On Your Mark, Get Set, Warmup!

Anonymous Author(s)
Affiliation
Address
email

Abstract

It is common in deep learning to warm up the learning rate η, often by a linear schedule between $\eta_{\text{init}} = 0$ and a predetermined target η_{trgt}. In this paper, we show through systematic experiments using SGD and Adam that the overwhelming benefit of warmup arises from allowing the network to tolerate larger η_{trgt} by forcing the network to more well-conditioned areas of the loss landscape. The ability to handle larger η_{trgt} makes hyperparameter tuning more robust while improving the final performance. We uncover different regimes of operation during the warmup period, depending on whether training starts off in a progressive sharpening or sharpness reduction phase, which in turn depends on the initialization and parameterization. We also suggest an initialization for the variance in Adam which provides benefits similar to warmup.

1 Introduction

One of the most important choices to make in gradient-based optimization is the learning rate (step size) η. If η is too small, then learning may take place too slowly or the model might get stuck in unfavorable regions of the loss landscape. If η is too large, training will typically diverge. In practice, it is common to pick a dynamical learning rate schedule η_t [2, 4, 39, 26]. Modern learning rate schedules for deep learning typically consist of a warmup period where η_t is increased linearly from zero to a target value η_{trgt} over a warmup time T_{wrm} [13, 33]. After the warmup period, it is common to eventually decay the learning rate, for example via a cosine decay schedule [33, 26, 39].

Given that warmup is standard in the practitioner's toolkit, it is important to understand it deeply and identify improvements. In modern settings, perhaps the earliest work to use warmup was [14], which used a small constant learning rate for the first few epochs of training and then switched to a larger learning rate. A linear warmup schedule was later introduced in [13]. The intuition given was that to scale the minibatch size in SGD by a factor of k, it is natural to also scale the learning rate by a factor of k, provided the model is not changing too rapidly and successive gradients are roughly aligned. However at the beginning of training, the model is changing rapidly, so it is natural to start with a lower learning rate and gradually increase it to the target value after the network has stabilized. Other explanations suggest that since the network is initialized randomly, the gradient steps at the beginning of training are not meaningful, and thus it would be harmful to take large steps in such directions [39], so it makes sense to take smaller steps early in training. The analysis by [12] suggests that warmup primarily limits the magnitude of weight updates in the deeper layers, preventing large instabilities. It has also been suggested that the key benefit of warmup arises for adaptive optimizers, such as Adam: [23] argues that the variance of the adaptive learning rate is large during early training because the network has seen too few training samples; it is asserted that this large variance is harmful, and that warmup acts as a variance reduction method by allowing the network to collect accurate statistics of the gradient moments before using larger learning rates. Alternatively, it is also sometimes stated that the initialization may start the model off at places in parameter space that are unstable, difficult to optimize, and easily lead to divergence, and that warmup can help alleviate this [39].

Submitted to the Mathematics of Modern Machine Learning Workshop at NeurIPS 2024. Do not distribute.

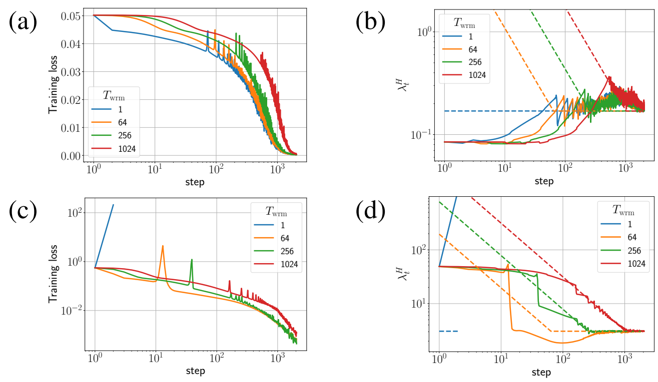


Figure 1: Training loss and sharpness trajectories of FCNs trained on a 5k subset of CIFAR-10 with MSE loss using GD. The dashed lines in the sharpness figures illustrate the instability thresholds $2/\eta_t$. (top) μP with $\eta_{\text{trgt}} = 1/\lambda_0^H$, (bottom) SP with $\eta_{\text{trgt}} = 32/\lambda_0^H$. Similar mechanisms are observed across different architectures, loss functions, and mini-batch sizes, as shown in Appendix E.

The above explanations are varied and do not clearly demonstrate why and to what extent warmup is necessary. A loss landscape perspective was given in [10] (and summarized in [26] Ch. 8), which argued that an important effect of warmup is to gradually reduce the sharpness (the top eigenvalue of the Hessian of the loss), thus causing the model to leave poorly conditioned areas of the loss landscape and move towards flatter regions which can tolerate larger learning rates. They argue that the mechanism for this is similar to the dynamical stability (catapult) mechanisms studied in [34, 22].

Our contributions. In this paper, we perform extensive studies on the effect of learning rate warmup across a variety of architectures, initializations and parameterizations, datasets, and for both SGD and Adam optimizers. We demonstrate through systematic experiments that by far the primary benefit of learning rate warmup is to allow the network to tolerate larger learning rates than it otherwise would have. This builds on the observations of [10] by showing that any other benefits are marginal, disentangling the effect of warmup duration and target learning rate, and by extending the empirical evidence to include adaptive optimizers and Transformers.

2 Notations and Preliminaries

Sharpness: The sharpness is defined as the maximum eigenvalue of the Hessian of the loss $\lambda_t^H := \lambda_{\max}(\nabla_\theta^2 L)$ at training step t. For adaptive optimizers with pre-conditioner P, $\lambda^{P^{-1}H} := \lambda_{\max}(P^{-1}\nabla_\theta^2 L)$ denotes the pre-conditioned sharpness. For details on Adam's pre-conditioner, see Appendix D.3.

Linear Warmup: This is defined by the schedule $\eta_t = \eta_{\text{init}} + (\eta_{\text{trgt}} - \eta_{\text{init}}) \left(\frac{t}{T_{\text{wrm}}}\right)$. Unless otherwise specified, we set $\eta_{\text{init}} = 0$ when referring to linear warmup.

Parameterizations in Neural Networks: The mechanism of warmup and its effectiveness is heavily influenced by the network parameterization (see Sections 3 and 4). Standard Parameterization (SP) [32] is a staple in common libraries [27, 3]. Another notable parameterization is the Neural Tangent Parameterization (NTP) [17], which along with SP resides in the kernel learning class at infinite width. Ref. [36] proposed Maximal Update Parameterization (μP) which exhibits feature learning at infinite width. Neural network parameterizations significantly impact training dynamics [19].

3 Warmup Mechanisms of Gradient and Adaptive Methods

This section analyzes the underlying mechanism of warmup through the lens of training instability. A key finding is a dichotomy between cooperative versus competitive dynamics based on how the natural evolution of the sharpness interplays with the training instability.

3.1 Stochastic Gradient Descent (SGD)

Learning rate warmup is intrinsically tied to sharpness dynamics, as sharpness determines the instability threshold η_c. As the learning rate is increased during warmup, these instabilities induce a temporary increase in the loss and a decrease in the sharpness to restore stability through the self-stabilization mechanism. Ultimately this allows the model to adapt to the increased learning rate. In other words, the primary goal of warmup is to gradually reduce sharpness, guiding training towards flatter regions that can accommodate training at higher learning rates [10].

However, digging deeper, we find that training has a 'natural' preference for sharpness evolution throughout the training course [19]. Before exceeding the instability threshold $(\eta < \eta_c)$, training naturally experiences either a progressive increase or decrease in sharpness, as observed in Figure 1, which is unrelated to warmup. Here, the natural sharpness evolution can be defined as the change in sharpness experienced by gradient flow. The interplay between this natural sharpness evolution and the deliberate intervention of warmup to reduce sharpness can result in completely distinct dynamics. Below, we detail these cases and describe the conditions that typically exhibit them.

(C1) Natural Progressive Sharpening (top row of Figure 1): The combined effect of the network naturally increasing sharpness while the learning rate is also being increased results in a "head-on collision" at which the network reaches the instability threshold η_c. This causes the loss to increase, leading to a decrease in sharpness and facilitating a return to stability. As training proceeds, both sharpness and learning rate continue to increase, again surpassing the instability threshold. This results in a *persistent catapult cycle*, characterized by $\eta_t \approx 2/\lambda_t^H \approx \eta_c$, for the remainder of the warmup period, as seen in Figure 1(b).

(C2) Natural Sharpness Reduction (bottom row of Figure 1): The network is naturally already reducing its sharpness during early training. However, if the learning rate is increased sufficiently quickly, eventually the instability threshold will be reached (akin to a "rear-end collision"), causing the loss to increase. For small enough learning rates, the increased loss induces a dramatically more pronounced decrease in sharpness than would naturally occur, ultimately restoring stability. To exceed the instability threshold again, the learning rate must significantly increase to account for the decreased sharpness, potentially requiring considerable training steps. Consequently, training experiences one or more separated catapults during the warmup phase, as seen in Figure 1(c, d). This contrasts with the progressive sharpening case, where training enters a continuous catapult cycle after reaching the instability threshold for the first time. Notably, training may eventually reach a very flat region of the landscape during warmup, with gradients pointing towards increasing sharpness (e.g., $T_{\text{wrm}} = 64$ in Figure 1(d)). Upon reaching such a region, the dynamics aligns with the natural progressive sharpening scenario.

3.1.1 A Toy Model for Understanding the Warmup Mechanisms

These two scenarios can be interpreted as cooperative or competitive dynamics between warmup and the natural evolution of sharpness. When training inherently undergoes sharpness reduction, it cooperates with warmup in decreasing sharpness. Conversely, if the natural trajectory of training is towards increasing sharpness, it opposes the warmup's effort, leading to a persistent cycle of catapults. We can understand these mechanisms by analyzing a model of self-stabilization derived by [8].

The model assumes that the top eigenvector $\boldsymbol{u}$ changes slowly through training and can be treated as constant. Next, consider a cubic approximation of the dynamics along a reference point $\boldsymbol{\theta}^*$. The dynamics along the projection $x_t := \boldsymbol{u}^T(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*)$ is given by two coupled non-linear equations:

$$x_{t+1} = (1 - \eta_t \lambda_t^H) x_t, \qquad \lambda_{t+1}^H = \lambda_t^H + \eta_t(\alpha - \beta x_t^2),$$

where $\alpha := -\nabla\lambda^H \cdot \nabla L(\theta)$ quantifies the instantaneous change in sharpness and $\beta := \|\nabla\lambda^H\|^2$ controls to the non-linear change in sharpness. Ref. [8] considered a constant learning rate η and assumed progressive sharpening ($\alpha > 0$). Here, in contrast, we consider a time-dependent learning rate and allow α to attain both positive and negative values. In this model, an instability arises when $\eta_t \lambda_t^H > 2$. During instability, x_t continues to increase until the higher order term in the sharpness update equation causes a significant decrease in sharpness. Once the sharpness has decreased sufficiently, the stability is restored ($\eta_t \lambda_t^2 < 2$), and training continues.

