On the Periodic Behavior of Neural Network Training with Batch Normalization and Weight Decay

Ekaterina Lobacheva[1][*], **Maxim Kodryan**[1][*], **Nadezhda Chirkova**[1]
Andrey Malinin[1,2], **Dmitry Vetrov**[1,3]
[1]HSE University   [2]Yandex   [3]AIRI
Moscow, Russia
elobacheva@hse.ru, mkodryan@hse.ru, nchirkova@hse.ru
am969@yandex-team.ru, dvetrov@hse.ru

Abstract

Training neural networks with batch normalization and weight decay has become a common practice in recent years. In this work, we show that their combined use may result in a surprising periodic behavior of optimization dynamics: the training process regularly exhibits destabilizations that, however, do not lead to complete divergence but cause a new period of training. We rigorously investigate the mechanism underlying the discovered periodic behavior from both empirical and theoretical points of view and analyze the conditions in which it occurs in practice. We also demonstrate that periodic behavior can be regarded as a generalization of two previously opposing perspectives on training with batch normalization and weight decay, namely the equilibrium presumption and the instability presumption.

1   Introduction

Normalization approaches, such as batch or layer normalization, have become vital for the successful training of modern deep neural networks [12, 2, 24, 21, 27]. Despite much recent work [3, 22, 9, 28], it is still not completely understood how normalization influences the training process. In this work, we investigate the surprising periodic behavior that may occur when a neural network is trained with a commonly used combination of some kind of normalization, in our case batch normalization (BN) [12], and weight decay regularization (WD). Examples of this behavior are provided in Figure 1.

The dynamics of neural network training with BN and WD have been examined extensively in literature due to the non-trivial competing influence of BN and WD on the norm of neural network's weights. More precisely, using BN makes (a part of) neural network's weights *scale-invariant*, i.e., multiplying them by a positive constant does not change the network's output. Although scale invariance allows optimizing on a sphere with a fixed weight norm [6], classic SGD-based approaches are usually preferred over constraint optimization methods in practice due to more straightforward implementation. Making an SGD step in the direction of the loss gradient always increases the norm of scale-invariant parameters, while WD aims at decreasing the weight norm (see illustration in Figure 2). In sum, training the neural network with BN and WD results in an interplay between two forces: a "centripetal force" of the WD and the "centrifugal force" of the loss gradients. Many works notice the positive effect of WD on optimization and generalization caused by the control of the scale-invariant weights norm and the subsequent influence on the *effective learning rate* [25, 10, 29, 18, 19, 26, 20], i.e., the learning rate on a unit sphere in the scale-invariant weights space. However, the general dynamics of the norm of the scale-invariant weights are viewed in the literature from two contradicting points, and this work is devoted to resolving this contradiction.

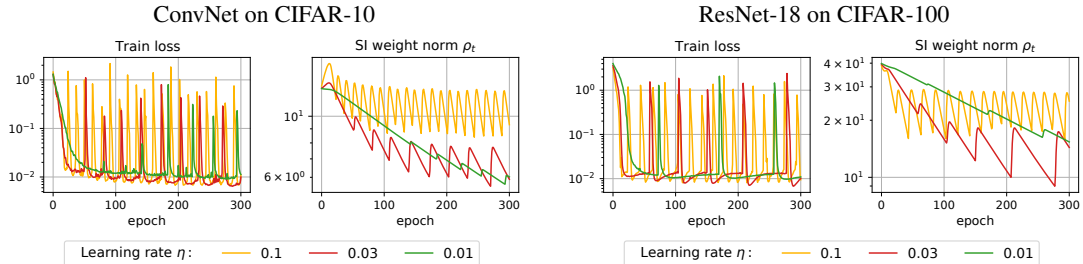[*]First two authors contributed equally.

Figure 1: Periodic behavior of ConvNet on CIFAR-10 and ResNet-18 on CIFAR-100 trained using SGD with weight decay of 0.001 and different learning rates. All weights are trainable, including non-scale-invariant ones.

On the one hand, Li et al. [19] claim that learning with SGD, BN, and WD leads to an *equilibrium* state, where the "centripetal force" is compensated by the "centrifugal force" and eventually the norm of scale-invariant weights (along with other statistics related to the training procedure) will converge to a constant value. Several other works hold a similar equilibrium view [25, 5, 26]. On the other hand, a number of works [17–19] underline that using WD may cause approaching the origin (zero scale-invariant weights), which results in training *instability* due to increasing effective learning rate. Particularly, Li et al. [17] reveal that approaching the origin in weight-normalized neural networks leads to numerical overflow in gradient updates and subsequent training failure. Li and Arora [18] also underline that scale-invariant functions are ill-conditioned near the origin and prove in a simplified setting that loss convergence is impossible if both BN and WD are used (but guaranteed if either of them is disabled). Moreover, despite their equilibrium view, Li et al. [19] empirically observe that the train loss permanently exhibits oscillations between low and high values when full-batch gradient descent is used.

In this work, we study the specified contradiction between the *equilibrium* presumption and the *instability* presumption and show that both are true only to some extent. Specifically, we show that the training process converges to a consistent *periodic* behavior, i.e., it regularly exhibits instabilities which, however, do not lead to a complete training failure but cause a new period of training (see Figure 1). Thus, our contributions are as follows.

- We discover the periodic behavior of neural network training with BN and WD and reveal its reasons by analyzing the underlying mechanism for fully scale-invariant neural networks trained with standard constant learning rate SGD (Section 4) or GD (Appendix C).

- We provide a theoretical grounding for our findings by generalizing previous results on the equilibrium condition, analyzing the necessary conditions for destabilization of training, and relating the frequency of destabilization to the choice of hyperparameters (Section 5).

- We conduct a rigorous empirical study of this periodic behavior (Section 6) and show its presence in more practical scenarios with momentum, augmentation, and neural networks incorporating trainable non-scale-invariant weights (Section 7), and also with Adam optimizer and other normalization techniques (Appendix I).

Our source code is available at `https://github.com/tipt0p/periodic_behavior_bn_wd`.

2 Background

As discussed in the introduction, batch normalization makes (a part of) neural network's weights scale-invariant. In this section, we describe the properties of scale-invariant functions, upon which we build our further reasoning. Consider an arbitrary scale-invariant function $f(x)$, i.e., $f(\alpha x) = f(x)$, $\forall x$ and $\forall \alpha > 0$. Then two fundamental properties may be inferred, see Lemma 1.3 in Li and Arora [18]:

$$
\begin{cases}
\langle \nabla f(x), x \rangle = 0, \ \forall x & \text{(1a)} \\
\nabla f(\alpha x) = \dfrac{1}{\alpha} \nabla f(x), \ \forall x, \ \alpha > 0. & \text{(1b)}
\end{cases}
$$

Consider optimizing $f(x)$ w.r.t. x using (S)GD[2] with learning rate η and weight decay λ:

$$x_{t+1} = (1 - \eta\lambda)x_t - \eta\nabla f(x_t). \tag{2}$$

The properties above lead to two important corollaries about the dynamics of the optimization process. First, according to property (1a), shifting x in the direction of $-\nabla f(x)$, i.e., making a gradient descent step, always increases $\|x\|$, while weight decay, on the other hand, decreases $\|x\|$. See Figure 2 for the illustration. The interaction of these "centripetal" and "centrifugal" forces may cause $\|x\|$ to change nontrivially during optimization. Second, according to property (1b), even though function value $f(x)$ is invariant to multiplying x by α, the optimization dynamics changes substantially when optimization is performed at different scales of $\|x\|$. For smaller norms, optimization makes larger steps, which may result in instabilities, while for larger norms, steps are smaller, and optimization process may converge slowly.

Figure 2: An illustration of the "centripetal force" of the weight decay and the "centrifugal force" of the function gradient in the optimization of scale-invariant functions.

Since scale-invariant $f(x)$ may be seen as a function on a sphere, its optimization dynamics are often analysed on a unit sphere $\|x\| = 1$. One can obtain equivalent optimization dynamics on the unit sphere as in the initial space by using the notion of *effective gradient* and *effective learning rate*. The effective gradient is defined as a gradient for a point on a unit sphere and may be obtained by substituting $\alpha = \|x\|^{-1}$ in (1b): $\nabla f(x/\|x\|) = \nabla f(x)\|x\|$. The effective learning rate can be defined as $\tilde{\eta} = \eta/\|x\|^2$ [10, 20]. Change in $\|x\|$ does not affect the effective gradient by definition and is reflected only in the effective learning rate: the lower the norm, the higher the effective learning rate, and the larger the optimization steps.
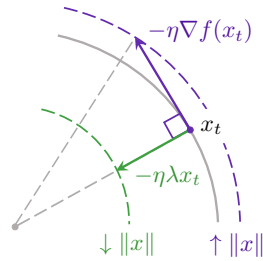
3 Methodology and experimental setup

In order to isolate the effect of the joint use of batch normalization and weight decay and avoid the influence of other factors, we conduct a series of experiments in a simplified setting, when all learnable weights of a neural network are scale-invariant and optimization is performed using SGD with constant learning rate, without momentum or data augmentation. This allows us to better understand the nature of the periodic behavior (Section 4) and analyse its empirical properties (Section 6). After that, we return to the setting with a more conventional training of standard neural networks and show that the periodic behavior occurs in this scenario as well (Section 7).

We conduct experiments with ResNet-18 and a simple 3-layer batch-normalized convolutional neural network (ConvNet)[3] on CIFAR-10 [15] and CIFAR-100 [16] datasets. To make standard networks fully scale-invariant, we rely on the approach of Li and Arora [18], i.e., we insert additional BN layers and fix the non-scale-invariant weights to be constant. Specifically, we use zero mean and unit variance in batch normalization layers instead of learnable location and scale parameters and freeze the weights of the last layer at random initialization. The latter action does not hurt the performance in practice [11]. However, we find that even with low train error, the training dynamics with the fixed last layer may still substantially differ from conventional training, as the neural network exhibits low confidence in predictions. To achieve high confidence for all objects and, consequently, low train loss, we increase the norm of the last layer's weights to 10. The influence of this rescaling is shown in Appendix G.

