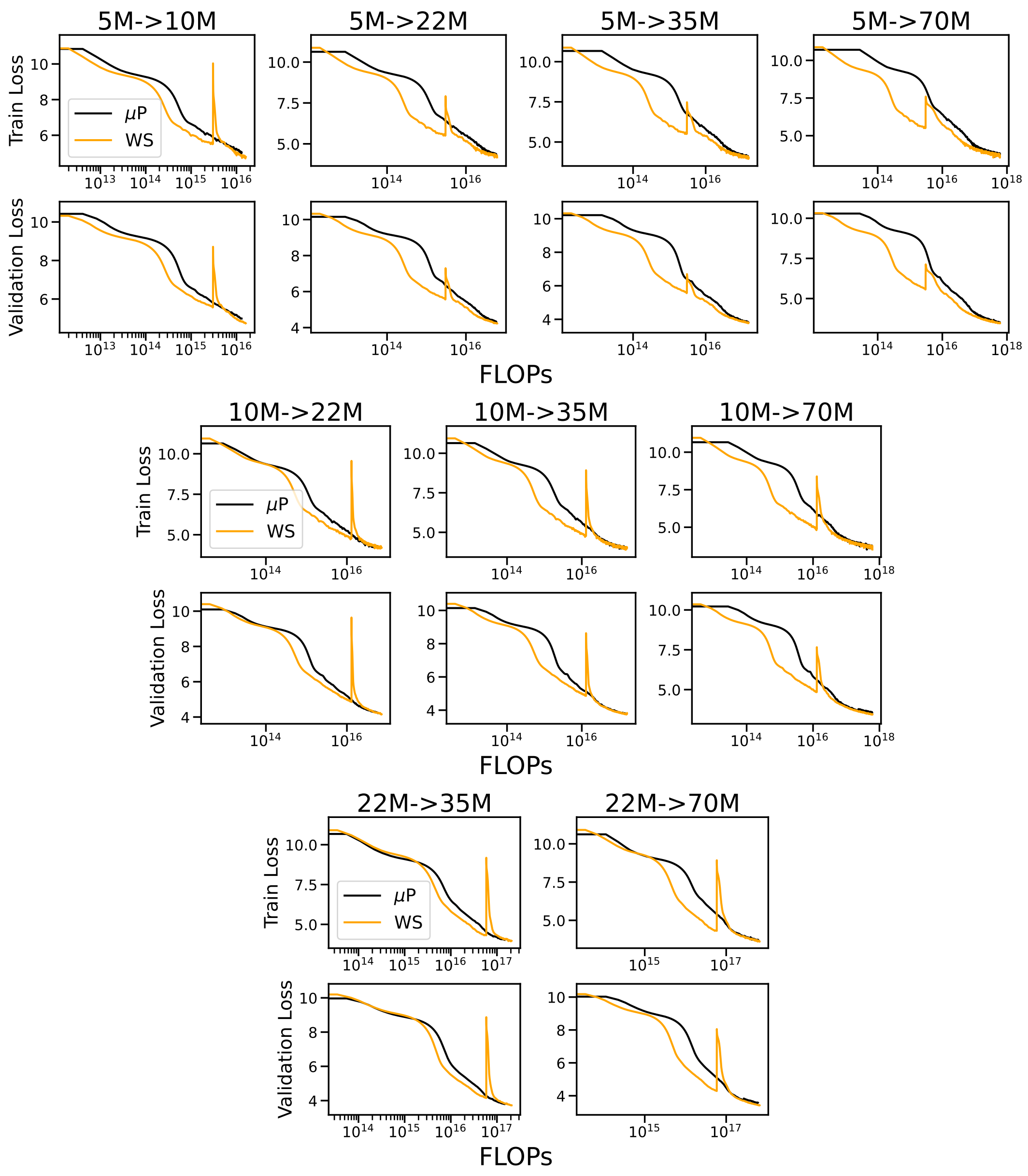

Figure 6: Warmstarting runs with the base model learning curve included. This can be seen with the position of spikes somewhat *delayed* the lower the difference between the target and base model scale is. For correct application of μP, in principle, the trained *base* model is also available given the tuning performed to find optimal hyperparameters at the *base* scale.