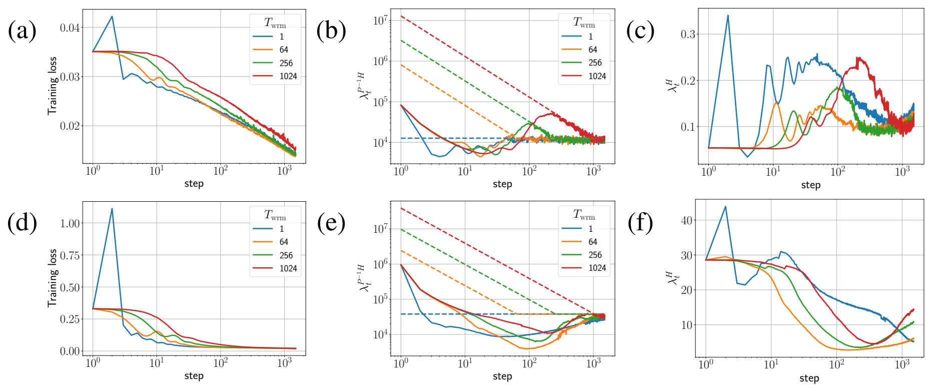


Figure 2: Training loss and sharpness trajectories of FCNs trained on the entire CIFAR-10 dataset with MSE loss using full batch Adam. (top) simple-μP (for details, see Appendix D.2.1) with $\eta_{\text{trgt}} = 0.003$ and (bottom) SP with learning rate $\eta_{\text{trgt}} = 0.001$. The dashed lines in the sharpness figures illustrate the instability thresholds ${(2+2\beta_1)}/{\eta_t(1-\beta_1)}$. Similar mechanisms are observed for different architectures, loss functions, and smaller batch sizes as detailed in Appendix E.

Next, we consider the two natural sharpness evolution scenarios:

(C1) Natural Progressive Sharpening ($\alpha > 0$)**:** The combined effect of naturally increasing sharpness ($\alpha > 0$) and the increasing learning rate from warmup leads to instability ($\eta_t \lambda_t^H > 2$). Resultantly, x_t increases until the higher order term in the sharpness update cause a decrease in sharpness ($x_t^2 > \frac{\alpha}{\beta}$). Once the sharpness has decreased appreciably so that $\eta_t \lambda_t^H < 2$, stability is restored and the training continues. As training proceeds, both progressive sharpening and increasing learning rate cause instability, resulting in a persistent catapult cycle characterized by $\eta_t \lambda_t^H \approx 2$.

(C2) Natural Sharpness Reduction ($\alpha < 0$)**:** In this case, sharpness is naturally decreasing during training ($\alpha < 0$). If the learning rate is increased quick enough relative to decreasing sharpness, an instability occurs ($\eta_t \lambda_t^H > 2$). The increase in x_t causes a more pronounced decrease in sharpness than it would have occurred naturally, restoring instability. To exceed the instability threshold again, the learning rate must significantly increase to account for the decreased sharpness. This results in one or more separated catapults.

3.1.2 The Effect of Warmup Duration

Given a fixed target learning rate η_{trgt}, increasing the warmup duration T_{wrm} delays the point at which training exceeds the instability threshold η_c, allowing the sharpness to evolve freely before reaching this point. In the sharpness reduction case, sharpness can significantly decrease by the time this threshold is reached, lowering the need for warmup to decrease sharpness actively. Consequently, increasing T_{wrm} results in catapults that are both delayed and smaller in magnitude, as seen in Figure 1(d). As the catapults become less intense on increasing the warmup duration, the model can train at higher learning rates without diverging, pushing the divergence boundary. For extended warmup durations, warmup may not actively reduce sharpness in these sharpness reduction cases and instead "piggy-backs" on the inherent sharpness decrease.

In the progressive sharpening case, increasing T_{wrm} allows the sharpness to naturally increase. As a result, training exceeds the instability threshold for the first time at a relatively lower learning rate compared to the constant learning rate case. Although warmup has to now undertake more work in decreasing sharpness, it does so in a more gradual manner since increasing the warmup duration amounts to a lower warmup rate ${\eta_{\text{trgt}}}/{T_{\text{wrm}}}$. As a result, the fluctuations observed on exceeding the instability threshold are much smaller in magnitude, as seen in Figure 1(a, b).

3.1.3 Small vs. Large Initializations

So far, we have outlined different warmup mechanisms without describing specific conditions that typically exhibit them. Small initializations, such as those using maximal update parameterization (μP) [36] or appropriately using normalizing layers (e.g. standard Transformer architectures, see Figure 14 in Appendix E.5), are characterized by a small initial network output. Such initializations start in flat regions where gradients point toward increasing sharpness [19], placing them in the

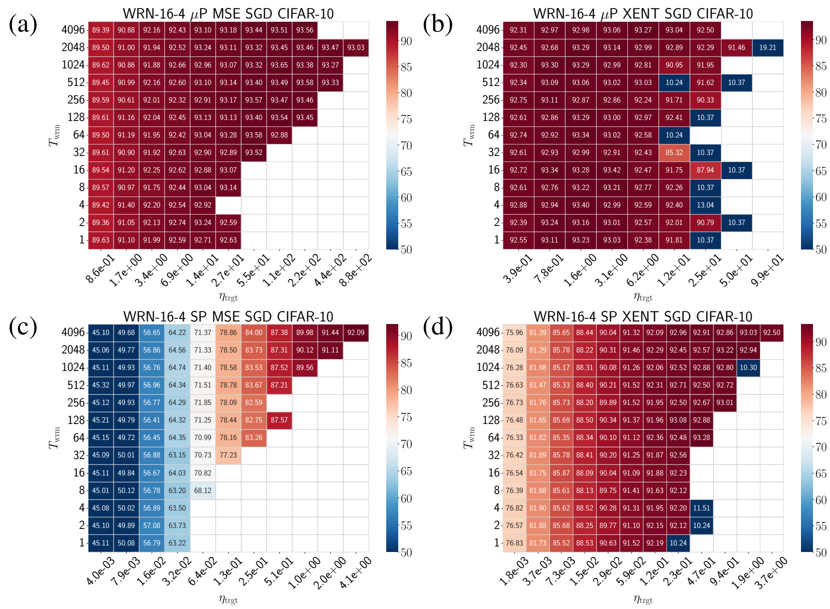


Figure 3: Test accuracy heatmaps of WideResNets (WRNs) trained on CIFAR-10 using different parameterizations and loss functions using SGD: (a) μP and MSE loss, (b) μP and cross-entropy loss, (c) SP and MSE loss, and (d) SP and cross-entropy loss. Empty cells correspond to training divergences. Additional results are provided in Appendix F.

progressive sharpening category (C1). As we will see in Section 4, such initializations may not significantly benefit from warmup as they already start in a flat region. In contrast, large initializations, such as FCNS, CNNs, ResNets with Standard Parameterization (SP) initialized at criticality [28, 30] or Transformers with the last layer-norm removed, undergo an early sharpness reduction, categorizing them into sharpness reduction category (C2). As the primary effect of warmup is to reduce sharpness, we expect such large initializations to considerably benefit from warmup. Notably, large initializations can eventually undergo progressive sharpening at later training stages [18, 19] and adhere to the second mechanism, especially for prolonged warmups. Instances of constant sharpness (C3) typically arise in models operating near the lazy regime [5], such as wide networks in NTP or SP.

SGD with momentum: The warmup mechanism of SGD with momentum, while at its core is similar to that of vanilla SGD, has a few subtleties. We discuss it in detail in Appendix E.2.

3.2 Adaptive Gradient Methods (Adam)

Figure 2 shows the training loss, pre-conditioned sharpness, and sharpness trajectories for full batch Adam. These results suggest that the local stability of adaptive optimizers is determined by the largest eigenvalue of the pre-conditioned Hessian, denoted by $\lambda^{P^{-1}H}$, rather than the sharpness itself (also, see Ref. [7] for late time instability). In these figures, sharpness is significantly smaller than its instability threshold ${}^{(2+2\beta_1)}/{}_{\eta_t} \approx 4000$, indicating that sharpness does not determine stability. Instead, loss catapults are associated with $\lambda^{P^{-1}H}$ exceeding its corresponding instability threshold.

The pre-conditioned sharpness starts high for both progressive sharpening (simple-μP) and sharpness reduction (SP) scenarios considered in the previous section. For simplicity, we considered a simpler version of μP, detailed in Appendix D.2.1. In particular, for μP models, $\lambda_0^{P^{-1}H} \sim 10^5$ despite being initialized in a flat region as measured by sharpness, while for SP models, $\lambda_0^{P^{-1}H} \sim 10^6$. These large initial values of $\lambda_0^{P^{-1}H}$ can lead to training failures. We put forward strategies to improve Adam's initialization in Section 5; here we continue characterizing the warmup mechanisms of Adam.

Given that the pre-conditioned sharpness consistently starts high and decreases during early training, this behavior can be viewed as an extreme example of the natural sharpness reduction scenario (C2) described in the previous section. Training Adam at high initial learning rates without warmup can cause large catapults, as seen in Figure 2(d), potentially leading to training failures. Increasing the warmup duration allows the pre-conditioned sharpness to naturally decrease. This prevents the loss from spiking during early training and avoids training failures. In the later stages of training, the pre-conditioned sharpness may continue reducing or exhibit progressive sharpening. From here on, the dynamics follows the warmup mechanisms discussed in the previous sections, with sharpness replaced with pre-conditioned sharpness. Similar to the momentum case, Adam's stability threshold at late training times significantly decreases for smaller batch sizes [7], also shown in Appendix E.4.

4 Impact of Warmup on Training and Generalization

Here we investigate the impact of warmup on training and generalization by disentangling the role of η_{trgt} and T_{wrm}. Our key findings are that generalization capability is primarily determined by η_{trgt} and that Adam is particularly sensitive to large learning rates (specifically, large catapults). The role of increasing T_{wrm} is to (i) allow the network to tolerate larger η_{trgt}, and (ii) move the network further away from the divergence (failure) boundary, leading to a marginal improvement in generalization. For experimental details, see Appendix D.

4.1 Stochastic Gradient Descent (SGD)

Figure 3 presents heatmaps that show the best test accuracy achieved during training, plotted in the η_{trgt}-T_{wrm} plane for different parameterizations and loss functions. These phase diagrams of warmup also show the convergence-divergence boundary, with empty cells indicating training divergences, illustrating the interplay between warmup duration and the maximum trainable η_{trgt}. Below, we discuss the crucial insights these results provide into warmup's role in training dynamics.

Longer Warmup Facilitates Training at Higher Learning Rates: These phase diagrams reveal that an extended warmup duration facilitates training at higher target learning rates. This benefit is particularly noticeable for large initializations (like SP) and MSE loss. In contrast, the advantage is less pronounced when using cross-entropy loss and smaller initializations (like μP). The diminished benefit for μP is likely due to its initialization in a relatively flat region of the loss landscape, which can already facilitate training at higher learning rates at initialization. This consistent increase in maximum η_{trgt} with warmup durations can be understood through the lens of warmup mechanisms described in the previous section. As observed in Figure 1, when the warmup duration is increased, loss catapults occurring on surpassing the instability thresholds become milder. This effectively pushes the divergent boundary to higher learning rates.

Final Performance Primarily Depends on the Target Learning Rate: A closer look into these phase diagrams reveals that, slightly away from the divergent boundary, the test accuracy primarily depends on the target learning rate and nominally on the warmup duration. Based on the model performance, we can categorize these phase diagrams into two distinct cases: (i) models that fail to achieve optimal performance when trained with a constant learning rate (e.g., Figure 3(c)), and (ii) models that attain optimal performance without warmup (e.g., Figure 3(b)). The first scenario corresponds to models with large initializations. Increasing the warmup duration improves performance by facilitating training at higher learning rates. Yet, similar performance is observed for different warmup durations, suggesting that the primary gain comes from the target learning rate, rather than the duration itself. The second case arises for flat initializations, which can already train at large learning rates, and resultantly the optimal performance is already achieved without warmup. While increasing warmup duration facilitates training at even higher learning rates, it does not enhance performance. Nevertheless, it does broaden the range of optimal learning rates, reducing the need for precise tuning of the target learning rate, and making training more practical and robust. We conclude that warmup can serve two key purposes: (i) it can significantly improve model performance in large initialization cases, and (ii) extend the range of optimal target learning rates for small initializations, making it easier to tune the target learning rate. In Appendix F.2, we demonstrate that these results hold on incorporating momentum and employing cosine learning rate decay.