We optimize cross-entropy loss, use the batch size of 128 and train neural networks for 1000 epochs to show the consistency of the discovered periodic behavior. We consider a range of learning rates, $\{10^{-k}, 3 \cdot 10^{-k}\}_{k=0,1,2,3}$ and choose the most representative ones for each visualization, since it is difficult to distinguish many periodic functions on one plot. For fully scale-invariant neural networks, training with a fixed weight decay – learning rate product converges to similar behavior, regardless of their ratio: we show it empirically and discuss it from the theoretical point of view in Appendix F.1; the same was noticed in [18, 19]. Thus, in the main text, we provide only the results for the varied

---

[2]Since both stochastic and full-batch gradients of a scale-invariant objective possess properties (1a) and (1b), we do not distinguish between them in our reasoning.

[3]Both architectures in the implementation of `https://github.com/g-benton/hessian-eff-dim`.
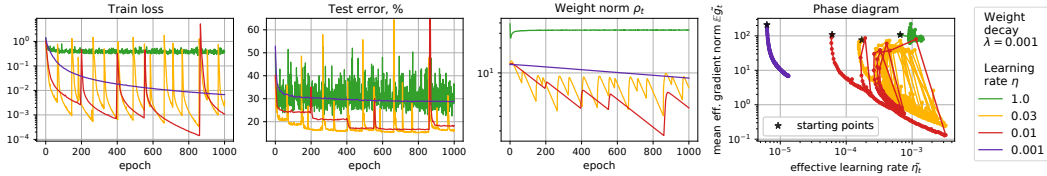
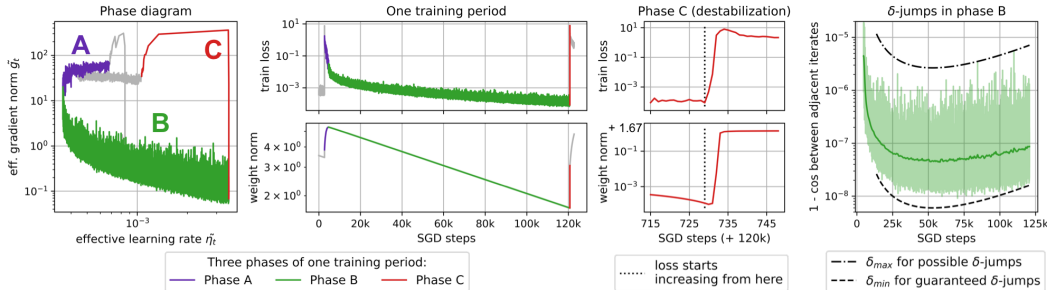Figure 3: Periodic behavior of scale-invariant ConvNet on CIFAR-10.

Figure 4: A closer look at one training period for scale-invariant ConvNet on CIFAR-10 trained using SGD with weight decay of 0.001 and the learning rate of 0.01. Three phases of the training period are highlighted. The train loss and the effective gradient norm computed over a mini-batch are logged after each SGD step (one epoch consists of 391 SGD steps). The rightmost plot compares empirically observed cosine distance between weights at adjacent SGD steps with theoretically derived bounds in Section 5.1. Cosine distance is presented along with the smoothed trend.

learning rate and the fixed weight decay of 0.001. Results for the varied weight decay are presented in Appendix F.3.

At each training epoch, we log standard train / test metrics, the norm of scale-invariant weights (SI weight norm), which is in the focus of this research, and metrics characterizing training dynamics on a unit sphere: effective learning rate and the norm of effective gradients (mean over mini-batches). We plot the two latter metrics over two axes of the *phase diagram* to visualize their simultaneous dynamics that will help us to understand the mechanism underlying the periodic behavior.

4 Periodic behavior and its underlying mechanism

As discussed in the previous section, we begin our study by considering a simplified setting with a fully scale-invariant neural network trained with standard SGD. Figure 3 shows the presence of the periodic behavior for a scale-invariant ConvNet on the CIFAR-10 dataset for a range of learning rates. In Appendix F.2, we show the presence of the periodic behavior for other dataset-architecture pairs. The same periodic behavior is also present for neural network training with full-batch gradient descent, see Appendix C. Moreover, this behavior can be observed even when optimizing common scale-invariant functions using the gradient descent method with weight decay (see Appendix E).

The observed periodic behavior occurs because of the interaction between batch normalization and weight decay, particularly because of their competing influence on the weight norm. As discussed in Section 2, weight decay aims at decreasing the weight norm, while loss gradients aim at increasing the weight norm due to scale invariance caused by batch normalization (see Figure 2). These two forces alternately outweigh each other for quite long periods of training, resulting in periodic behavior.

Let us examine a single period in greater detail by analyzing Figure 4 that shows the dynamics of relevant training metrics logged after each SGD step of ConvNet training. At the beginning of the period, the train loss is high, and the large gradients of the loss outweigh weight decay. This results in increasing weight norm and decreasing effective learning rate, i.e., we move along phase A of the phase diagram. SGD continues optimizing train loss, and at some point, train loss and its gradients become small and outweighed by weight decay. As a result, the weight norm starts decreasing, and the effective learning rate increases, i.e., we move along phase B of the phase diagram. We note
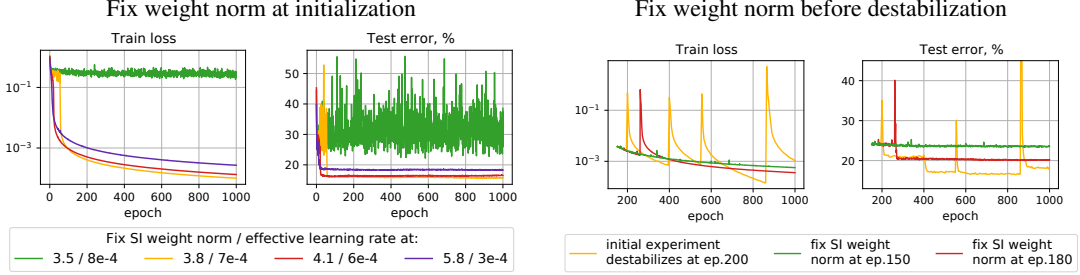
Figure 5: The absence of the periodic behavior for training with the fixed weight norm. Scale-invariant ConvNet on CIFAR-10 trained using SGD with weight decay of 0.001 and learning rate of 0.01. Left pair: the weight norm is fixed at random initialization of different scales. Right pair: the weight norm is fixed at some epoch of regular training before destabilization.

that the transition between phases A and B correlates with achieving near-zero train error. When the weight norm becomes too small, and the effective learning rate becomes too high, SGD makes several large steps and leaves the low loss region. Gradients grow along with train loss and, multiplied by a high effective learning rate, lead to the fast growth of the weight norm, i.e., we move along phase C of the phase diagram. The detailed plot of phase C in Figure 4 confirms that train loss starts increasing earlier than the weight norm. When the weight norm becomes large, the effective learning rate becomes low and stops the process of divergence. After that, a new period of training begins.

We also conducted an ablation experiment to show that the discovered periodic behavior is indeed a result of the competing influence of BN and WD on the weight norm. To do so, we prohibit this influence and train the network on a sphere by fixing the weight norm and rescaling the weights after each SGD step. We firstly fix the weight norm at random initialization, considering different values of the initialization weight norm and hence different (fixed) effective learning rates. Figure 5 (left pair) shows that in this case, there is no periodic behavior, and the train loss either converges (for relatively low effective learning rates) or gets stuck at high values (for high effective learning rates). We repeat this ablation fixing the weight norm at some epoch preceding destabilization in the experiment where we observe the periodic behavior. Specifically, as an initial experiment, we use the one with the learning rate of 0.01 from Figure 3 and fix the weight norm at the 150-th and 180-th epochs, preceding the destabilization at epoch 200. Figure 5 (right pair) shows the absence of the periodic behavior in both cases. When we fix the weight norm closer to the destabilization at the 180-th epoch, we observe a single increase in train loss, as the training process has already become unstable. However, after converging from this increase, train loss never destabilizes again.

5 Theoretical grounding for periodic behavior

In this section, we theoretically investigate the reasons for the training destabilization between phases B and C, and after that, we generalize the overall training process equilibrium condition of Li et al. [19] taking into account the discovered periodic behavior. To do so, we study the optimization dynamics of an arbitrary scale-invariant function $f(x)$ trained using (S)GD with learning rate η and weight decay of strength λ (2). Hereinafter, we will assume that the $\eta\lambda$ product is small, i.e., we can suppress $\mathcal{O}\left((\eta\lambda)^2\right)$ terms. We also refer to Appendix A for the proofs, derivations, and further discussion on our theoretical results.

We recall that (stochastic) gradients of an arbitrary scale-invariant function $f(x)$ possess two fundamental properties (1a) and (1b). Based on these properties, we obtain the dynamics of the parameters norm induced by Eq. (2) which we also leverage in our analysis (derivation of this and other equations is deferred to Appendix A.2):

$$\rho_{t+1}^2 = (1 - \eta\lambda)^2 \rho_t^2 + \eta^2 \tilde{g}_t^2 / \rho_t^2, \tag{3}$$

where $\rho_t = \|x_t\|$ denotes the parameters norm, $g_t = \|\nabla f(x_t)\|$ — the gradient norm, $\tilde{g}_t = \|\nabla f(x_t/\|x_t\|)\| = \rho_t g_t$ — the effective gradient norm. In this work, we also use the notion of effective learning rate which is formally defined as $\tilde{\eta}_t = \eta/\rho_t^2$.

5

5.1 The notion of δ-jumps

As scale-invariant functions are essentially defined on a sphere, cosine distance is a natural choice for a metric in parameter space. The following notion defines a situation when adjacent iterates become distant from each other, indicating training destabilization.

Definition 1 *We say that dynamics* (2) *performed a* **δ-jump** *once the cosine distance between adjacent iterates exceeds some value $\delta > 0$:*

$$1 - \cos(x_t, x_{t+1}) > \delta.$$

We conjecture that *the necessary condition for the training dynamics' destabilization is performing δ-jumps with sensible values of δ*. Otherwise, as long as adjacent iterates remain too close, the model (and hence its training dynamics) cannot change significantly. This holds strictly, for instance, if the Lipschitz constant of f is bounded (at least locally), which can be relevant for neural networks with BN [22]. But even if f has unstable regions on a unit sphere with very high or even unbounded Lipschitz constant, our analysis is still relevant since making larger steps in such regions would lead to a higher chance of divergence. Further, we show that the closer we approach the origin, the larger effective steps (steps on a unit sphere) we start making, thereby paving the way for destabilization.