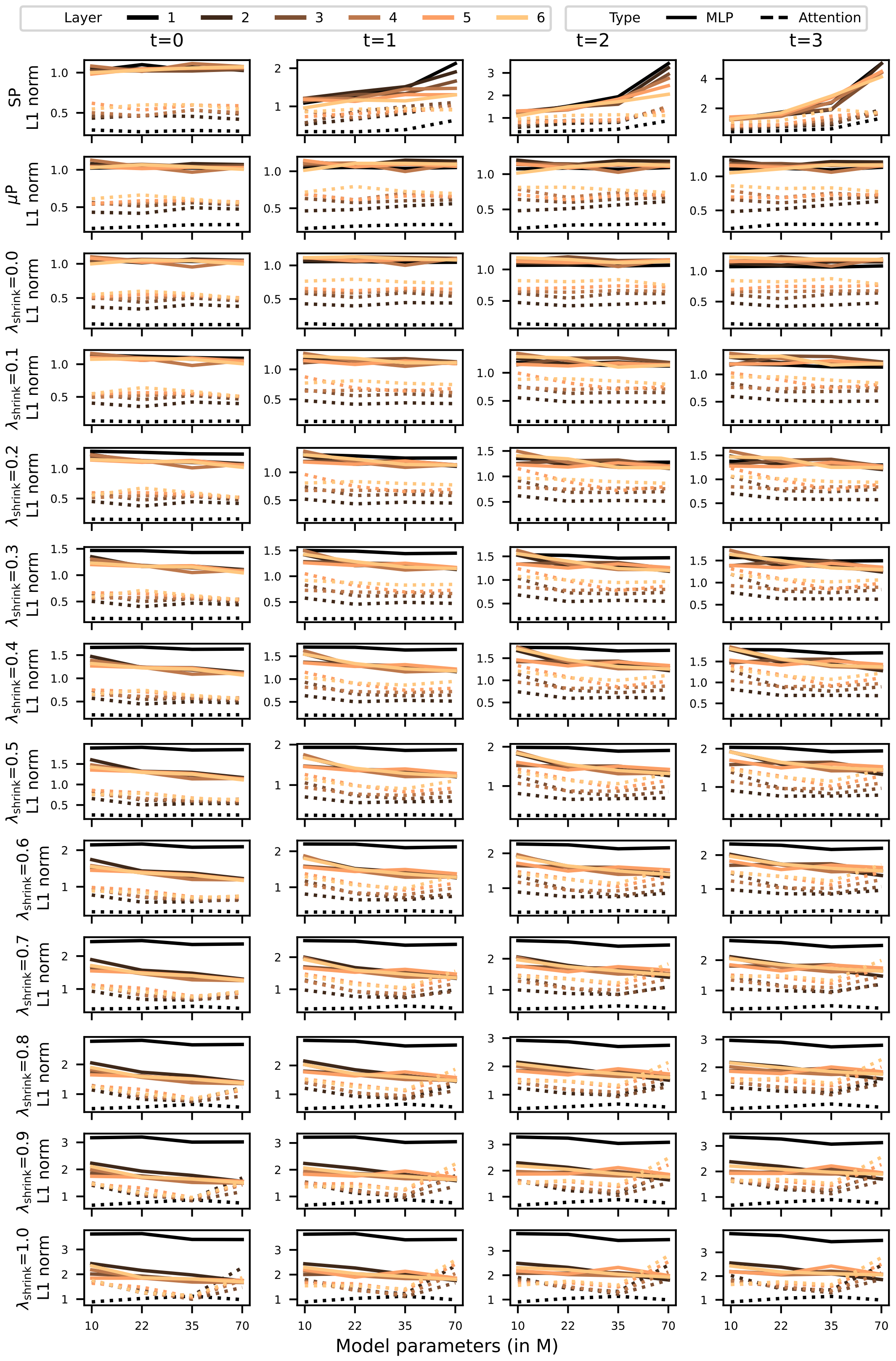

Figure 7: L1 norm of the layers activation across model scales (x-axis) for standard parametrization (SP), μP and WS with varying λ_{shrink} values (see, Equation 1). The lesser the shrinking (higher λ_{shrink}), the more does the instability grow with time, especially across individual layers.

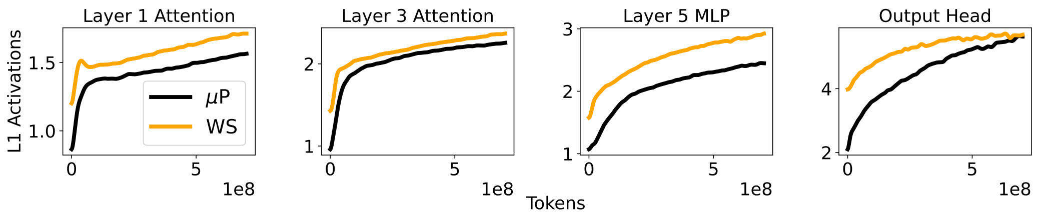

Figure 8: Aggregates and extends Figure 7, showing L1 norm of the layers activations over the entire training run. Importantly warmstarting, leads to stable training, closely matching μP trends.

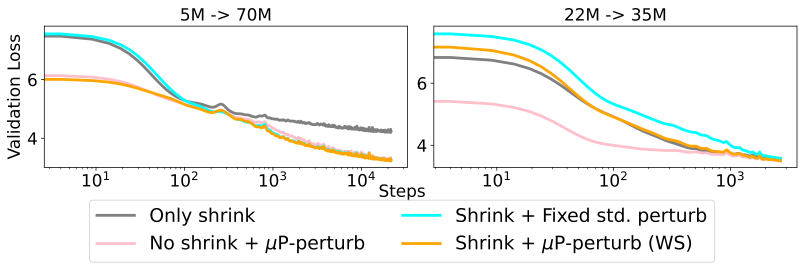

Figure 9: Comparison of different techniques of warmstarting around *shrink-and-perturb*, to highlight the role of individual components of shrinking the base model weights and the perturbation strength. The relative ranking of these methods vary significantly over the relative scale differences. See Figure 10 for the corresponding plot showing the role of the magnitude of shrink.