4.2 Adam

The warmup phase diagrams for Adam, as shown in Figure 4(a), exhibit characteristics similar to the sharpness reduction case of SGD, with notable differences. Increasing the warmup duration enables training at higher learning rates by allowing the pre-conditioned sharpness to decrease naturally, thereby reducing the severity of catapults. These large catapults, which may persist in Adam's

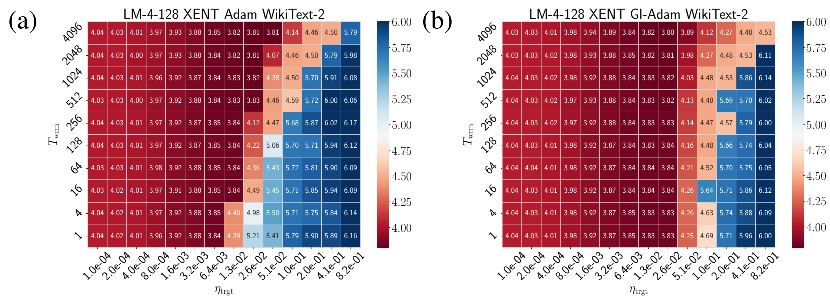


Figure 4: Test loss heatmaps of Pre-LN Transformers in SP trained on WikiText-2 with cross-entropy loss using (a) Adam, and (b) GI-Adam (introduced in Section 5).

memory, can lead to performance degradation and training failures. Thus, in addition to facilitating training at higher rates similar to SGD, warmup further improves Adam's performance by addressing its vulnerability to large catapults, justifying its widespread use with Adam. Below, we discuss the distinct properties of Adam phase diagrams in detail.

Training Failures of Adam: Remarkably, we find that models trained with Adam always exhibit training failures rather than divergences where the loss grows without bound, as further demonstrated in Appendix G. In cases of training failure, we often observed that certain layers or residual blocks output zero, leading to vanishing gradients. This implies that the model gets stuck at a critical point and is unable to train further. Understanding this unexpected phenomenon requires further study, which we leave to future work.

Performance Degradation prior to Failure Boundary: Test accuracy in these phase diagrams declines well before the failure boundary, in stark contrast to SGD where optimal learning rates are observed near the divergence boundary. This discrepancy stems from Adam's property of retaining a memory of gradient magnitudes. At large learning rates, along with the loss, the gradients spike during early training, as seen in Figure 23 in Appendix G. While the gradients decrease after a few training steps, the second moment of gradients $\boldsymbol{v}$ remains large for an extended period, leading to a small effective learning rate ηP^{-1}. As a result, training struggles to escape high-loss regions. Therefore, a longer warmup is more beneficial for Adam compared to SGD, as it is crucial to stay away from the failure boundary.

5 GI-Adam: Improving Adam's Initialization

In Section 3.2, we observed that the pre-conditioned sharpness for Adam starts at a high value, even for low sharpness initializations like μP, and can lead to training failures at large learning rates. We propose Gradient Initialized Adam (GI-Adam), which initializes the second moment using the gradient squared, $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$. In Appendix H.2, we show that a bias correction is not required when the second moment is initialized using the gradients. As a result, GI-Adam can be viewed as standard Adam with an automated warmup given by $\eta_t = \eta_{\text{trgt}} \sqrt{1 - \beta_2^t}$.

This simple trick reduces the initial pre-conditioned sharpness by around two orders of magnitude (more precisely by a factor of $\sqrt{1-\beta_2}$) at initialization, preventing large catapults, as illustrated in Figure 25 of Appendix H.1 (c.f. Figure 2(d-f)). Moreover, it consistently shows improvement over standard Adam across datasets and prevents training failures by pushing the training failure boundary to higher η_{trgt}, as shown in Figure 4(b). We provide additional results for different datasets in Appendix F.3. To further assess that the primary cause of instability during early training is the large pre-conditioned sharpness, we randomly initialize $\boldsymbol{v}_0$ but with the same norm as the gradients at initialization. Like GI-Adam, this also results in improved performance as shown in Appendix H.3.

6 Discussion

Our analysis provides new insights into the role of warmup across optimizers and parameterizations. We found compelling evidence that the primary effect of warmup is to facilitate training at higher learning rates and stabilizing the training dynamics by keeping it away from the failure (divergence) boundary. Looking under the hood, we found a variety of underlying mechanisms, which also suggested several improvements for hyperparameter initialization. In Appendix A we provide practical guidance for practitioners on choosing the warmup duration.

References

[1] Tiny imagenet challenge. `https://cs231n.stanford.edu/reports/2017/pdfs/930.pdf`.

[2] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[4] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[5] Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[6] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[7] Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability, 2022.

[8] Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] Emily Dinan, Sho Yaida, and Susan Zhang. Effective theory of transformers at initialization, 2023.

[10] 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.

[11] Gabriel Goh. Why momentum really works. *Distill*, 2017.

[12] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *International Conference on Learning Representations*, 2019.

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[14] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020.

[16] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[17] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[18] Dayal Singh Kalra and Maissam Barkeshli. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[19] Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. Universal sharpness dynamics in neural network training: Fixed point analysis, edge of stability, and route to chaos. *arXiv preprint arXiv:2311.02076*, 2023.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[23] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[24] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[25] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[26] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[28] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NIPS*, 2016.

[29] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[30] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory*. Cambridge University Press, 2022. `https://deeplearningtheory.com`.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[32] Jascha Narain Sohl-Dickstein, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *ArXiv*, abs/2001.07301, 2020.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[34] Lei Wu, Chao Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31, 2018.

[35] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR, 13–18 Jul 2020.

[36] 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.

[37] 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.

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. 2017.

[39] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. *Dive into deep learning*. Cambridge University Press, 2023.

[40] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

A Practical Guidance for Practitioners

How to Select the Warmup Duration? Given a target learning rate η_{trgt}, if the training loss during the warmup period exhibits large instabilities (loss spikes), the warmup duration T_{wrm} should be increased until such instabilities are sufficiently small. This effectively moves training away from the divergent / failure boundary, as illustrated in Figure 3. This is particularly crucial for Adam, as large instabilities can be detrimental and lead to considerable performance degradation without divergence, as discussed in Section 4.2.

How to Select the Target Learning Rate? As the primary effect of warmup is to anneal sharpness by increasing the learning rate beyond the instability threshold, it suggests that the target learning rate should be at least greater than the instability threshold at initialization.

When to Decay the Learning Rate? Figure 18 suggests that employing learning rate decay at small learning rates can result in performance degradation for a fixed training budget. Therefore, the learning rate should be decayed at large target learning rates only. The underlying intuition is that we use large target learning rates to train in a flat region of the landscape. However, these large learning rates restrict training to go into sharper regions of the basin and learning rate decay helps.

Leveraging μP for Effecient Training: Our analysis suggests that the primary role of warmup facilitates training at higher learning rates by gradually reducing sharpness. Given this perspective, beginning training with flat initializations, such as μP, is advantageous. These initializations might allow for achieving optimal performance without the need for warmup, as observed in Figure 3.

B Overview of Training Instabilities and the Self-Stabilization Mechanism

The underlying mechanism of warmup is intimately tied to training instabilities. These training instabilities, often referred to as 'catapults' [22, 6], arise when the learning rate η exceeds a critical threshold η_c, where both η and η_c generally change with time. When the instability threshold is exceeded ($\eta > \eta_c$), two cases arise: (i) if the learning rate is higher than the instability threshold but smaller than a maximum stable learning rate (which varies with time), i.e., $\eta_c < \eta < \eta_{\text{max}}$, training stabilizes through a self-stabilization process and training continues, (ii) if the learning rate exceeds this maximum stable learning rate $\eta > \eta_{\text{max}}$, training experiences severe instabilities. For SGD, these can result in training divergence, characterized by the loss increasing to infinity, whereas for Adam, training may cease, resulting in a training failure, where the loss fails to improve significantly over its initial value.

For vanilla GD, the critical threshold is related to sharpness as $\eta_c \approx 2/\lambda^H$ [1], and the self-stabilization mechanism can be described as a four-step process [22, 8]. To illustrate this, consider the $T_{\text{wrm}} = 64$

[1]This relationship holds for the MSE loss and simple settings only. For an overview of instability thresholds in various settings and different optimizers, see Appendix C.1.

trajectories depicted in Figure 1(c, d). In the sharpness plot, the dashed lines represent the ${}^{2}/_{\eta_t}$ curves, and when λ_t^H is above these curves, training exceeds the instability threshold ($\eta > \eta_c$). The four steps of the self-stabilization mechanism are:

(1) **Approaching instability:** Due to increasing learning rate and/or progressive sharpening, training approaches the instability threshold $\eta = \eta_c$. In Figure 1(d), this occurs within the first 10 steps due to increasing learning rate.

(2) **Loss increases:** The loss begins to rise when the instability threshold is exceeded ($\eta > \eta_c$), as seen in Figure 1(c).

(3) **Sharpness reduction:** For small enough learning rates, the increasing loss causes an abrupt decrease in sharpness, as observed in Figure 1(d). If the sharpness fails to decrease over extended steps, it may result in training divergence (e.g., see $T_{\text{wrm}} = 1$ trajectories in the same figure).

(4) **Return to stability:** The reduction in sharpness causes $\eta_c = {}^{2}/_{\lambda^H}$ to increase, restoring stability ($\eta < \eta_c$) and allowing for an eventual loss decrease.

While the self-stabilization process for more complex optimizers, such as SGD with momentum or Adam, remains poorly understood, a qualitatively similar mechanism is observed in practice, as we will see in the later sections.

The critical learning rate η_c is influenced by a variety of factors, including the choice optimizer [6, 7], mini-batch size [34, 7], and model properties such as depth, width, parameterization, and initialization [18, 19]. For a detailed overview of instability thresholds, see Appendix C.

C Instability Thresholds

C.1 Overview of Instability Thresholds

Lewkowycz et al. [22] showed that for wide networks in NTP/SP trained with MSE loss and SGD, this critical learning rate is ${}^{2}/_{\lambda_0^H}$ early in training. Further investigation by Kalra and Barkeshli [18] demonstrated that sharpness reduction during early training causes η_c to increase with depth and ${}^{1}/_{\text{width}}$. In such scenarios, η_c can be as large as ${}^{40}/_{\lambda_0^H}$. Cohen et al. [6] demonstrated that sharpness at late training times for GD with momentum coefficient β oscillates above ${}^{(2+2\beta)}/_{\eta}$, suggesting $\eta_c \gtrsim {}^{(2+2\beta)}/_{\lambda_t^H}$ at late training times. Expanding on this, Cohen et al. [7] analyzed adaptive optimizers and found that for Adam, the pre-conditioned sharpness $\lambda^{P^{-1}H}$ oscillates around ${}^{(2+2\beta_1)}/_{\eta(1-\beta_1)}$ at late training times. The instability threshold also depends on the mini-batch size [34] and is often observed to be smaller than their full batch counterparts [6, 7].

D Experimental Details

This section provides additional experimental details. All models were implemented using the JAX [3], and Flax libraries [15]. The key results can be reproduced using the GitHub repo: `https://github.com/dayal-kalra/why-warmup`.

Experimental Setup for Section 4: We consider WideResNets (WRNs) and Transformers (LM) parameterized in either SP or μP. WRNs are trained on standard classification tasks such as CIFAR-10, CIFAR-100, and Tiny-ImageNet, employing data augmentation. Transformers are trained on the next token prediction task using the WikiText-2 dataset. These models are trained with MSE or cross-entropy (xent) loss functions using SGD or Adam optimizers for a fixed training budget of $T = 10^5$ steps unless otherwise specified. Training begins with a linear warmup phase from $\eta_{\text{init}} = 0$ to η_{trgt} over T_{wrm} steps. After the warmup phase, training continues at η_{trgt} for the remaining training budget. In some cases, following the warmup period, we gradually decrease the learning rate using cosine decay [24]. Target learning rates are sampled exponentially until divergence or a 'training failure' is observed. Here, training failure refers to instances where the performance at the end of the training fails to improve significantly compared to its initial value. For example, if the final training accuracy for a classification task is less than 1.5 times the accuracy of a random guess, we consider it as a training failure. We refer to the transition between convergence and training failure as the failure boundary. Further details are provided in Appendix D.