Now, our question is, *given the value δ, what are the conditions for a δ-jump to occur?* By assuming that effective gradients are bounded, i.e., we can set two values $0 \leq \ell \leq L < +\infty$ such that $\tilde{g}_t \in [\ell, L]$, we answer this question in the following proposition. Proof can be seen in Appendix A.3.

Proposition 1 *If $f(x)$ is a scale-invariant function optimized according to dynamics* (2) *with bounded effective gradients $0 \leq \ell \leq \tilde{g}_t \leq L < +\infty$, then for sufficiently small δ and assuming $(1 - \eta\lambda) \lessapprox 1$, the following approximate conditions on δ-jump hold:*

$$\begin{cases} \rho_t^2 \lessapprox \dfrac{\eta L}{\sqrt{2\delta}} \implies \delta\text{-jump is possible}, & \text{(4a)} \\[3mm] \rho_t^2 \lessapprox \dfrac{\eta\ell}{\sqrt{2\delta}} \implies \delta\text{-jump is guaranteed}. & \text{(4b)} \end{cases}$$

Remark 1 *Our results hold for any values ℓ, L bounding the effective gradient norm, but the tighter these bounds are, the more precisely our theory describes the properties of the actual dynamics, thus we generally assume that ℓ and L are taken as local bounds on $\tilde{g}_t$ valid for several current iterations.*

To connect our theoretical results with practice, we examine the behavior of effective steps length of a scale-invariant neural network, compare it with theoretical bounds and observe gradually increasing destabilization of training dynamics. The rightmost plot of Figure 4 visualizes the cosine distance $1 - \cos(x_t, x_{t+1})$ between neural network's weights x_t and x_{t+1} at adjacent SGD steps for phase B of the training period. The dashed lines denote the theoretical upper and lower bounds on the cosine distance corresponding to the maximal and minimal δ-jumps derived from Eq. (4a) and (4b), respectively: $\delta_{\max} = \frac{\eta^2 L^2}{2\rho_t^4}$, $\delta_{\min} = \frac{\eta^2 \ell^2}{2\rho_t^4}$. To obtain those, we calculated the network's parameters norm ρ_t at each iteration and chose ℓ and L as smooth functions locally bounding the effective gradient norm in phase B (see Appendix D for details). We can see that both bounds, along with the measured cosine distance, start growing in the second half of the phase. This indicates that the performed δ-jumps are gradually increasing, hence instability accumulates until the training diverges. We note that such a long-lasting increase in cosine distance is common but, in general, not obligatory in the case of training with SGD because SGD may exhibit destabilization even with small δ-jumps due to stochasticity. For full-batch GD, this effect is even more prominent, see Appendix C.

Next, we formulate a proposition about how the initial parameters norm value ρ_0 and hyperparameters η and λ affect the time of δ-jumps occurrence and hence the frequency of the periods since training dynamics destabilization is closely connected with δ-jumps. Proof can be found in Appendix A.5. Note that ρ_0 should be interpreted as the norm at some initial moment $t = 0$ of a given period, when the conditions of the proposition are met, (typically, at the beginning of phase B) rather than the norm after initialization, i.e., at the very first iteration of training.

Proposition 2 *Denote* $\kappa = \sqrt{\frac{\eta}{2\lambda}}$. *Under the assumptions of Proposition 1:*

1. *if* $\rho_0^2 > \kappa\ell$ *and* $\delta < \eta\lambda\frac{L^2}{\ell^2}$, *then the **minimal** time required for the δ-jump to occur:*

$$t_{\min} = \max\left\{0, \frac{\log\left(\rho_0^2 - \kappa\ell\right) - \log\left(\frac{\eta L}{\sqrt{2\delta}} - \kappa\ell\right)}{-\log(1 - 4\eta\lambda)}\right\};\tag{5}$$

2. *if* $\rho_0^2 > \kappa L$ *and* $\delta < \eta\lambda\frac{\ell^2}{L^2}$, *then the **maximal** time required for the δ-jump to occur:*

$$t_{\max} = \max\left\{0, \frac{\log\left(\rho_0^2 - \kappa L\right) - \log\left(\frac{\eta\ell}{\sqrt{2\delta}} - \kappa L\right)}{-\log(1 - 2\eta\lambda)}\right\}.\tag{6}$$

Corollary 1 *Since both $t_{\max}$ and $t_{\min}$ are inversely proportional to $\eta\lambda$ as $-\log(1-\varepsilon) \approx \varepsilon$ for small ε, δ-jumps (and hence periods) must occur more often for larger values of $\eta\lambda$.*

5.2 Generalization of the equilibrium condition

We now generalize the equilibrium condition of Li et al. [19] and characterize the behavior of the parameters norm globally in the following proposition. The proof is provided in Appendix A.6.

Proposition 3 *Denote* $\kappa = \sqrt{\frac{\eta}{2\lambda}}$. *Under the assumptions of Proposition 1, if* $2\eta\lambda L \leq \ell$, *then*

$$\kappa\ell \leq \rho_t^2 \leq \kappa L, \ t \gg 1.\tag{7}$$

Furthermore, if $\rho_0^2 > \kappa L$, then ρ_t^2 converges linearly to $[\kappa\ell, \kappa L]$ interval in $\mathcal{O}\left(1/\eta\lambda\right)$ time.

Note that Li et al. [19] and Wan et al. [26] similarly predict that the equilibrium state can be reached in a linear rate regime. The condition $2\eta\lambda L \leq \ell$ is generally fulfilled in practice for small $\eta\lambda$ product even for globally chosen bounds ℓ, L. A similar assumption is made, e.g., in Theorem 1 in Wan et al. [26]. We discuss it in more detail (including the non-fulfillment case) in Appendix A.7.

Proposition 3 generalizes the results of Li et al. [19] who claimed that the effective learning rate $\tilde{\eta}_t = \eta/\rho_t^2$ converges to a constant. Their derivation relies on the assumption of stabilization of the effective gradient variance, which contradicts the observed periodic behavior. We relax this assumption by putting bounds on the effective gradient norm, thus bounding the parameters norm limits. These bounds can be either local, which defines the local trend of parameters norm dynamics, or global, which describes its general behavior. Also, note that we, in some sense, extend the results of Wan et al. [26] as we provide the exact limiting interval for ρ_t^2, not just bound its variance.

6 Empirical study of the periodic behavior

After discussing the reasons for the occurrence of the periodic behavior, we now further analyze its properties. In particular, we investigate: how hyperparameters affect the periodic behavior, how the periodic behavior evolves over epochs, and how minima achieved in different training periods differ both in parameter and functional space. In this section, we again consider the simplified setting with a fully scale-invariant neural network trained with standard SGD.

Influence of hyperparameters. We investigate the influence of two key training hyperparameters: learning rate and weight decay, but since the training dynamics mainly depend on their product (see discussion in Section 3 and Appendix F.1), we only vary the learning rate. The results for ConvNet on CIFAR-10 are given in Figure 3, the results for other dataset-architecture pairs are presented in Appendix F.2, and the results on the variable weight decay are given in Appendix F.3. Our first observation is that with higher learning rates, consistent periodic behavior occurs at larger weight norms. This is because SGD with a high learning rate can only converge with relatively small gradient norms, which are achieved at large weight norms according to Eq. (1b). This observation also agrees with Proposition 3 in Section 5. The second observation is that the periodic behavior is present for a wide range of learning rates, e.g., $0.003 - 0.3$ for ConvNet on CIFAR-10, and the higher the learning

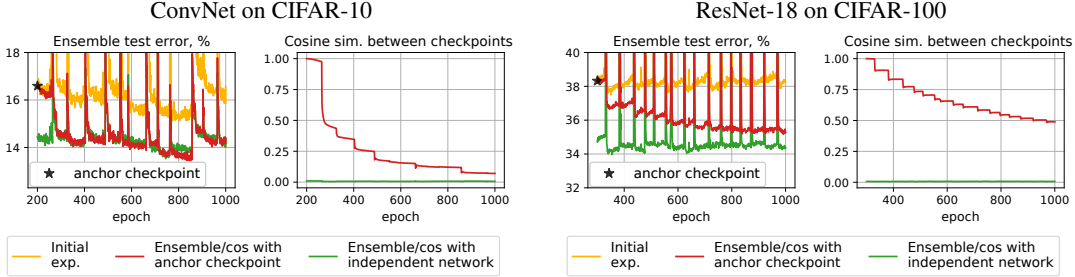ConvNet on CIFAR-10   ResNet-18 on CIFAR-100

Figure 6: Similarity in the weight space (cosine sim.) and the functional space (ensemble test error) for different checkpoints of training scale-invariant ConvNet on CIFAR-10 (left pair) and ResNet on CIFAR-100 (right pair) using SGD with weight decay of 0.001 and learning rate of 0.03.

rate, the shorter the periods, which agrees with Corollary 1 in Section 5. When using a learning rate that is too high, e.g., 1 in Figure 3, we expect training to yield very large weight norms, however weight decay prohibits us from reaching them, thus the gradients are not able to shrink sufficiently, and training gets stuck at high train loss. On the other hand, using a learning rate that is too low, e.g., 0.001 in Figure 3, leads to prolonged training which does not reach a small enough weight norm to yield a high effective learning rate, resulting in the absence of the periodic behavior in the given number of epochs. We note that the periodic behavior is present for the learning rates giving the lowest test error. In Appendix F.3, we show that varying weight decay leads to similar effects: the periodic behavior is present for a wide range of reasonable weight decays but is absent for too low or too high weight decays, and the higher the weight decay, the faster the periods.

Dynamics of periodic behavior. We now analyze how the discovered periodic behavior evolves over epochs. As discussed in the previous paragraph, consistent periodic behavior occurs at larger weight norms with higher learning rates. However, the initialization may have a substantially different norm. Thus we observe a *warm-up stage* in some plots of Figures 1 and 3, when the beginning of training is spent on moving towards the appropriate norm of scale-invariant weights. Expectedly, this warm-up stage is more prolonged for lower learning rates. Reaching the proper weight norm initiates a consistent periodic behavior. During the warm-up stage, SGD can still exhibit regular destabilization happening at higher weight norms than in the stage of consistent periodic behavior. We hypothesize that at the early stage of training, SGD converges to less stable basins with larger effective gradients, in which destabilization happens at larger norms of the scale-invariant parameters. We notice that test error decreases after each warm-up destabilization episode and reaches a lower level than training with a fixed effective learning rate, as shown in Figure 5 (right pair). In other words, the performance may benefit from the repeating destabilization.