Taking this further such that for the same amount of total compute spent, we start from the base model and progressively scale a model in stages. Figure 11 (left) shows in *blue* how a 5M model was warmstarted to 22M and continued training, which was then again warmstarted to 70M, leading to a much higher consumption in tokens. As expected and observed (see, Figure 9) there is a spike when warmstarting and continuing a learning curve (see, Figure 11 (right)). However, the increase of model scale and our approach enables a quick drop in loss, enough for it to match μP's final performance.

Given that smaller models will tend to have higher loss gains per unit compute (FLOPs), this shows up in the *blue* curve generally being better[5] than the warmstarted-70M until its size is grown. Therefore, there is potential in staging warmstarting and continually growing models for improved loss-compute or tokens-compute efficiency. One flaw that Figure 11 also highlights is that despite having *spent more tokens* for the staged, successive *warmstarting* run, the extra tokens seen do not show up as improved loss. Either, the compute spent in recovering from the loss spike prevents from improving

[5]Note that until the blue spike, loss of a warmstarted-22M model is better than a warmstarted-70M model

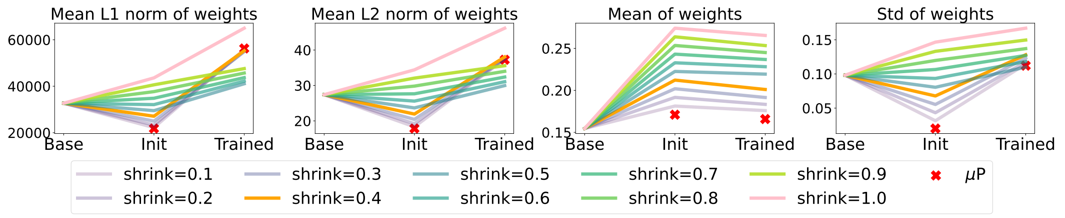

Figure 10: Compares the state of weights at different training times: (i) *Base* indicates the weights after completion of the full training budget (here, on scale 22M); (ii) *Init at Warmstart* indicates the weights post-warmstarting, before beginning training (here, for scale 35M); (iii) *Trained* indicates the state of the warmstarted weights after expending the corresponding training budget. Any higher shrinking factor (> 0.4) interestingly leads to divergence of the final trained L1/L2 norms of the weights, especially compared to μP. Note, *shrink*$= 1.0$ corresponds to no-shrinking of weights before applying warmed-μP.

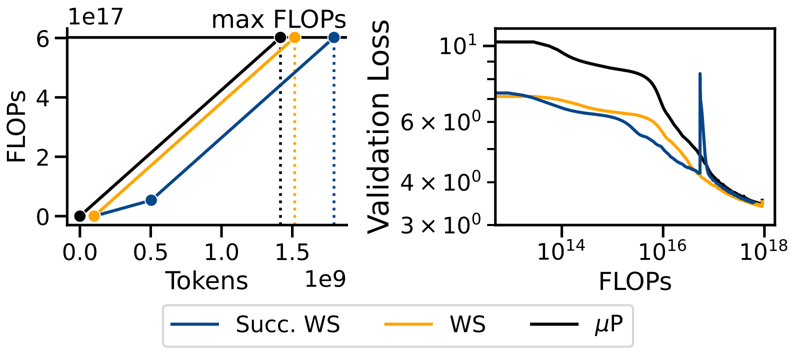

Figure 11: *WS*: our approach of warmstarting-μP from 5M$\rightarrow$70M; *Succ. WS*: warmstarting from 5M$\rightarrow$22M, and then from 22M$\rightarrow$70M, with 5M as the base scale for μTransfer under both scale increase; (*Left*): Under same compute allocation for training a 70M model, using μP, we see that *warmstarting* leads to the consumption of more tokens as we do not repeat data when *warmstarting*, and naturally successively *warmstarting* further leads to more token usage; (*Right*): The learning curves for the same runs, under the same compute budget as expected for a 70M model. Despite a spike in loss (*blue*) for successive warmstarting, it is able to achieve *similar* loss as the vanilla-μP run and the warmstarted-70M. Even though the 70M model in the successive warmstarting run sees much lesser compute.

the loss under total available equivalent compute. Or, the warmstarting method is suboptimal and is under-utilizing the tokens seen overall across *base* scale training and *warmstarting*.

We believe this is a promising direction and together with an improved *warmstarting* technique will lead to much more practical speed ups and offer avenues for improved hyperparameter tuning strategies. Moreover, studying such setups through the lens of compositional generalization [Schug et al., 2024] and *abc*-parameterization [Everett et al., 2024] will offer novel and practical insights and is a certain future direction to pursue.