D.1 Datasets Details

D.1.1 Image Classification Tasks

We consider standard image classification datasets such as CIFAR-10, CIFAR-100 [21], and Tiny-ImageNet [1]. The images are normalized to have zero mean and unit variance. For MSE loss, we use one-hot encoding for the labels.

Data augmentation: For various image classification tasks, we employ data augmentation techniques, applied in the following order: random horizontal flips, random cropping, and mixup [40].

D.1.2 Language Modeling Tasks

We consider the next token prediction task on the Wikitext-2 dataset [25], consisting of $\sim$ 2M tokens. We use Byte Pair Encoding (BPE) tokenizer [31] with a Whitespace pre-tokenizer. Due to the high computational cost associated with hyperparameter tuning, we restrict to smaller models with $\sim$ 2M parameters. Furthermore, we restrict the vocabulary size to 4096 to ensure that embedding parameters do not dominate the total number of parameters in the model.

D.2 Model Details

This section describes the models considered, including their parameterization and initialization details. We adopt parameterizations outlined in Table 9 of Ref. [37]. Unless otherwise specified, we employ ReLU non-linearities and initialize the weights with a truncated normal distribution [2], with a variance $\sigma_w^2 = 2.0$ in appropriate parameterizations (details below), except for the last layer, which has a weight variance of $\sigma_w^2 = 1.0$. All biases are initialized to zeros.

D.2.1 Parameterizations

Standard Parameterization (SP): For SP, the weights are initialized with truncated Gaussian distribution $\mathcal{N}(0, \sigma_w^2/\text{fan}_\text{in})$ and the biases are initialized to zero.

Maximal Update Parameterization (μP): For μP, different schemes are employed for the intermediate and last layers. The intermediate layers are initialized using $\mathcal{N}(0, \sigma_w^2/\text{fan}_\text{out})$ and the layer outputs are scaled by the factor $\sqrt{\text{fan}_\text{out}/\text{fan}_\text{in}}$. In comparison, the layer weights are initialized with $\mathcal{N}(0, \sigma_w^2/\text{fan}_\text{in})$, and the final output is rescaled by the factor $\sqrt{1/\text{fan}_\text{in}}$. Conveniently, for SGD, the learning rate does not scale with width in the above μP formulation. In comparison, for Adam, the learning rate corresponding to input, intermediate, and output layers are rescaled by the factors $1/\sqrt{\text{fan}_\text{out}}$, $1/\sqrt{\text{fan}_\text{in}}$ and $1/\text{fan}_\text{in}$. Since we are utilizing μP only to obtain flat initializations, we omit the additional scaling of the learning rate for Adam in some experiments (e.g., Figure 2). As a result, the instability threshold is only dependent on the target learning rate η_trgt during late training, rather than on the largest learning rate across layers. We refer to this parameterization as 'simple-μP' for Adam.

D.2.2 Architectures

Fully Connected Networks (FCNs): We consider fully connected networks with a constant width of n and a depth of d layers. These networks are denoted by FCN-d-n. Unless specified, we considered $d = 4$ layer FCNs with width $n = 512$.

WideResNets (WRNs): We consider WideResNets [38] with d layers, S stages, and a widening factor of k, denoted by WRN-d-k. The number of channels in each stage $s \in [0, S)$ is given by $2^s \times 16 \times k$, with the input layer having 16 channels. For example, WRN-16-4 consists of $S = 3$ stages, each with $[2, 2, 2]$ layers, and the corresponding number of channels in each stage is $[64, 128, 256]$. In all our experiments, we use LayerNorm instead of BatchNorm.

Transformers: We consider Transformers with GPT-2 style architecture [29]. These models use sinusoidal positional embeddings [33] and are implemented in the Standard Parameterization (SP)

[2] for details, see `https://jax.readthedocs.io/en/latest/_autosummary/jax.nn.initializers.truncated_normal.html`

with GELU activation [16]. We initialize all layers using the $\sigma_w^2/\text{fan}_{\text{in}}$ scheme, except for the embedding layers, as they do not involve matrix multiplication [9]. We consider both Pre-LN [35] and Post-LN [33] Transformer variants. We denote a Transformer with d blocks and an embedding dimension of n as LM-d-n. Unless specified, the model has $d = 4$ blocks, embedding dimension $n = 128$, context length $T_{\text{cntxt}} = 64$ and are trained for 10^4 steps.

D.3 Optimization Details

D.3.1 Optimizers

SGD(-M): Given gradients $\boldsymbol{g}_t$ at step t, Stochastic Gradient Descent with momentum (SGD-M) updates the parameters $\boldsymbol{\theta}_t$ using learning rate η_t and momentum $\boldsymbol{m}_t$ with coefficient β. The update equations are:

$$\boldsymbol{m}_t = \boldsymbol{g}_t + \beta \boldsymbol{m}_{t-1}, \tag{1}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{m}_t. \tag{2}$$

Here, $\beta = 0$ corresponds to SGD. In all experiments incorporating momentum, the default value of the coefficient is set to $\beta = 0.9$.

Adam: Given gradients $\boldsymbol{g}_t$ at step t, Adam [20] updates the parameters $\boldsymbol{\theta}_t$ using learning rate η_t and the first two moments of the gradient $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ with their coefficients β_1 and β_2, respectively. The equations governing the updates are:

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t, \tag{3}$$

$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2, \tag{4}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}, \tag{5}$$

where $\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}$ and $\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}$ are the bias-corrected moments, and ϵ is a small scalar used for numerical stability. The pre-conditioner for Adam is given by:

$$P_t = (1 - \beta_1^t) \left[\text{diag}\left(\frac{\boldsymbol{v}_t}{1 - \beta_2^t} \right) + \epsilon \mathbf{I} \right]. \tag{6}$$

In all experiments, the default values are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, unless otherwise specified.

D.3.2 Linear Warmup

Warmup linearly increases the learning rate from an initial value η_{init} to a target value η_{trgt} over T_{wrm} training steps. The learning rate η_t at step t is given by:

$$\eta_t = \eta_{\text{init}} + (\eta_{\text{trgt}} - \eta_{\text{init}}) \left(\frac{t}{T_{\text{wrm}}} \right). \tag{7}$$

Here, $\alpha := \frac{(\eta_{\text{trgt}} - \eta_{\text{init}})}{T_{\text{wrm}}}$ is referred to as the rate of warmup. Under the above definition, constant learning rate training corresponds to $T_{\text{wrm}} = 1$. $T_{\text{wrm}} = 1$ corresponds to constant learning rate. Unless otherwise specified, we set $\eta_{\text{init}} = 0$ when referring to linear warmup.

D.3.3 Learning Rate Decay

In several experiments, we employ learning rate decay following the warmup phase. Specifically, we use cosine learning rate decay, which is detailed below.

Cosine Decay: Towards the end of training, it is typical to reduce the learning rate to a small value. Cosine decay is a commonly used method for decaying the learning rate from an initial value of η_{trgt} down to a value $\eta_{\min}$ over $T_{\cos}$ steps, according to the rule:

$$\eta_t = \eta_{\text{trgt}} + (\eta_{\min} - \eta_{\text{trgt}}) \left[\frac{1}{2} \left(1 + \cos \left(\frac{\pi t}{T_{\cos}} \right) \right) \right]^{\rho}, \tag{8}$$

where ρ governs the rate of decay, with $\rho = 1$ being the standard. Note that with $\rho = 0$, the learning rate is not decayed and instead maintained at η_{trgt}. In the above expression, t counts the steps from the initiation of cosine decay and not the current training step. As per standard practice, we consider $\rho = 1$ and decay the learning rate to $\eta_{\min} = \eta_{\text{trgt}}/10$.

D.3.4 Target Learning Rate Sampling for Phase Diagrams

For SGD, target learning rates η_{trgt} are exponentially sampled using the initial sharpness λ_0^H. Starting with $\eta_{\text{trgt}} = 1/\lambda_0^H$, subsequent rates are sampled until divergence as $2^x/\lambda_0^H$ for values of x increased in integer steps starting from zero. For WRNs trained with Adam, we sample target learning rates exponentially as $\eta_{\text{trgt}} = 2^x \times 10^{-5}$, where x is incremented in integer steps starting from zero until training failure. For Transformers, we sample the learning rate in a similar fashion but starting from 10^{-4} and increment x in steps of 0.5.

D.4 Sharpness and Pre-conditioned Sharpness Measurement

We measured sharpness / pre-conditioned sharpness using the JAX implementation of the LOBPCG sparse eigenvalue solver with the tolerance set to 10^{-9} and maximum number of iterations to $n_{\text{iter}} = 1000$. In most cases, the solver converges within 40 iterations. We performed these computations in float64, as the solver would not converge with float32 in some cases.

In certain instances, the pre-conditioned sharpness computation did not converge within 1000 solver iterations. Moreover, we observed that the solver converges on restarting it with a new initial guess of the eigenvector within 40 iterations. To address these edge cases, we employed the following method: if the solver did not converge within 100 iterations, we restarted it with a new initial guess for the eigenvector. We allowed for at most 10 restarts with the maximum number of iterations set to $n_{\text{iter}} = 1000$ in the last attempt. In all reported cases, the solver converges using this method.

D.5 Additional Figure Details

Figure 1: Training trajectories of 4-layer FCNs with width $n = 512$, trained on a 5k subset of CIFAR-10 using MSE loss and GD in (top) μP with $\eta_{\text{trgt}} = 1/\lambda_0^H$, where $\lambda_0^H \approx 0.05$, and (bottom) SP with $\eta_{\text{trgt}} = 32/\lambda_0^H$, where $\lambda_0^H \approx 50$.

Figure 2: Training loss and sharpness trajectories of 4 layer FCNs with width $n = 512$, in (top) μP with learning rate $\eta_{\text{trgt}} = 0.003$ and (bottom) SP with $\eta_{\text{trgt}} = 0.001$ trained the CIFAR-10 dataset with MSE loss using full batch Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. In these experiments, we use data augmentation as described in Appendix D.1.1.

Figure 3: Test accuracy heatmaps of WRN-16-4 trained on CIFAR-10 using different parameterizations and loss functions using SGD with a batch size $B = 128$: (a) SP and MSE loss, (b) μP and cross-entropy loss (c) SP and cross-entropy loss. All models are trained for 10^5 steps. In these experiments, we use data augmentation as described in Appendix D.1.1.

Figure 4: Test loss heatmaps of Pre-LN Transformers in SP trained on WikiText-2 with cross-entropy loss using (a) Adam, and (b) GI-Adam (introduced in Section 5) over Adam. The Transformer models have $d = 4$ blocks, embedding dimension $n = 128$, a context length of $T_{\text{cnxt}} = 64$. These experiments also employ cosine decay, as described in Appendix D.3.3.