Minima achieved at different training periods. Next, we aim at understanding whether minima achieved in different training periods are close in weight and functional spaces. We use the cosine similarity function for the weight space and estimate similarity in the functional space by comparing with ensembles of independent models, following Fort et al. [8]. If training process converges to the same minimum in each period, then cosine similarity between two minima achieved in different periods should be close to one and their ensemble error should be close to the error of a single network. On the contrary, if destabilization moves training so far that it is equivalent to retraining a model from a new random initialization, then the cosine similarity between the two minima should be close to zero and their ensemble error should be close to the error of an ensemble of two independently trained networks.

The setup of the experiment is as follows. We select some initial experiment and its checkpoint (called anchor checkpoint) corresponding to the minimum achieved when the training process has already converged to the consistent periodic behavior. After that, we measure weight/function similarities between the anchor checkpoint and all the subsequent checkpoints of the initial experiment — this is our primary measurement. For comparison, we independently train one more neural network with the same hyperparameters as in the initial experiment but from a different random initialization, select its checkpoint with the same test error as that of the anchor checkpoint, and measure the similarity between this new checkpoint and the checkpoints of the original experiment.
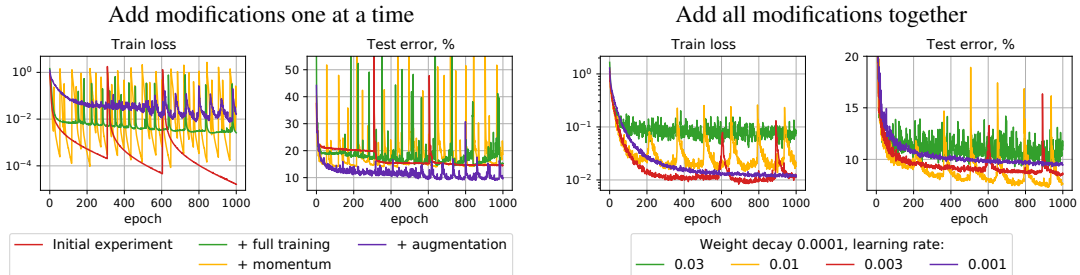
8

Figure 7: Periodic behavior of ConvNet (of increased width) on CIFAR-10 trained with the more practical modifications. Left: weight decay of 0.001, learning rate of 0.01.

The results for ConvNet on CIFAR-10 and ResNet-18 on CIFAR-100 are presented in Figure 6, the results for other dataset-architecture pairs are given in Appendix H. Inside one training period, checkpoints do not step far from the anchor checkpoint, i.e., the cosine similarity is close to one, and the ensemble test error is close to the error of a single network. However, when the next training period begins after destabilization, SGD moves to another region in the weight space, and both similarities start decreasing: the cosine similarity drops, and ensemble test error becomes smaller than that of a single network. Each following training period moves SGD farther away from the anchor checkpoint. For networks on CIFAR-100, late training periods continue to be correlated with the anchor checkpoint, i.e., the cosine similarity only reaches ~ 0.5 value, and ensemble test error does not reach the level of the independent networks ensemble. Still, both similarities continue decreasing. For networks on CIFAR-10, the cosine similarity decreases faster, and the ensemble test error quickly reaches the test error of an ensemble of two independently trained networks. To sum up, minima achieved at two neighboring training periods are substantially different, but their similarity is usually higher than that of two independently trained networks.

7 Periodic behavior in a practical setting

In the previous sections, we conducted experiments with scale-invariant neural networks trained with the simplest version of SGD. This allowed us to analyze the periodic behavior of train loss in detail. However, in practice, a portion of the weights of a neural network are not scale-invariant, e.g., the weights of the last layer and learnable BN parameters. Furthermore, networks are trained using more advanced procedures, e.g., SGD with momentum, data augmentation, and a learning rate schedule. At the same time, periodic behavior was mainly not noticed in previous works to the best of our knowledge. In this section, we show the presence of the periodic behavior for standard neural networks trained with momentum and data augmentation and discuss why periodic behavior may be not observed in practice. In Appendix I, we also show the presence of the periodic behavior for the networks with other normalization approaches or trained with Adam [13].

We select one of our initial experiments and add modifications one at a time to see their effects more clearly. We also present the results for training with all modifications turned on together. The plots for ConvNet on CIFAR-10 are given in Figure 7 and for other setups — in Appendix I. In this section, we use a wider ConvNet, as the standard version is too small to learn the augmented dataset.

Training non-scale-invariant weights. To achieve full scale-invariance, we froze the weights of the last layer and the parameters of BN layers since they all are not scale-invariant. We now consider the conventional procedure that implies training all neural network weights. In addition to the results presented in Figure 7 (left), we refer the reader to Figure 1. We observe that training non-scale-invariant weights retains the periodic behavior and affects the frequency of periods. The last-mentioned effect relates to the trainable last layer that automatically adjusts prediction confidence. In Appendix G, we show that variable prediction confidence results in different periodic behavior.

SGD with momentum. Next, we investigate the effect of using a more complex optimization algorithm. We consider SGD with momentum as the algorithm most commonly used for training convolutional neural networks. We observe that using momentum does not break the periodic

behavior and increases the frequency of periods. This agrees with the commonly observed effect that momentum speeds up training [23], i.e., momentum accelerates phases A and B. Interestingly, momentum does not prevent destabilization.

Data augmentation. We next consider training on the dataset with standard CIFAR-10 data augmentations, see details in Appendix B. Augmentation prevents over-fitting to the training data, which results in less confident predictions and larger train loss. As a result, train loss gradients outweigh WD more easily. If the number of parameters in the neural network is insufficient to achieve low train loss gradients, phase A never ends (at least in 1000 epochs), resulting in the absence of the periodic behavior. On the other hand, a sufficiently large neural network learns the augmented dataset at some epoch and proceeds to phase B, launching the periodic process. Still, we note that the periodic behavior begins much later than for the network trained without augmentation. This is one of the main reasons why the periodic behavior is often not observed in practice: it requires a much larger number of epochs to start than conventionally used for training.

All modifications together. In the two right plots of Figure 7, we visualize training with momentum, data augmentation, and unfrozen non-scale-invariant parameters used simultaneously and observe the presence of the periodic behavior.

So, what factors do interfere with observing the periodic behavior in practice? We underline two main factors. First, the interplay between different modifications narrows the range of hyperparameter values for which periodic behavior is present. When non-scale-invariant parameters are trained, the model converges to low test error only with specific values of weight decay. Moreover, with data augmentation, periodic behavior occurs only with relatively high learning rates (with lower learning rates, the training is too slow to reach phase C in 1000 epochs), while with momentum, using too high learning rates may result in training failure in phase A. In sum, periodic behavior appears only for a limited range of hyperparameters. Despite that, we note that the model generally achieves its best performance exactly in this range. Second, practical settings also imply learning rate schedules and a relatively small number of epochs, which do not preserve periodic behavior. We provide further discussion on comparison of our experimental setup with other works in Appendix J.

8   Conclusion

In this work, we described the periodic behavior of neural network training with batch normalization and weight decay occuring due to their competing influence on the norm of the scale-invariant weights. The discovered periodic behavior clarifies the contradiction between the equilibrium and instability presumptions regarding training with BN and WD and generalizes both points of view. In our empirical study, we investigated what factors and in what fashion influence the discovered periodic behavior. In our theoretical study, we introduced the notion of δ-jumps to describe training destabilization, the cornerstone of the periodic behavior, and generalized the equilibrium conditions in a way that better describes the empirical observations.

Limitations and negative societal impact. We discuss only conventional training of convolutional neural networks for image classification and do not consider other architectures and tasks. However, we believe that our findings extrapolate to training any kind of neural network with some type of normalization and weight decay. We also focus on a particular source of instability induced by BN and WD, yet, other factors may make training unstable [7]. This is an exciting direction for future research. To the best of our knowledge, our work does not have any direct negative societal impact. However, while conducting the study, we had to spend many GPU hours, which, unfortunately, could negatively affect the environment.

References

[1] Arora, S., Li, Z., and Lyu, K. (2019). Theoretical analysis of auto rate-tuning by batch normalization. In *International Conference on Learning Representations*.

[2] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[3] Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in Neural Information Processing Systems*, 31.

[4] Carmon, Y., Duchi, J., Hinder, O., and Sidford, A. (2017). Lower bounds for finding stationary points ii: First-order methods. *Mathematical Programming*, 185.

[5] Chiley, V., Sharapov, I., Kosson, A., Koster, U., Reece, R., Samaniego de la Fuente, S., Subbiah, V., and James, M. (2019). Online normalization for training neural networks. *Advances in Neural Information Processing Systems*, 32.

[6] Cho, M. and Lee, J. (2017). Riemannian approach to batch normalization. *Advances in Neural Information Processing Systems*, 30.

[7] Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. (2021). Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*.

[8] Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

[9] Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR.

[10] Hoffer, E., Banner, R., Golan, I., and Soudry, D. (2018a). Norm matters: efficient and accurate normalization schemes in deep networks. *Advances in Neural Information Processing Systems*, 31.

[11] Hoffer, E., Hubara, I., and Soudry, D. (2018b). Fix your classifier: the marginal value of training the last weight layer. In *International Conference on Learning Representations*.

[12] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[13] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[14] Kostenetskiy, P. S., Chulkevich, R. A., and Kozyrev, V. I. (2021). HPC resources of the higher school of economics. *Journal of Physics: Conference Series*, 1740:012050.

[15] Krizhevsky, A., Nair, V., and Hinton, G. CIFAR-10 (canadian institute for advanced research).

[16] Krizhevsky, A., Nair, V., and Hinton, G. CIFAR-100 (canadian institute for advanced research).

[17] Li, X., Chen, S., and Yang, J. (2020a). Understanding the disharmony between weight normalization family and weight decay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4715–4722.

[18] Li, Z. and Arora, S. (2020). An exponential learning rate schedule for deep learning. In *International Conference on Learning Representations*.

[19] Li, Z., Lyu, K., and Arora, S. (2020b). Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *Advances in Neural Information Processing Systems*, 33.

[20] Roburin, S., de Mont-Marin, Y., Bursuc, A., Marlet, R., Pérez, P., and Aubry, M. (2020). A spherical analysis of adam with batch normalization. *arXiv preprint arXiv:2006.13382*.