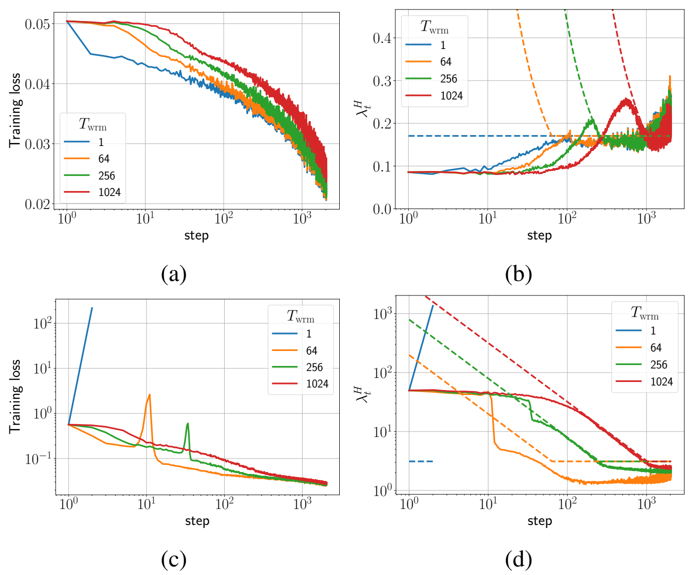


Figure 5: Training loss and sharpness trajectories of FCNs trained on CIFAR-10 with MSE loss using SGD with a batch size $B = 512$. The dashed lines in the sharpness figures illustrate the instability thresholds $^2/_{\eta_t}$. (top) μP with learning rate $^1/_{\lambda_0^H}$, (bottom) SP with learning rate $^{32}/_{\lambda_0^H}$.

D.6 Estimation of Computational Resources

The phase diagram experiments typically required about an hour on per run on an A100 GPU. Consequently, each phase diagram consumed approximately 100 A100 hours of computational time. With a total of 16 phase diagrams, this equates to 1600 A100 hours dedicated solely to phase diagram computations. Additionally, the warmup mechanism experiments, which were conducted over 2000 steps, required sharpness estimation. The FCN experiments required approximately 1200 A100 hours, while the WRN mechanism experiments consumed 1600 A100 hours. The experiments concerning the initial learning rate took about 20 A100 hours. This brings the total computational time amounted to approximately 4500 A100 hours. Preliminary experiments took about 1000 A100 hours. Hence, we estimate the total computational cost to be around 5500 A100 hours.

E Additional Results for Mechanisms of Warmup

This section presents additional trajectories for warmup mechanisms discussed in Section 3 covering various architectures, loss functions, and optimizers.

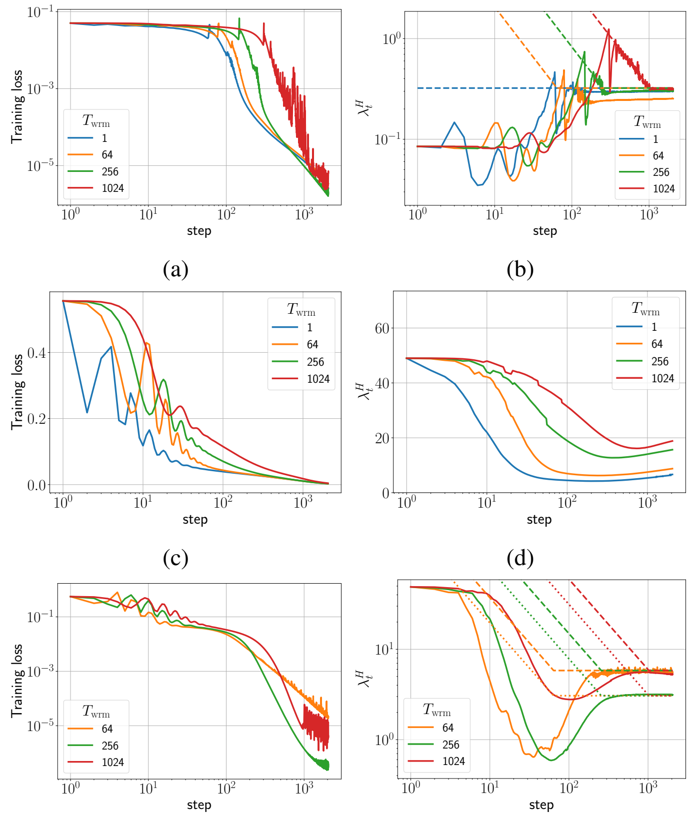
E.1 Stochastic Gradient Descent

Figure 5 shows that the warmup mechanisms for full batch GD are also observed in the SGD with a batch size $B = 512$. The results for other optimizers in the mini-batch setting are discussed in their respective sections.

E.2 Stochastic Gradient Descent with Momentum

While the warmup mechanisms of SGD with momentum are fundamentally similar to those of vanilla SGD, three key differences arise, as discussed below.

During early training, the loss may decrease non-monotonically on incorporating momentum, even at small learning rates. Such oscillations are also observed when quadratic loss functions are optimized using GD with momentum [11]. These oscillations make it challenging to differentiate between warmup-induced catapults and fluctuations in loss due to the intrinsic effects of momentum. Nevertheless, we can still observe loss spikes correlated with an abrupt decrease in sharpness at large learning rates, as detailed in Appendix E.2.



(a) (b)

(c) (d)

(e) (f)

Figure 6: Training loss and sharpness trajectories of FCNs trained on 5k subset of CIFAR-10 using MSE loss and full batch GD with momentum $\beta = 0.9$: (top) μP with learning rate $1/\lambda_0^H$ (middle) SP with learning rate $1/\lambda_0^H$, and (bottom) SP with learning rate $32/\lambda_0^H$. The dotted lines in the sharpness figures correspond to the $(2+2\beta)/\eta_t$ curves, while dashed lines show the $2/\eta_t$ for reference.

Additionally, the instability threshold η_c itself evolves differently during training. It changes from $2/\lambda_0^H$ at initialization to $(2+2\beta)/\lambda_t^H$ later in training. Moreover, the late-time instability threshold is significantly influenced by the batch size, exhibiting a much smaller value than SGD for the same batch size. These properties make it more challenging to analyze the training dynamics of SGD with momentum. Nonetheless, the fundamental warmup mechanisms closely mirror the vanilla SGD case. We leave a more detailed analysis of the early training dynamics of SGD-M for future studies.

Besides these differences, we note that the warmup mechanisms of SGD with momentum are similar to the vanilla SGD case. We leave a thorough analysis of the early sharpness dynamics of SGD with momentum for future works.

E.3 Stochastic Gradient Descent and Cross-entropy Loss

The warmup mechanisms for models trained with cross-entropy loss exhibit trends similar to those observed with MSE loss with one crucial difference. Near convergence, sharpness first increases and then abruptly decreases. The decrease in sharpness towards the end of training is observed in previous studies analyzing SGD with fixed learning rate [6]. Additionally, we observe higher fluctuations compared to the MSE loss case. Figure 8 shows trajectories of FCNs under different parameterizations trained on CIFAR-10 with cross-entropy loss using vanilla SGD. Meanwhile, Figure 9 shows the loss and sharpness trajectories of FCNs in SP trained on CIFAR-10 with cross-entropy loss using full batch GD with and without momentum.

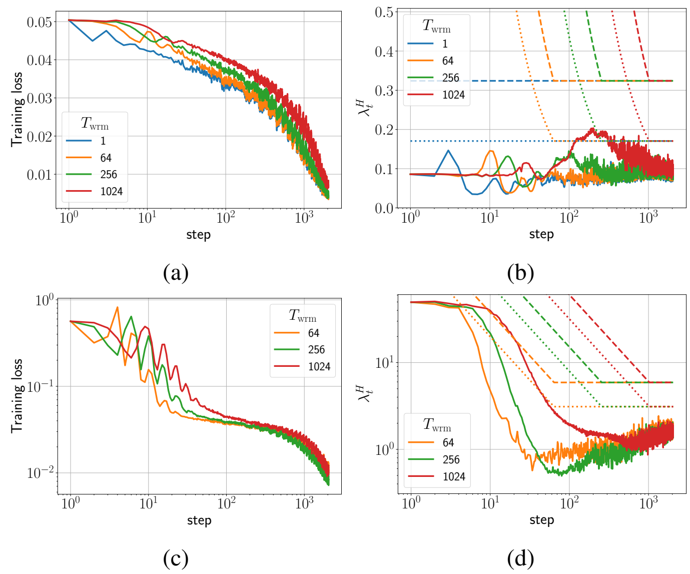


Figure 7: Training loss and sharpness trajectories of FCNs trained on CIFAR-10 with MSE loss using SGD with a batch size $B = 512$ and momentum $\beta = 0.9$: (top) μP with learning rate $1/\lambda_0^H$, and (bottom) SP with learning rate $32/\lambda_0^H$. The dotted lines in the sharpness figures correspond to the $(2+2\beta)/\eta_t$ curves, while dashed lines show the $2/\eta_t$ for reference. Similar mechanisms are observed for cross-entropy loss with a decrease in sharpness at late training times, as detailed in Appendix E.3.

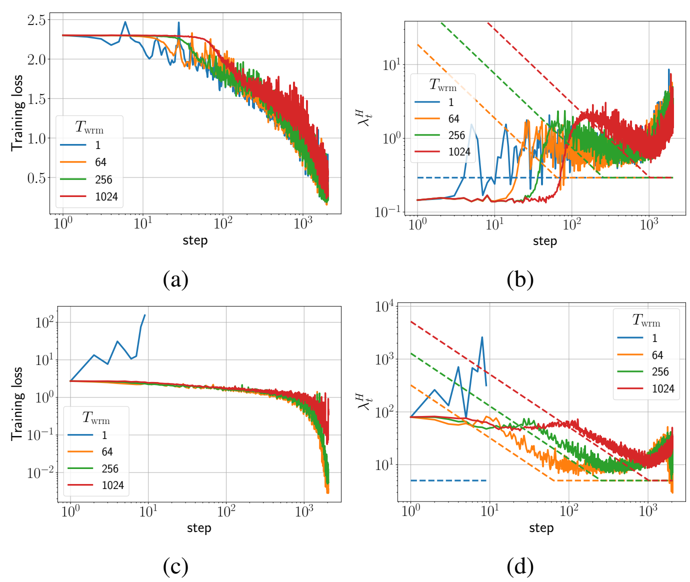


Figure 8: Training loss and sharpness trajectories of FCNs trained on CIFAR-10 with cross-entropy loss using SGD with a batch size $B = 512$. (Top row) μP with learning rate $1/\lambda_0^H$ (Bottom row) SP with learning rate $32/\lambda_0^H$.

E.4 Warmup Mechanisms of Adam

As discussed in Section 3.2, the instability threshold for Adam is determined by the pre-conditioned sharpness $\lambda^{P^{-1}H}$ and not by the sharpness itself. Moreover, training dynamics falls under the sharpness reduction case as the pre-conditioned sharpness starts off large and reduces considerably during the first few training.

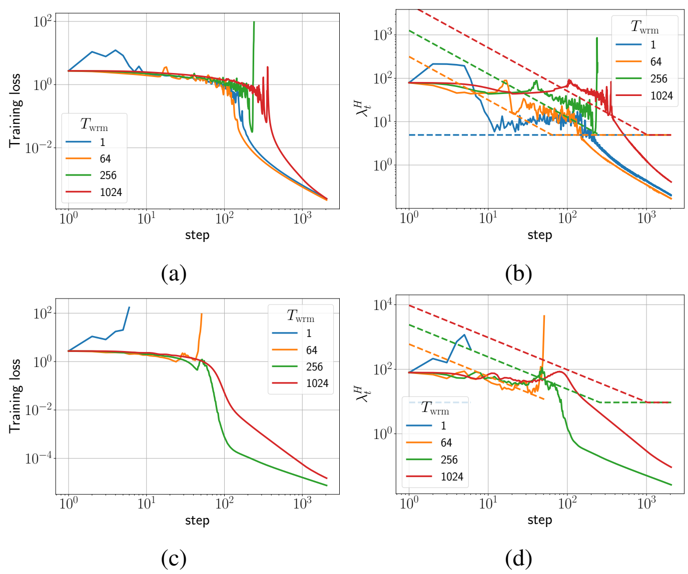


Figure 9: Training loss and sharpness trajectories of FCN-4-512 in SP trained on 5k subset of CIFAR-10 with cross-entropy loss using full batch GD with learning rate $32/\lambda_0^H$ with momentum coefficient (top) $\beta = 0.0$ and (bottom) $\beta = 0.9$.