[21] Salimans, T. and Kingma, D. P. (2016). Weight normalization: a simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 29.

[22] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31.

[23] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

[24] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

[25] Van Laarhoven, T. (2017). L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*.

[26] Wan, R., Zhu, Z., Zhang, X., and Sun, J. (2020). Spherical motion dynamics: Learning dynamics of neural network with normalization, weight decay, and sgd. *arXiv preprint arXiv:2006.08419*.

[27] Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

[28] Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2019). A mean field theory of batch normalization. In *International Conference on Learning Representations*.

[29] Zhang, G., Wang, C., Xu, B., and Grosse, R. (2019). Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*.

A Theoretical results

This section contains details on our theoretical results.

A.1 Invariance to hyperparameters rescaling

Based on properties (1a) and (1b), we derive a simple yet useful proposition tying together different hyperparameter settings of initialization x_0, learning rate η, and weight decay coefficient λ. This proposition provides grounds for fixing the initialization scale in our experiments and iterating over learning rates and weight decay coefficients when studying the dependence of the behavior of scale-invariant neural networks on hyperparameters.

Proposition 4 *Given $f(x)$ is scale-invariant and optimized according to Eq. (2), settings (x_0, η, λ) and $(x_0', \eta', \lambda') = (cx_0, c^2\eta, \lambda/c^2)$, $c > 0$ lead to equivalent dynamics in function space.*

Proof. Eq. (2) and property (1b) give $x_{t+1} = \|x_t\| \left[(1 - \eta\lambda)\frac{x_t}{\|x_t\|} - \tilde{\eta}_t \nabla f(x_t / \|x_t\|) \right]$, where $\tilde{\eta}_t = \frac{\eta}{\|x_t\|^2}$ is the effective learning rate. Since the term in square brackets does not depend on the scale of x_t provided that the effective learning rate and $\eta\lambda$ product are unchanged, by induction, from $x_t' = cx_t$ we have $x_{t+1}' = cx_{t+1}$, hence $f(x_{t+1}') = f(x_{t+1})$. ∎

A.2 Derivations

Parameters norm dynamics (3)

$$\begin{aligned}
\rho_{t+1}^2 &= \langle x_{t+1}, x_{t+1} \rangle = \{\text{Eq. (2)}\} = (1 - \eta\lambda)^2 \rho_t^2 + \eta^2 g_t^2 + 2\eta(1 - \eta\lambda)\langle \nabla f(x_t), x_t \rangle = \\
&= \{\text{property (1a)}\} = (1 - \eta\lambda)^2 \rho_t^2 + \eta^2 g_t^2 = \{\text{property (1b), i.e., } g_t = \tilde{g}_t / \rho_t\} = \\
&= (1 - \eta\lambda)^2 \rho_t^2 + \eta^2 \tilde{g}_t^2 / \rho_t^2
\end{aligned}$$

Cosine distance between adjacent iterates (8)

$$\begin{aligned}
\cos(x_t, x_{t+1}) &= \frac{\langle x_t, x_{t+1} \rangle}{\rho_t \rho_{t+1}} = \{\text{Eq. (2)}\} = \frac{(1 - \eta\lambda)\langle x_t, x_t \rangle - \eta \langle \nabla f(x_t), x_t \rangle}{\rho_t \rho_{t+1}} = \\
&= \{\text{property (1a)}\} = \frac{(1 - \eta\lambda)\rho_t}{\rho_{t+1}} = \{\text{Eq. (3)}\} = \left(1 + \frac{\eta^2 \tilde{g}_t^2}{(1 - \eta\lambda)^2 \rho_t^4} \right)^{-1/2}
\end{aligned}$$

δ-jump conditions (9)

$$1 - \cos(x_t, x_{t+1}) > \delta \iff \{\text{Eq. (8)}\} \iff \left(1 + \frac{\eta^2 \tilde{g}_t^2}{(1 - \eta\lambda)^2 \rho_t^4} \right)^{-1/2} < 1 - \delta \iff$$

$$\iff 1 + \frac{\eta^2 \tilde{g}_t^2}{(1 - \eta\lambda)^2 \rho_t^4} > \frac{1}{(1 - \delta)^2} = 1 + 2\delta + \mathcal{O}(\delta^2) \gtrapprox 1 + 2\delta.$$

Omitting $\mathcal{O}(\delta^2)$ leaves the condition necessary and also approximately sufficient for small δ:

$$1 - \cos(x_t, x_{t+1}) > \delta \implies \frac{\eta^2 \tilde{g}_t^2}{(1 - \eta\lambda)^2 \rho_t^4} > 2\delta \iff \rho_t^2 < \frac{\eta \tilde{g}_t}{(1 - \eta\lambda)\sqrt{2\delta}}.$$

A.3 Proof of Proposition 1

For the convenience of reading, we defer the derivation details of all equations to Appendix A.2.

Proof. Using property (1a) and Eq. (3), we obtain the exact value of the cosine between adjacent iterates:

$$\cos(x_t, x_{t+1}) = \left(1 + \frac{\eta^2 \tilde{g}_t^2}{(1 - \eta\lambda)^2 \rho_t^4} \right)^{-1/2}. \tag{8}$$

From Eq. (8) we deduce the following δ-jump condition:

$$1 - \cos(x_t, x_{t+1}) > \delta \implies \rho_t^2 < \frac{\eta \tilde{g}_t}{(1 - \eta\lambda)\sqrt{2\delta}}. \tag{9}$$

During the derivation, we omitted $\mathcal{O}(\delta^2)$ terms. This implies that the right inequality represents not only the necessary but also (approximately) the sufficient condition for a δ-jump when δ is small.

Assuming $(1 - \eta\lambda) \lesssim 1$ and substituting the effective gradient bounds ℓ and L into Eq. (9) in place of $\tilde{g}_t$ finally yields the approximate necessary and sufficient δ-jump conditions (4a) and (4b), respectively. ∎

A.4 On β-undetermined recurrent sequences

Here we provide some results related to convergence of sequences of the following kind:

$$x_{t+1} = (1 - \alpha)x_t + \frac{\beta_t}{x_t}, \tag{10}$$

where α is a fixed coefficient, and β_t may vary from iteration to iteration. We assume $x_0 > 0$, $0 < \alpha < 0.5$, and $\beta_t \in [a, b]$, $\forall t$, where $0 \le a \le b < +\infty$ are some fixed values. We call sequences of type (10) β-*undetermined* recurrent sequences.

A.4.1 β-determined sequences

To derive the basic properties of β-undetermined sequences (10), we first consider β-*determined* recurrent sequences:

$$x_{t+1} = (1 - \alpha)x_t + \frac{\beta}{x_t}, \tag{11}$$

where β is now a fixed non-negative value.

If $\beta = 0$, (11) boils down to a classical linear sequence converging to zero at rate $1 - \alpha$. Assume now that $\beta > 0$. First of all, $x^* = \sqrt{\frac{\beta}{\alpha}}$ is the only stationary point of sequence (11). This holds from solving the following equation:

$$x_{t+1} = x_t \iff x_t = x^* = \sqrt{\frac{\beta}{\alpha}}.$$

Suppose $x_t = \gamma_t x^*$. By dividing the left and right sides of Eq. (11) by x^*, we can derive the formula for γ_{t+1} as a function of γ_t which we denote as $\varphi(\gamma_t)$:

$$\gamma_{t+1} = \varphi(\gamma_t) = (1 - \alpha)\gamma_t + \frac{\alpha}{\gamma_t}. \tag{12}$$

The sequence induced by (12) is a special case of Eq. (11) with a stationary point $\gamma^* = 1$. One important property is that γ_{t+1} does not depend on β explicitly, only on γ_t and α. This unifies the convergence analysis for sequences with different β coefficients.

For function (12) the following facts hold (see Figure 8 for an illustration):

$$
\begin{cases}
\gamma_t < 1 \implies \gamma_{t+1} > \gamma_t\text{: the sequence is increasing once it's below } x^*, & \text{(13a)} \\[4pt]
\gamma_t > 1 \implies 1 < \gamma_{t+1} < \gamma_t\text{: the sequence is decreasing once it's above } x^*, & \text{(13b)} \\[4pt]
\gamma_{t+1} = 1 \iff \gamma_t = \dfrac{\alpha}{1-\alpha} \vee \gamma_t = 1\text{: pre-stationary conditions}, & \text{(13c)} \\[4pt]
\gamma_{t+1} < 1 \iff \gamma_t \in \left(\dfrac{\alpha}{1-\alpha}, 1\right)\text{: conditions for staying below the stationary point}, & \text{(13d)} \\[4pt]
\varphi(\gamma_t) \text{ is a decreasing function for } \gamma_t < \sqrt{\dfrac{\alpha}{1-\alpha}}, & \text{(13e)} \\[4pt]
\varphi(\gamma_t) \text{ is an increasing function for } \gamma_t > \sqrt{\dfrac{\alpha}{1-\alpha}}, & \text{(13f)} \\[4pt]
\gamma_{t+1} = \min_{\gamma_t} \varphi(\gamma_t) = 2\sqrt{\alpha(1-\alpha)} \iff \gamma_t = \sqrt{\dfrac{\alpha}{1-\alpha}}\text{: the lowest achievable value.} & \text{(13g)}
\end{cases}
$$

Note that for $0 < \alpha < 0.5$ we have

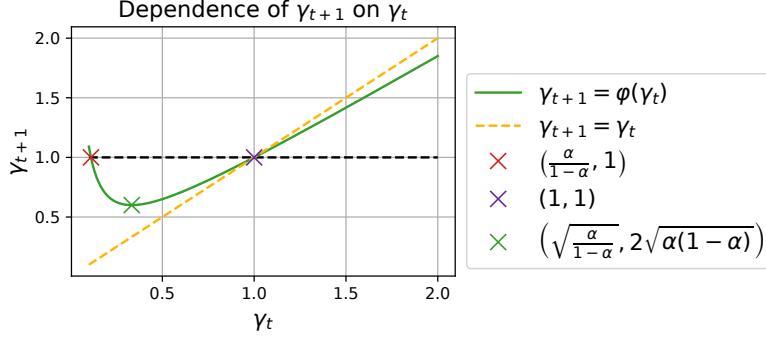$$\frac{\alpha}{1-\alpha} < \sqrt{\frac{\alpha}{1-\alpha}} < 2\sqrt{\alpha(1-\alpha)} < 1.$$

14

Figure 8: Dependence of γ_{t+1} on γ_t from Eq. (12) for $\alpha = 0.1$.

Properties (13b) and (13d) imply that x_{t+1} can "hop" over x^* if only $x_t < \frac{\alpha}{1-\alpha} x^*$. Otherwise, x_t is monotonically approaching its stationary point. That is an important threshold that will help derive the convergence of β-undetermined sequences to a specific equilibrium interval.

The derivative of $\varphi(\gamma_t)$ can help estimate the convergence rate of the sequence (11) to its stationary point. Specifically, using the mean value theorem, we obtain that

$$x_{t+1} - x^* = x^* (\gamma_{t+1} - 1) = x^* (\varphi(\gamma_t) - \varphi(1)) = x^* \varphi'(\xi) (\gamma_t - 1), \tag{14}$$

where ξ is some point between 1 and γ_t. Therefore, by bounding the derivative $\varphi'(\gamma_t)$, we can also bound the x_t convergence to x^*.

Suppose that $\gamma_0 > 1$. From (13b) it follows that $\gamma_t > 1$, $\forall t$. In this case, we can bound the derivative of $\varphi(\gamma_t)$ for $\gamma_t > 1$ and obtain the approximate convergence rates for (11):

$$1 - 2\alpha < \varphi'(\gamma_t) = (1 - \alpha) - \frac{\alpha}{\gamma_t^2} < 1 - \alpha, \, \gamma_t > 1,$$

which, after recursively applying (14), yields

$$(1 - 2\alpha)^t (\gamma_0 - 1) < \gamma_t - 1 < (1 - \alpha)^t (\gamma_0 - 1), \, t \geq 1,$$

or equivalently, formulating this for (11) as a lemma:

Lemma 1 *For an arbitrary β-determined sequence (11) with $\beta \geq 0$, given $x^* = \sqrt{\frac{\beta}{\alpha}}$ and $x_0 > x^*$, the following bounds on its convergence rate hold:*

$$(1 - 2\alpha)^t (x_0 - x^*) \leq x_t - x^* \leq (1 - \alpha)^t (x_0 - x^*), \, \forall t.$$

This is the main result concerning the convergence of β-determined sequences (11). Note that Lemma 1 also covers the case of $\beta = 0$ because then $x^* = 0$ and $x_t = (1 - \alpha)^t x_0$.

A.4.2 β-undetermined sequences convergence bounds

Now, we can return back to the β-undetermined sequences (10) and derive its convergence bounds. The following lemma allows to bound an arbitrary β-undetermined sequence with β-determined ones.

Lemma 2 *For an arbitrary β-undetermined sequence of type (10) with $0 \leq a \leq \beta_t \leq b < +\infty$ the following β-determined bounds hold.*

1. *Let $x_{a,t}$ be a β-determined sequence (11) with $\beta = a$ and $x_{a,0} = x_0$. Then $x_{a,t} \leq x_t$, $\forall t$.*

2. *Let $x_{b,t}$ be a β-determined sequence (11) with $\beta = b$ and $x_{b,0} = x_0$. Then, if $x_t > \sqrt{\frac{b}{1-\alpha}}$, $t = 0, \ldots, T$, we have $x_t \leq x_{b,t}$, $t = 0, \ldots, T + 1$.*

Proof. We will prove the first statement since the second one can be proved similarly.

15

Let $\sqrt{\frac{a}{1-\alpha}} < x_{a,t} \le x_t$. Then the following inequalities hold:

$$x_{a,t+1} \le (1-\alpha)x_t + \frac{a}{x_t} \le x_{t+1}.$$

The first inequality holds since $x_{a,t+1}$ is a monotonically increasing function of $x_{a,t}$ due to (13f). The second one is valid because $a \le \beta_t$.

Note that due to (13g) and $\sqrt{\frac{\alpha}{1-\alpha}} < 2\sqrt{\alpha(1-\alpha)}$, we have $\sqrt{\frac{a}{1-\alpha}} < x_{a,t}$, $t \ge 1$, plus, as $a \le \beta_0$, $x_{a,1} \le x_1$, hence, induction is valid for all t for the lower bound (in contrast with the upper bound case, where we explicitly demand $x_t > \sqrt{\frac{b}{1-\alpha}}$ for T consecutive timesteps). $\blacksquare$

Remark 2 *An important special case when the upper bound $x_{b,t}$ is valid for all t is if $\frac{\alpha}{1-\alpha}\sqrt{b} \le \sqrt{a}$ and $x_0 > \sqrt{\frac{b}{\alpha}}$. Then, while $x_t \ge \sqrt{\frac{b}{\alpha}} > \sqrt{\frac{b}{1-\alpha}}$ the bound is valid due to the second statement of the lemma. As soon as x_t crosses the $\sqrt{\frac{b}{\alpha}}$ threshold, it can never "hop" over it again due to (13d) and $x_t \ge x_{a,t} > \sqrt{\frac{a}{\alpha}} \ge \frac{\sqrt{\alpha b}}{1-\alpha}$, $\forall t$; at the same time, $x_{b,t} > \sqrt{\frac{b}{\alpha}}$, $\forall t$ due to (13b).*

Based on the convergence results of β-determined sequences, the following corollary allows estimating the convergence rates of β-undetermined sequences.

Corollary 2 *Given Lemma 1, Lemma 2, and the reasoning from Remark 2, we obtain the following bounds on convergence rates of an arbitrary β-undetermined sequence* (10)*:*

1. *if $x_0 > \sqrt{\frac{a}{\alpha}}$, then $(1-2\alpha)^t \left(x_0 - \sqrt{\frac{a}{\alpha}}\right) \le x_t - \sqrt{\frac{a}{\alpha}}$, $\forall t$;*

2. *if $x_0 > \sqrt{\frac{b}{\alpha}}$, then $x_t - \sqrt{\frac{b}{\alpha}} \le (1-\alpha)^t \left(x_0 - \sqrt{\frac{b}{\alpha}}\right)$ while $x_t \ge \frac{\sqrt{\alpha b}}{1-\alpha}$.*

Our final important result about the β-undetermined sequences convergence is a case of convergence to the interval determined by the stationary points of the bounding β-determined sequences $x_{a,t}$ and $x_{b,t}$. We formulate it in the following proposition (see Figure 9 for an illustration).

Proposition 5 *An arbitrary β-undetermined sequence* (10)*, given $\frac{\alpha}{1-\alpha}\sqrt{b} \le \sqrt{a}$, converges to the following interval:*

$$\sqrt{\frac{a}{\alpha}} \le x_t \le \sqrt{\frac{b}{\alpha}}, \ t \gg 1.^4$$

Furthermore, if $x_0 > \sqrt{\frac{b}{\alpha}}$, then x_t converges to the interval linearly in $\mathcal{O}(1/\alpha)$ time.

Proof. Due to the first statement of Lemma 2, $x_t \ge x_{a,t} \to \sqrt{\frac{a}{\alpha}}$, hence, we deduce that the lower bound will eventually hold for $t \to \infty$. Since $\frac{\alpha}{1-\alpha}\sqrt{b} \le \sqrt{a}$ and due to the reasoning in Remark 2, when $\sqrt{\frac{a}{\alpha}} \le x_t$ is fulfilled, the series either stays in the stated interval (if $x_t \le \sqrt{\frac{b}{\alpha}}$) and never crosses it or approaches it from above thanks to the upper β-deterministic bounding sequence $x_t \le x_{b,t} \to \sqrt{\frac{b}{\alpha}}$, so the upper bound is also (asymptotically) valid.

If $x_0 > \sqrt{\frac{b}{\alpha}}$, Corollary 2 allows us to enclose x_t (while it is above $\sqrt{\frac{b}{\alpha}}$) between two linear sequences converging to $\sqrt{\frac{a}{\alpha}}$ and $\sqrt{\frac{b}{\alpha}}$, respectively, with one-minus-rate proportional to α. This is consistent with the convergence time $\mathcal{O}(1/\alpha)$ since the convergence time of linear sequences is inversely proportional to the one-minus-rate value. $\blacksquare$

---

[4]These bounds are, in general, asymptotic, so, for complete correctness, $t \gg 1$ must be substituted with $t \to \infty$; however, excluding the degenerate cases, we can often observe that x_t reaches the interval in finite time.
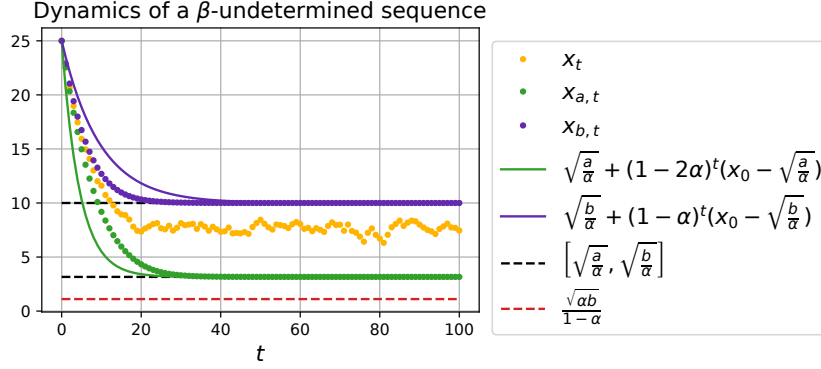
Figure 9: β-undetermined sequence (10) convergence to the $\left[\sqrt{\frac{a}{\alpha}}, \sqrt{\frac{b}{\alpha}}\right]$ interval (Proposition 5). Setting: $\alpha = 0.1$, $a = 1$, $b = 10$, $\beta_t \sim \mathcal{U}(a, b)$.

A.5 Proof of Proposition 2

We prove Proposition 2 using the general convergence theory for so-called β-undetermined recurrent sequences of type $x_{t+1} = (1 - \alpha)x_t + \frac{\beta_t}{x_t}$, where $0 < \alpha < 0.5$ and $0 \leq a \leq \beta_t \leq b < +\infty$, $\forall t$ (see Appendix A.4). Note that the parameters norm dynamics (3) is a special case of such a sequence with $x_t := \rho_t^2$, $\beta_t := \eta^2 \tilde{g}_t^2$, $a := \eta^2 \ell^2$, $b := \eta^2 L^2$, and $\alpha := 2\eta\lambda$ (recall that we suppress $\mathcal{O}\left((\eta\lambda)^2\right)$ terms).