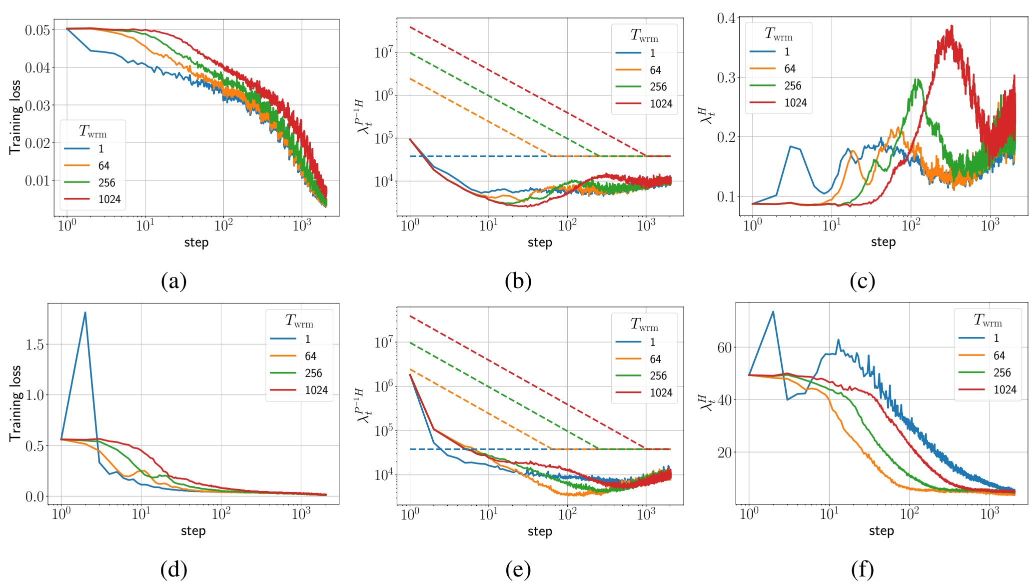


Figure 10: Training loss and sharpness trajectories of FCN-4-512 in (top) μP and (bottom) SP trained on CIFAR-10 with MSE loss using Adam with learning rate $\eta = 0.001$, batch size $B = 512$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The dashed lines in the sharpness figures illustrate the instability thresholds $(2+2\beta_1)/\eta_t(1-\beta_1)$.

Figure 10 shows the training trajectories of FCNs trained with Adam in the same setting as in Figure 2 but with a batch size of $B = 512$. Similar to the SGD with momentum case, the late time sharpness oscillates far below the instability threshold ($(2+2\beta_1)/\eta_t(1-\beta_1)$), suggesting that the instability threshold heavily decreases with a smaller batch size. We note similar findings by Ref. [7].

Next, Figure 11 show the warmup mechanism of FCNs trained with cross-entropy loss using Adam under the full-batch setting. Similar to the SGD case, the pre-conditioned sharpness decreases towards the end of training.

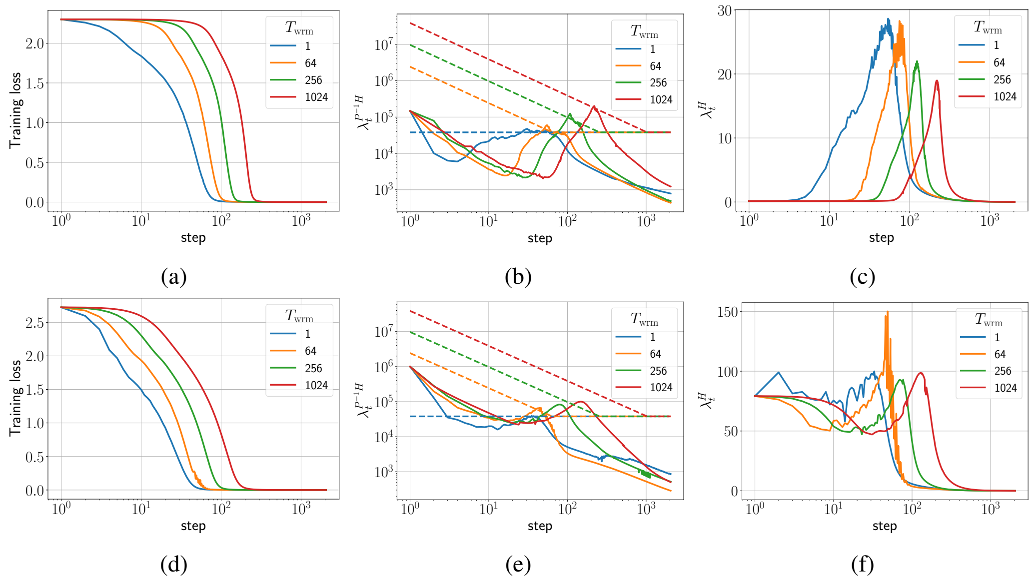


Figure 11: Training loss and sharpness trajectories of FCNs in (top) μP and (bottom) SP trained on CIFAR-10 with cross-entropy loss using full-batch Adam with learning rate $\eta = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The dashed lines in the sharpness figures illustrate the instability thresholds ${}^{(2+2\beta_1)}/_{\eta_t(1-\beta_1)}$.

E.5 Different Architectures and Datasets

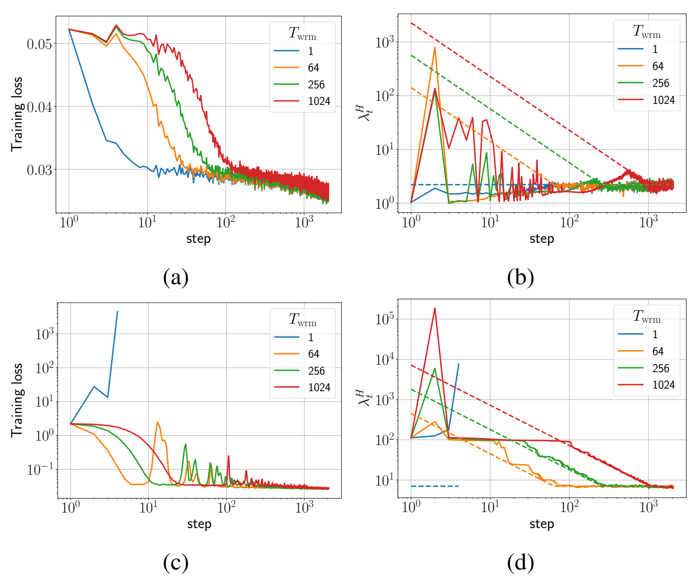


Figure 12: WRN-16-1 trained on CIFAR-10 with MSE loss using vanilla SGD with batch size $B = 512$: (top) μP with $\eta_{\text{trgt}} = {}^{1}/_{\lambda_0^H}$ and (bottom) SP with $\eta_{\text{trgt}} = {}^{32}/_{\lambda_0^H}$.

In the previous sections, we confined our analysis to FCNs to thoroughly explore the effects of different optimizers and loss functions. This section expands on those results by demonstrating that the observed warmup mechanisms apply to ResNets and Transformers as well. The Resnet experiments also employ data augmentation as detailed in Appendix D.1.

Figures 12 and 13 show the training trajectories of WideResNets (WRNs) trained on CIFAR-10 with MSE and cross-entropy loss using SGD. These trajectories generally reflect the warmup mechanisms discussed in Section 3. However, certain additional features obscure the clarity of these mechanisms.

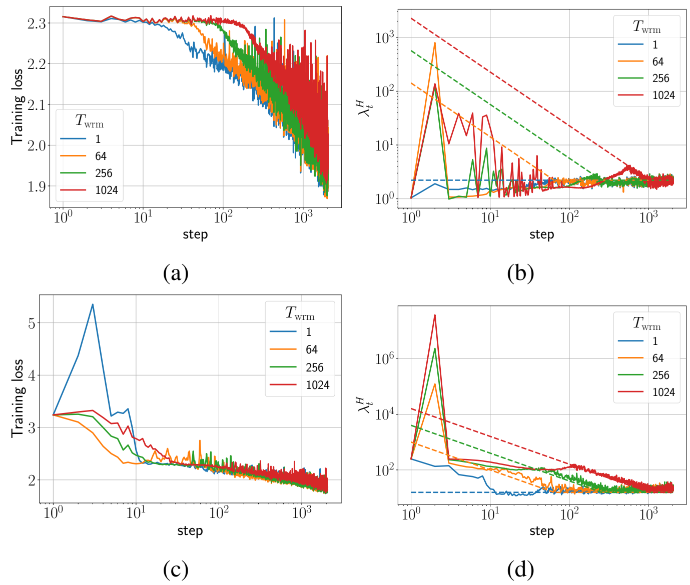


Figure 13: WRN-16-1 trained on CIFAR-10 with cross-entropy loss using vanilla SGD with batch size $B = 512$: (top) μP with $\eta_{\text{trgt}} = 1/\lambda_0^H$ and (bottom) SP with $\eta_{\text{trgt}} = 32/\lambda_0^H$.

Notably, we observed a significant sharpness spike on the first training step when using longer warmup durations, which automatically resolves in the subsequent step. The magnitude of this spike increases with longer warmup periods. Further analysis revealed that this phenomenon is associated with an initial increase in the first LayerNorm parameters, which also resolves automatically by the second step. Beyond this observation, the training trajectories align with the warmup mechanisms described in the main text.

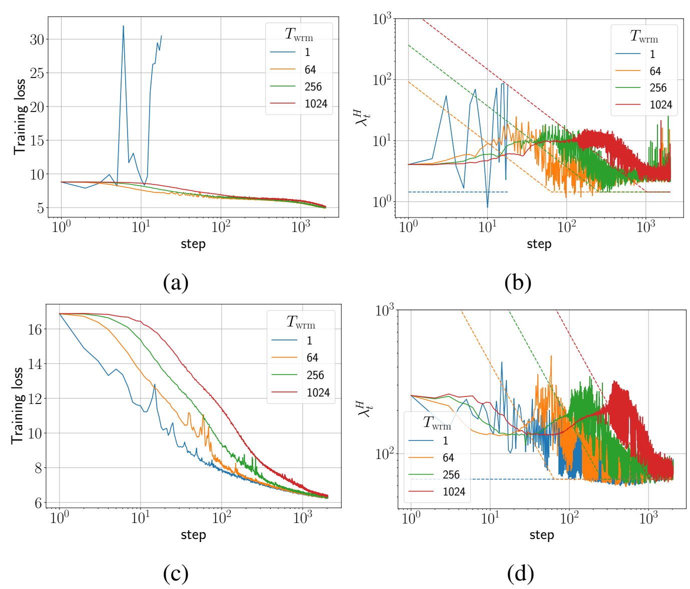
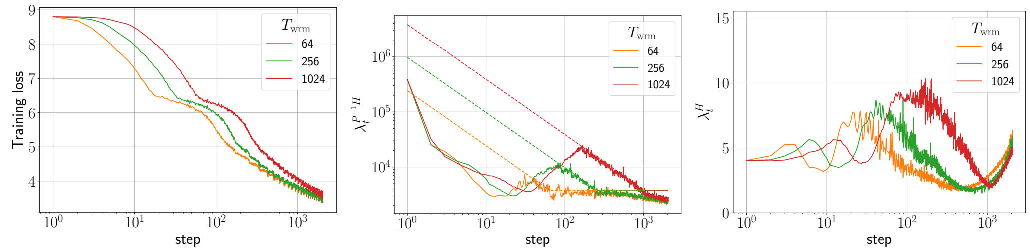


Figure 14: LM-4-128 trained on the WikiText-2 dataset with cross-entropy loss using SGD with a batch size $B = 512$ and a context length $T_{\text{cntx}} = 64$. The top row shows the warmup mechanisms of a Pre-LN Transformer with $\eta_{\text{trgt}} = 5.65/\lambda_0^H$, while the bottom row shows the results for the same Pre-LN Transformer but with the last LayerNorm removed and a learning rate of $\eta_{\text{trgt}} = 8/\lambda_0^H$.