Proof. Denote $\kappa = \sqrt{\frac{\eta}{2\lambda}}$.

In the notation of β-undetermined sequences, the condition $\rho_0^2 > \kappa\ell$ translates into $x_0 > \sqrt{\frac{a}{\alpha}}$. Thus, by applying Corollary 2, we can bound the convergence of parameters norm from below with the following linear sequence:

$$\kappa\ell + (1 - 4\eta\lambda)^t \left(\rho_0^2 - \kappa\ell\right) \leq \rho_t^2.$$

The necessary δ-jump condition (4a) can be equivalently reformulated as an upper bound on δ:

$$\kappa\ell < \frac{\eta L}{\sqrt{2\delta}} \iff \delta < \eta\lambda\frac{L^2}{\ell^2}.$$

If this condition is fulfilled, we can estimate the minimal time required for a δ-jump — the moment when the lower bound on ρ_t^2 intersects the $\frac{\eta L}{\sqrt{2\delta}}$ threshold. If $\rho_0^2 \leq \frac{\eta L}{\sqrt{2\delta}}$, obviously, $t_{\min} = 0$, else, by solving the following equation for t:

$$\sqrt{\frac{\eta}{2\lambda}}\ell + (1 - 4\eta\lambda)^t \left(\rho_0^2 - \sqrt{\frac{\eta}{2\lambda}}\ell\right) = \frac{\eta L}{\sqrt{2\delta}},$$

we obtain (5).

Again, $\rho_0^2 > \kappa L$ is equivalent to $x_0 > \sqrt{\frac{b}{\alpha}}$ and, due to Corollary 2, the following upper bound on ρ_t^2 holds (at least while $\rho_t^2 \geq \kappa L$):

$$\rho_t^2 \leq \kappa L + (1 - 2\eta\lambda)^t \left(\rho_0^2 - \kappa L\right).$$

Now, if δ is so small that the sufficient condition for a jump (4b) is fulfilled before ρ_t^2 converges to κL, i.e.,

$$\kappa L < \frac{\eta\ell}{\sqrt{2\delta}} \iff \delta < \eta\lambda\frac{\ell^2}{L^2},$$

we can similarly estimate the maximal required time for a δ-jump (6) as the moment when the upper bound on ρ_t^2 intersects the $\frac{\eta\ell}{\sqrt{2\delta}}$ threshold. ∎

A.6 Proof of Proposition 3

As in the previous section, we prove Proposition 3 using the general theory on β-undetermined sequences (see Appendix A.4). We remarked above that the parameters norm dynamics (3) is a special case of such a sequence with parameters $a := \eta^2\ell^2$, $b := \eta^2 L^2$, and $\alpha := 2\eta\lambda$.

Proof. According to Proposition 5, if for a β-undetermined sequence x_t the condition $\frac{\alpha}{1-\alpha}\sqrt{b} \leq \sqrt{a}$ is fulfilled, then one can show that $x_t \in \left[\sqrt{\frac{a}{\alpha}}, \sqrt{\frac{b}{\alpha}}\right]$, $t \gg 1$; furthermore, if $x_0 > \sqrt{\frac{b}{\alpha}}$, then x_t converges to the interval linearly in $\mathcal{O}(1/\alpha)$ time. For the parameters norm dynamics, the condition $\frac{\alpha}{1-\alpha}\sqrt{b} \leq \sqrt{a}$ is equivalent (up to $\mathcal{O}((\eta\lambda)^2)$ terms) to $2\eta\lambda L \leq \ell$ as $\frac{\alpha}{1-\alpha} = \frac{2\eta\lambda}{1-2\eta\lambda} = 2\eta\lambda + \mathcal{O}((\eta\lambda)^2)$. Now, if it holds, we can apply Proposition 5 and conclude the proof. ∎

Remark 3 *We can reformulate the same result in terms of the effective learning rate $\tilde{\eta}_t = \eta/\rho_t^2$:*

$$2\eta\lambda L \leq \ell \leq \tilde{g}_t \leq L \implies \frac{\sqrt{2\eta\lambda}}{L} \leq \tilde{\eta}_t \leq \frac{\sqrt{2\eta\lambda}}{\ell}, \; t \gg 1.$$

A.7 Discussion on $2\eta\lambda L \leq \ell$ condition

In this section, we discuss the assumption $2\eta\lambda L \leq \ell$ made in Proposition 3, implying that the lower and the upper effective gradient norm bounds must not differ too much. First of all, we would like to remark that this condition is generally fulfilled in practice for small $\eta\lambda$ product even when the bounds ℓ and L are taken globally, i.e., they satisfy $\ell \leq \tilde{g}_t \leq L$, $\forall t$. We also note that Wan et al. [26] made a very close assumption in their main Theorem 1 (Assumption 3). However, even if it is not fulfilled, our generalized parameters norm equilibrium result is still valid to some extent.

First, consider the case when $0 < \ell < 2\eta\lambda L$. Then, according to the general β-undetermined sequences theory presented in Appendix A.4, the lower bound $\kappa\ell \leq \rho_t^2$ remains valid for large t. If ρ_t^2 falls below $2\eta\lambda L$, it can potentially "hop" over the upper bound of the interval κL. However, due to $\tilde{g}_t \leq L$ and property (13e) of β-determined sequences (see Appendix A.4.1) ρ_t^2 is still upper bounded by the value $(1-\eta\lambda)^2\kappa\ell + \frac{\eta^2 L^2}{\kappa\ell}$. Hence, globally, the parameters norm stays bounded even when $2\eta\lambda L \leq \ell$ does not hold. Furthermore, according to the second statement of Corollary 2, once ρ_t^2 exceeds the κL value, it immediately starts converging to it again. So the same $[\kappa\ell, \kappa L]$ interval of attraction is still preserved.

Now, we argue that setting $\ell = 0$, i.e., bounding the effective gradient norm from below with zero, is vacuous.[5] Again, we remark that the assumption about separating ℓ from zero was made, e.g., by Wan et al. [26]. Arora et al. [1] show that effective gradients (in case of learning without WD) decay sublinearly, which by itself means that in finite time horizon, it is always reasonable to set $\ell > 0$. Moreover, as we show, parameters norm evolves linearly, i.e., faster than the effective gradients; therefore, it must quickly acclimate to local ℓ, L changes and hence respect the boundaries from Proposition 3. But even based on general results on gradient-based optimization, we anticipate that, in general, ℓ should not approach zero. We can rewrite the expression for ρ_t^2 (3) in the following way:

$$\rho_t^2 = (1-\eta\lambda)^2\rho_{t-1}^2 + \eta^2 g_{t-1}^2 = \cdots = (1-\eta\lambda)^{2t}\rho_0^2 + \eta^2\sum_{t'=0}^{t-1}(1-\eta\lambda)^{2(t-t'-1)}g_{t'}^2 = \quad (15)$$

$$= (1-\eta\lambda)^{2t}\rho_0^2 + \eta^2\frac{1-(1-\eta\lambda)^2}{1-(1-\eta\lambda)^{2t}}\bar{g}_t^2 \approx \{t \gg 1\} \approx (1-\eta\lambda)^{2t}\rho_0^2 + C\bar{g}_t^2, \quad (16)$$

where $\bar{g}_t$ is an exponential moving average of the gradient norm and $C = 2\eta^3\lambda + \mathcal{O}((\eta\lambda)^2)$ is constant. It is well-known that for first-ordered methods, the lower gradient norm bound generally decays sublinearly [4]. Note that the cosine between adjacent iterates (8) depends only on the g_t^2/ρ_t^2 ratio. For large t, this ratio, due to (16), is determined only by the $g_t^2/\bar{g}_t^2$ ratio since the first term decays linearly, i.e., faster than g_t^2. It is reasonable to conjecture that g_t oscillates around its mean value $\bar{g}_t$ hence hindering stabilization of the training dynamics which, in turn, implies that the effective gradient does not vanish. Thus, implying $\ell > 0$ seems to be a reasonable assumption.

---

[5]Excluding, perhaps, some exceptional degenerate cases when the function and hyperparameters are chosen so that the dynamics converge to a stationary point in a finite number of steps.
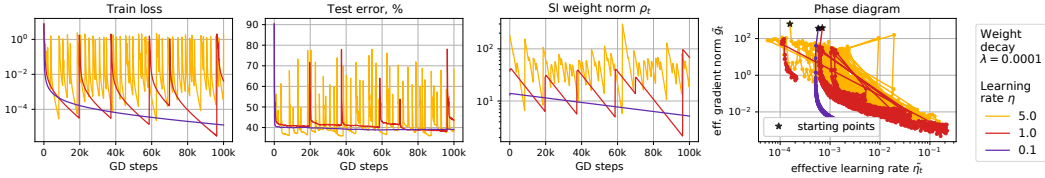
Figure 10: Periodic behavior of scale-invariant ConvNet on CIFAR-10 trained using full-batch GD with the weight decay of 0.0001 and different learning rates.

B Experimental details

Datasets and architectures. We conduct experiments with two convolutional architectures, namely a three-layer convolutional neural network (ConvNet) and ResNet-18, on CIFAR-10 [15] and CIFAR-100 [16] datasets. We use the implementation of both architectures available at `https://github.com/g-benton/hessian-eff-dim`. CIFAR datasets are distributed under the MIT license, and the code is under Apache-2.0 License. To make the majority of neural network weights scale-invariant, we insert additional BN layers according to Appendix C of Li and Arora [18]. We use the standard PyTorch initialization for all layers. We use ResNet of standard width. For ConvNet, we use the width factor of 32 for fully scale-invariant networks on CIFAR-10 and the width factor of 64 for all experiments on CIFAR-100 and experiments with practical modifications on CIFAR-10.

Fully scale-invariant setup. Most of the experiments are conducted with the scale-invariant modifications of both architectures obtained using the approach of Li and Arora [18]. In addition to inserting extra BN layers, we fix all non-scale-invariant weights, i.e., BN parameters and the last layer's parameters. For BN layers, we use zero mean and unit variance. We fix the bias vector at random initialization and the weight matrix at rescaled random initialization for the last layer. In most of the experiments, we rescale the last layer's weight matrix so that its norm equals 10, but we discuss other scales in Appendix G.