Figure 14 illustrates the warmup mechanisms of Pre-LN Transformers trained on the WikiText-2 with SGD. The Pre-LN Transformer (top row) starts in a flat landscape region ($\lambda_0^H \sim 5$) and experiences



(a) (b) (c)

Figure 15: Pre-LN LM-4-128 trained on the WikiText-2 dataset with cross-entropy loss using Adam with a target learning rate $\eta_{\text{trgt}} = 0.003$, a batch size $B = 512$ and a context length $T_{\text{cntx}} = 64$.

progressive sharpening right from initialization. In contrast, when the last LayerNorm (just before the final linear layer) is removed (bottom row), the model starts training in a significantly sharper region, with the initial sharpness 100 times larger than the standard Pre-LN Transformer. This modified Pre-LN Transformer experiences a reduction in sharpness during the early stages of training.

Figure 15 presents the warmup mechanisms of Pre-LN Transformers trained on WikiText-2 using the Adam optimizer. Consistent with the results in the main text, the pre-conditioned sharpness exhibits a reduction early in training, despite the model initializing in a very flat region.

These experiments demonstrate that Transformers trained on language modeling tasks exhibit warmup mechanisms consistent with those discussed in the main text.

F Additional Phase Diagrams

This section presents further results related to the phase diagrams of warmup shown in Section 4.

F.1 Phase Diagrams for different Models and Datasets

Figure 16 shows the test accuracy heatmaps of WRN-16-4 trained on CIFAR-100 and Tiny-ImageNet. These models are trained using cross-entropy loss using SGD with a batch size of $B = 128$. Additional phase diagrams for Adam are presented in Appendix F.3.

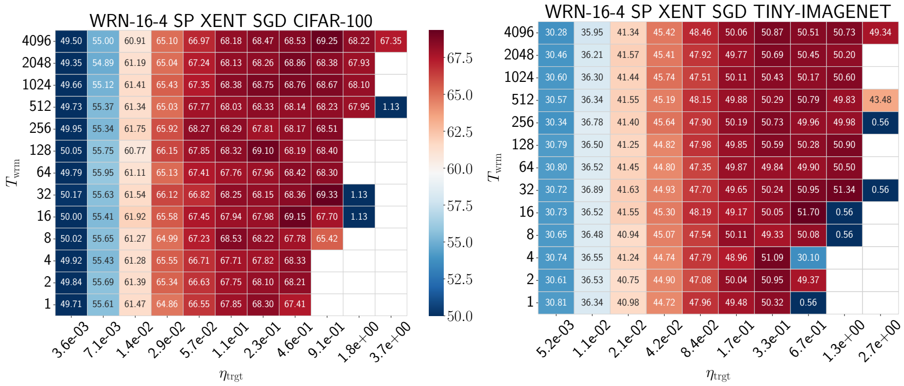
Figure 17(a) shows the test loss heatmaps of Pre-LN Transformer trained on the WikiText-2 dataset using SGD with a batch size $B = 64$. Figure 17(b) shows the Pre-LN Transformer under the same setup except for the last layer LayerNorm removed. The standard Pre-LN Transformer starts off with a small sharpness, while the version without the last LN starts off with 100 times higher curvature and requires warmup to achieve good performance.

F.2 The Effect of Momentum and Learning Rate Decay

Figure 18 shows that incorporating momentum and cosine decay (for details, see Appendix D.3.3) minimally affects the warmup phase diagrams. While the conclusions regarding warmup presented in the main text remain unaffected, we note a few interesting observations.

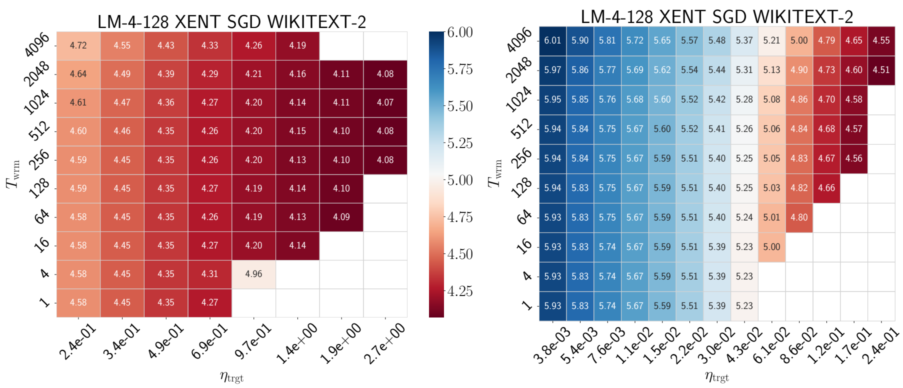
First, the divergent boundary shifts leftward on incorporating momentum, indicating that momentum permits smaller target learning rates without warmup, and warmup helps SGD-M more. Meanwhile, cosine decay has a minimal effect on the divergent boundary.

Additionally, we observe a performance enhancement by incorporating momentum, especially at small learning rates. In contrast, a decaying learning rate beyond warmup degrades performance at small learning rates while improving at higher ones. Finally, incorporating both momentum and cosine decay leads to further enhancement, indicating a synergistic interaction between the two.



(a) (b)

Figure 16: Test accuracy heatmaps of WideResNets (WRNs) in SP trained on (a) CIFAR-100 and (b) Tiny ImageNet with cross-entropy loss using SGD with batch size $B = 128$.

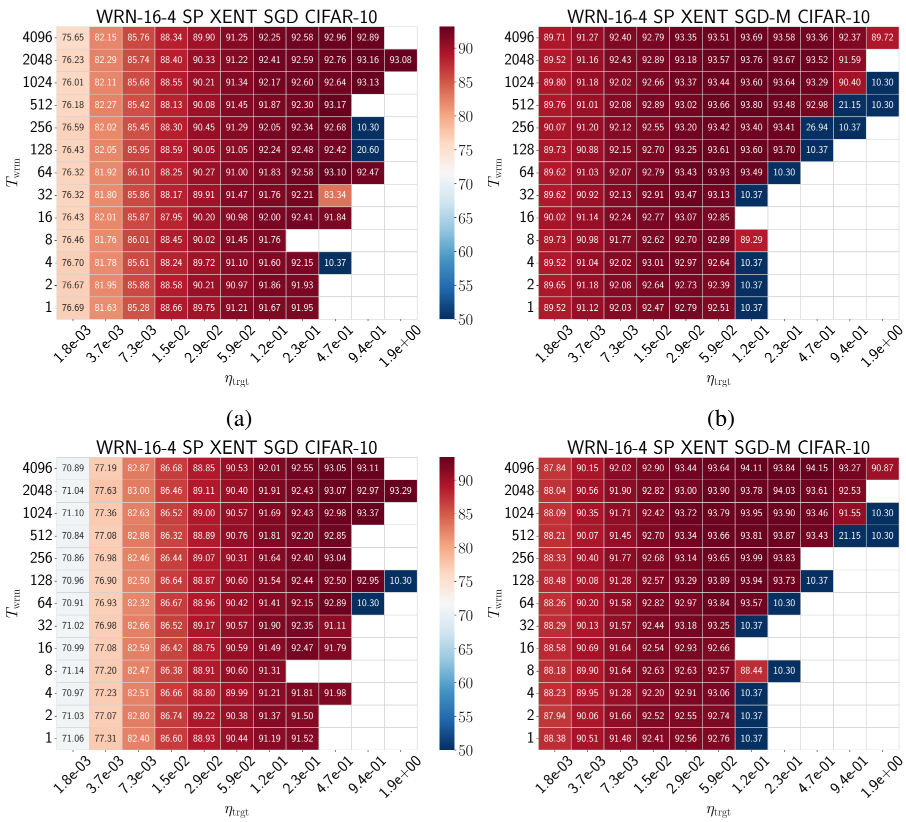


(a) (b)

Figure 17: Test loss heatmaps of LM-4-128 in SP trained on WikiText-2 with cross-entropy loss using SGD with a batch size $B = 64$: (a) Pre-LN Transformer and (b) Pre-LN Transformer without the last LayerNorm.

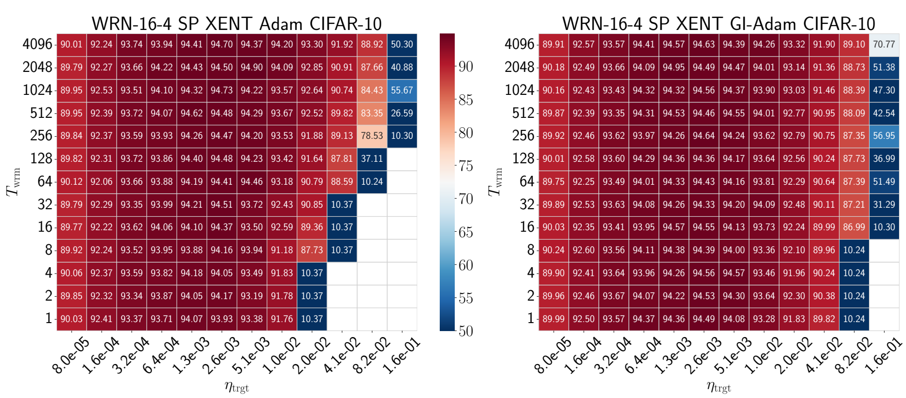
F.3 Phase Diagrams of Adam and GI-Adam

Figures 20 to 22 compare the warmup phase diagrams of Adam and GI-Adam of WRNs trained on CIFAR-100, Tiny-ImageNet and of Transformers trained on WikiText-2 dataset. Similar to the results shown in the main text, GI-Adam enhances performance over standard Adam by pushing the failure boundary.



(a) (b) (c) (d)

Figure 18: Test accuracy heatmaps of WideResNets (WRNs) in SP trained on CIFAR-10 with cross-entropy loss using SGD with batch size $B = 128$: (top row) no cosine decay (a) no momentum, (b) momentum with $\beta = 0.9$, and (bottom row) with cosine decay (c) no momentum, and (d) momentum with $\beta = 0.9$. The setting of (a) is the same as in Figure 3(c) but with a different mini-batch sequence.

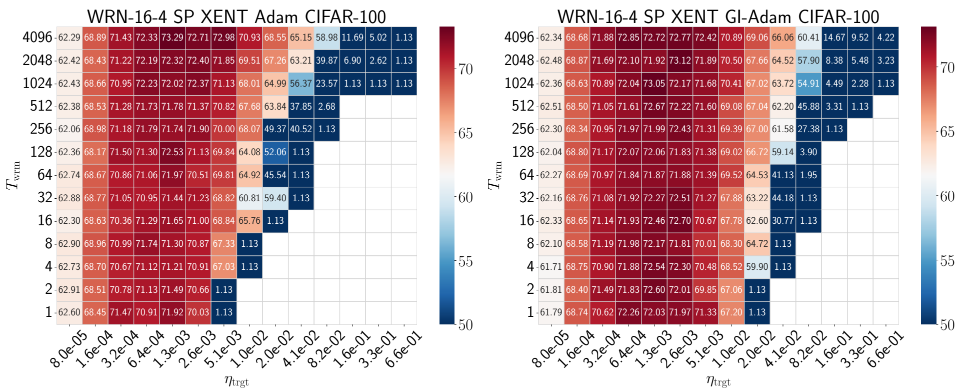


(a) (b)

Figure 19: Test accuracy heatmaps of WRN-16-4 trained on CIFAR-100 with cross-entropy loss using (left) standard Adam, and (right) GI-Adam with batch size $B = 128$.

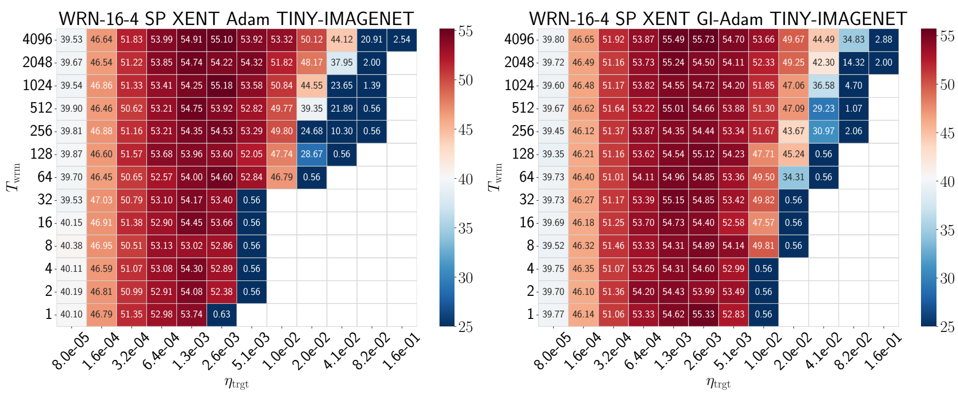
G Non-divergence of Adam

Figure 23 shows that, despite experiencing catastrophic instabilities during early training, Adam does not diverge well beyond the training failure boundary. While Adam can recover from these



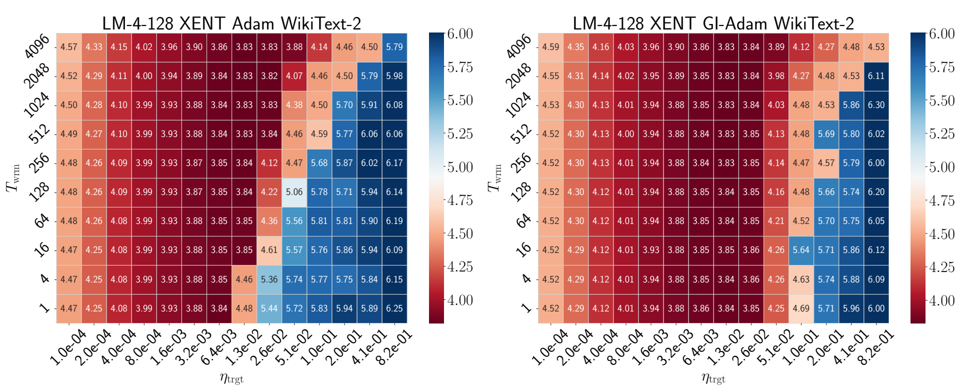
(a) (b)

Figure 20: Test accuracy heatmaps of WRN-16-4 trained on CIFAR-100 with cross-entropy loss using (left) standard Adam, and (right) GI-Adam with batch size $B = 128$.



(a) (b)

Figure 21: Test accuracy heatmaps of WRN-16-4 trained on Tiny-ImageNet with cross-entropy loss using (left) standard Adam, and (right) GI-Adam with batch size $B = 128$.



(a) (b)

Figure 22: Test loss heatmaps of LM-4-128 in SP trained on WikiText-2 with cross-entropy loss using (a) standard Adam, and (right) GI-Adam with batch size $B = 64$.

instabilities, the model's performance is severely impacted, resulting in training failures rather than convergence to a reasonable minimum.

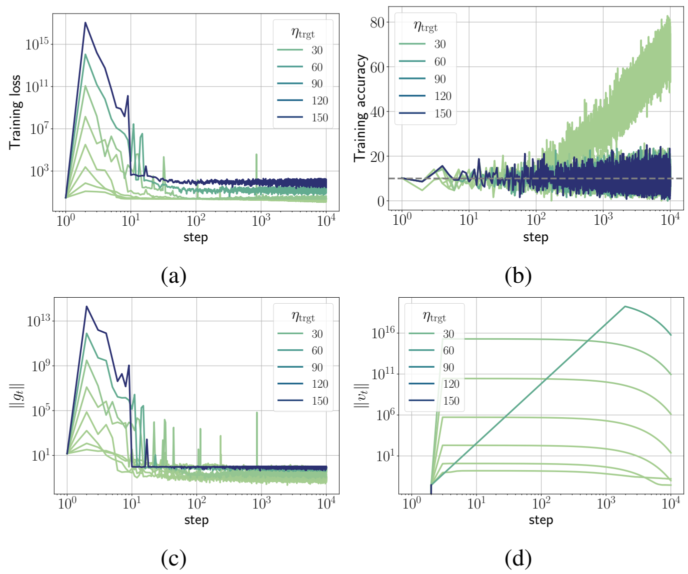


Figure 23: Training trajectories of WRNs trained on CIFAR-10 using Adam with cross-entropy loss and varying learning rates. The setup is identical to the $T_{\text{wrm}} = 1$ row of Figure 4, but without employing cosine learning rate decay. The first training failure is observed at a learning rate of $\eta_{\text{trgt}} = 0.02048$. To investigate the behavior beyond the training failure boundary, learning rates are sampled from $\eta_{\text{trgt}} = 0.01024$ (just below the failure boundary) up to $\eta_{\text{trgt}} \approx 150$.

These large loss catapults cause the gradients $\boldsymbol{g}$ to spike during early training, leading to a substantial increase in its second moment $\boldsymbol{v}$. While the gradients return to a lower value after a few training steps, the second moment remains large in magnitude for a prolonged period. These large values of $\boldsymbol{v}$ result in a small effective learning rate, which hinders training to escape these high-loss regions. Consequently, the models remain stuck in a suboptimal state rather than converging. We refer to this as a training failure.

Upon closer examination of the individual layers during training failures, we found that certain layers or residual blocks output zero. This results in vanishing gradients except for the last layer bias and training halts. We defer the detailed analysis of Adam's failures to future work.

H Additional Results on GI-Adam

This section presents additional results for GI-Adam. We provide further insights into the mechanisms and interpretations of GI-Adam.

H.1 Warmup Mechanisms of GI-Adam

Figure 24 shows the training trajectories of FCNs with different parameterizations trained with GI-Adam. Notably, the pre-conditioned sharpness starts at significantly lower values than standard Adam. Specifically, for the μP model, the initial pre-conditioned sharpness $\lambda^{P^{-1}H}$ is around 2000 instead of the value 10^5 observed for Adam (c.f. Figure 2). Remarkably, this almost eliminates initial sharpness reduction. Similarly, the pre-conditioned sharpness for the SP model starts around 10^4 instead of 10^6. Notably, in the SP scenario, there is no initial spike in the $T_{\text{wrm}} = 1$ (c.f. Figure 2), demonstrating that this simple modification effectively reduces instabilities during the early training.

H.2 GI-Adam as an Automated Warmup

In this section, we show that a bias correction is not required when the second moment is initialized with the gradients at initialization in GI-Adam. Therefore, employing a bias correction as in the original Adam algorithm in this case serves as an automated warmup given by $\eta_t = \eta_{\text{trgt}} \sqrt{1 - \beta_2^t}$.

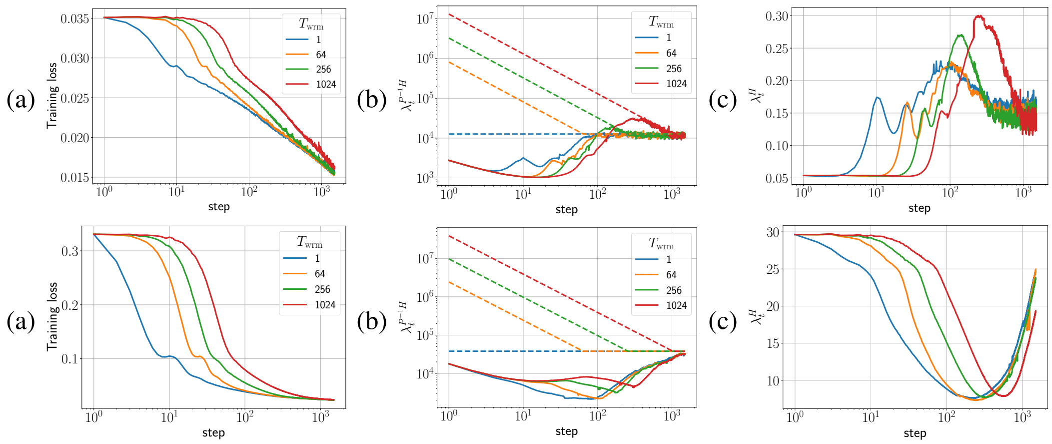


Figure 24: Training loss and sharpness trajectories of FCNs in (top) μP and (bottom) SP. The experimental setup is identical to Figure 2 but with GI-Adam instead of standard Adam.

The moving average of the second moment is given by:

$$\boldsymbol{v}_t = (1-\beta_2)\sum_{i=0}^{t-1}\beta_2^i \boldsymbol{g}_{t-i}^2 + \beta_2^t \boldsymbol{v}_0, \tag{9}$$

where $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$. Following standard assumptions, we assume that the second moment of the gradient is constant during early training $\mathbb{E}[\boldsymbol{g}_t^2] = \sigma^2$. Taking the expectation of the above equation over the gradient distribution yields

$$\mathbb{E}[\boldsymbol{v}_t] = (1-\beta_2)\sum_{i=0}^{t-1}\beta_2^i \mathbb{E}[\boldsymbol{g}_{t-i}^2] + \beta_2^t \mathbb{E}[\boldsymbol{v}_0]. \tag{10}$$

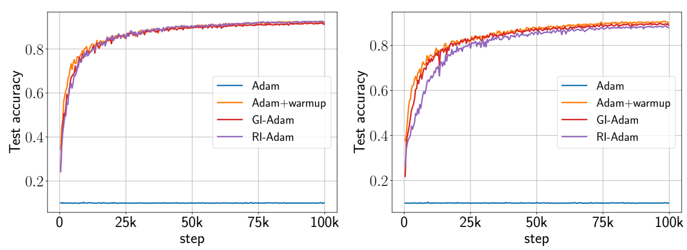
Simplifying the above equation, we have

$$\mathbb{E}[\boldsymbol{v}_t] = (1-\beta_2)\sigma^2\frac{1-\beta_2^t}{1-\beta_2} + \beta_2^t\sigma^2 = \sigma^2. \tag{11}$$

This result demonstrates that when the second moment is initialized with the gradients at initialization, it does not require bias correction, as the expected value of the second moment is equal to the constant σ^2. If we apply the usual bias correction on top of initializing the second moment with the gradients, we effectively downscale the second moment by a factor $\sqrt{1-\beta_2^t}$. Assuming small enough ϵ, this can be viewed as a multiplicative factor to the learning rate. As a result, GI-Adam is equivalent to having a natural warmup given by $\eta_t = \eta_{\text{trgt}}\sqrt{1-\beta_2^t}$.

H.3 The Primary benefit of GI-Adam results from the magnitude of the second moment at initialization

To further assess if the primary cause of instability during early training is the large $\lambda^{P^{-1}H}$, we randomly initialize $\boldsymbol{v}_0$ but with the same norm as the gradients at initialization. We refer to this as Randomly Initialized Adam (RI-Adam). Like GI-Adam, this also results in improved performance as shown in Figure 25.



(a) (b)

Figure 25: Comparison of test accuracy trajectories of WRNs trained with different Adam variants for two target different learning rates: (a) $\eta_{\text{trgt}} = 0.020480$, and (b) $\eta_{\text{trgt}} = 0.040960$. For Adam+warmup, the warmup duration is set to $T_{\text{wrm}} = 1024$.