Training. We train all networks using SGD with a batch size of 128 and various weight decays and learning rates. In the experiments with momentum, we use the momentum of 0.9. In the experiments with data augmentation, we use standard CIFAR augmentations: random crop (size: 32, padding: 4) and random horizontal flip. All models were trained on NVidia Tesla V100 or NVidia GeForce GTX 1080. Obtaining the results reported in the paper took approximately 1K GPU hours.

Full-batch GD experiments. Full-batch GD training experiments are conducted on the 4.5K-sized random subset of the train dataset. The test set in this experiment consists of 5K randomly chosen test objects.

Logging. In all experiments except Figures 4, 11, and 13 we log all metrics after each epoch, computing train loss and its gradients by making an additional pass through the training dataset. We log all metrics after each (S)GD step in three specified figures, computing train loss and its gradients over a batch.

C Full-batch gradient descent

In the main paper, we presented the periodic behavior results for SGD. In this section, we show that the periodic behavior is observed for full-batch GD training and hence is not a consequence of stochastic training. We replicate all experiments of Section 4: Figure 10 visualizes training dynamics for different learning rate values, Figure 11 presents a closer look at one period of training (see also Figure 13 for the plots of cosines between adjacent steps), and Figure 12 replicates the ablation experiment with fixing the weight norm. All the effects discussed in the main text for the SGD case hold for the GD case. We note that phase B is longer for full-batch GD training because the absence of stochasticity allows stable training at lower train loss, and destabilization occurs later.
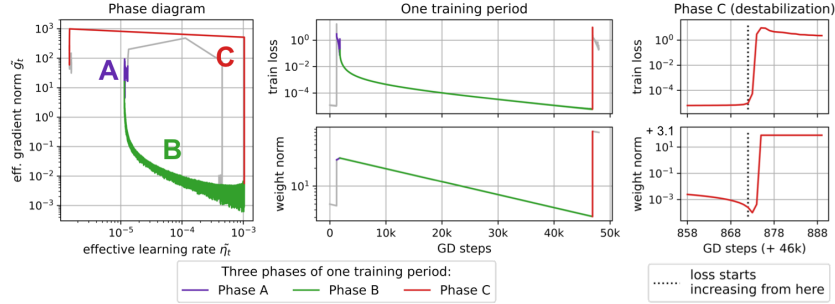
19

Figure 11: A closer look at one training period for scale-invariant ConvNet on CIFAR-10 trained using full-batch GD with weight decay of 0.001 and the learning rate of 0.5. Three phases of the training period are highlighted.
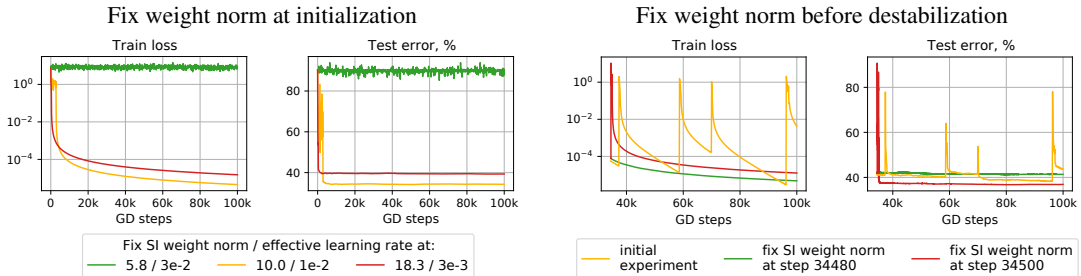

Figure 12: The absence of the periodic behavior for training with the fixed weight norm. Scale-invariant ConvNet on CIFAR-10 trained using full-batch GD with weight decay of 0.0001 and learning rate of 1.0. Left pair: the weight norm is fixed at random initialization of different scales. Right pair: the weight norm is fixed at some step of regular training before destabilization.

D   Bounds on the effective gradient norm and δ-jumps

In Section 4, we compared cosine distance between weights at adjacent SGD steps of phase B with theoretically derived bounds for δ-jumps from Section 5.1. In Figure 13, right pair, we present a similar comparison for the full-batch GD case: the effect of both bounds and the cosine metric itself growing in the second half of the phase is even more prominent for the GD case than for SGD. Below we describe how we choose the local bounds ℓ and L on the effective gradient norm $\tilde{g}_t$ which are used in the theoretical bounds. All bounds are visualized in Figure 13.

In both GD and SGD cases, we chose $\ell(t)$ and $L(t)$ as smooth functions of t. Note that taking such dynamical bounds does not contradict our theoretical results (see Remark 1). For the SGD case, we chose $\ell(t) = \frac{c}{t-t_0}$ and $L(t) = \frac{C}{t-t_0}$, where t_0 is the first iteration of the considered training period. For the GD case we used the same approach, but had to take $\ell(t) = \frac{c}{(t-t_0)^2}$ to better mimic the behavior of the lower envelope of the effective gradients norm. We handpick constants $0 < c < C$ and iteration t_{valid} separately for SGD and GD cases so that

$$\ell(t) \leq \tilde{g}_t \leq L(t) \tag{17}$$

for all $t \geqslant t_{\text{valid}}$ in phase B.

E   Optimization of common scale-invariant functions with weight decay

In this section, we show that periodic behavior may be observed not only when training neural networks but also during gradient decent optimization of common scale-invariant functions with weight decay and a constant learning rate. As an example we consider a function of two variables $f(x, y) = \frac{x^2}{x^2+y^2}$, which is naturally scale-invariant. The minimum value of f equals 0 and is achieved at any point with $x = 0$.
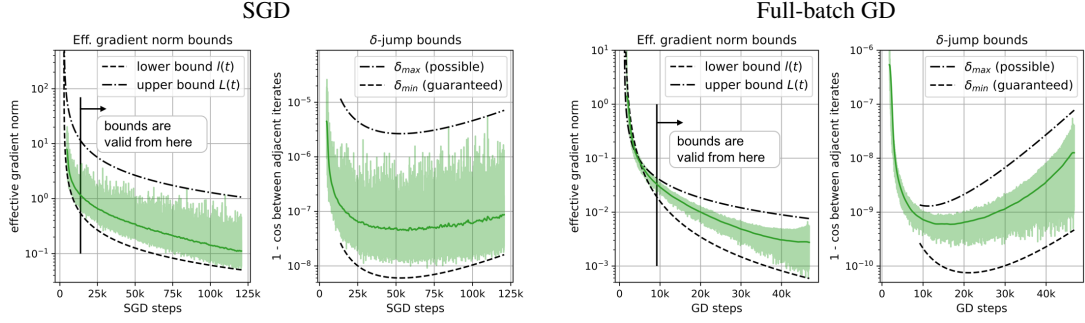
Figure 13: Effective gradient norm and cosine distance between weights at adjacent (S)GD steps, presented along with their smoothed trends. Phase B of one period of training scale-invariant ConvNet on CIFAR-10 is shown. Weight decay / learning rate: 0.001 / 0.01 for SGD, 0.0001 / 0.5 for GD. δ-jump bounds are obtained using the bounds on the effective gradient norm.
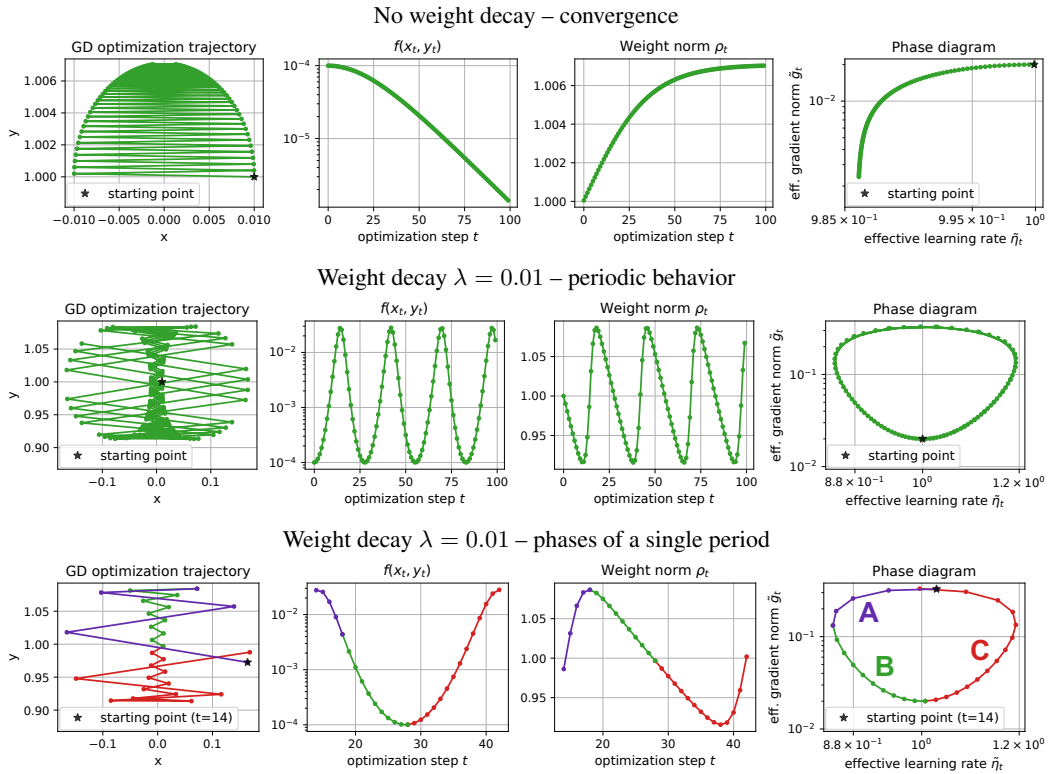

Figure 14: Minimization of a simple scale-invariant function $f(x, y) = x^2/(x^2 + y^2)$ with and without weight decay. For all experiments the initial point $(x_0, y_0) = (0.01, 1.0)$, learning rate $\eta = 1$.

If we minimize f without weight decay, the optimization procedure converges to a stationary point since its effective learning rate monotonically decays, as can be seen in the top row of Figure 14. This behavior accords with the results of Arora et al. [1].

However, with weight decay we can observe the same periodicity of the optimization dynamics as demonstrated by experiments with neural networks (see the middle row of Figure 14). Moreover, in this case, the optimization experiences the same three phases in the period (see the bottom row of Figure 14, which is analogous to Figure 4 in the main text).

This confirms that the periodicity of optimization dynamics is a general property of scale-invariant functions optimized with weight decay and is not specific to neural networks.
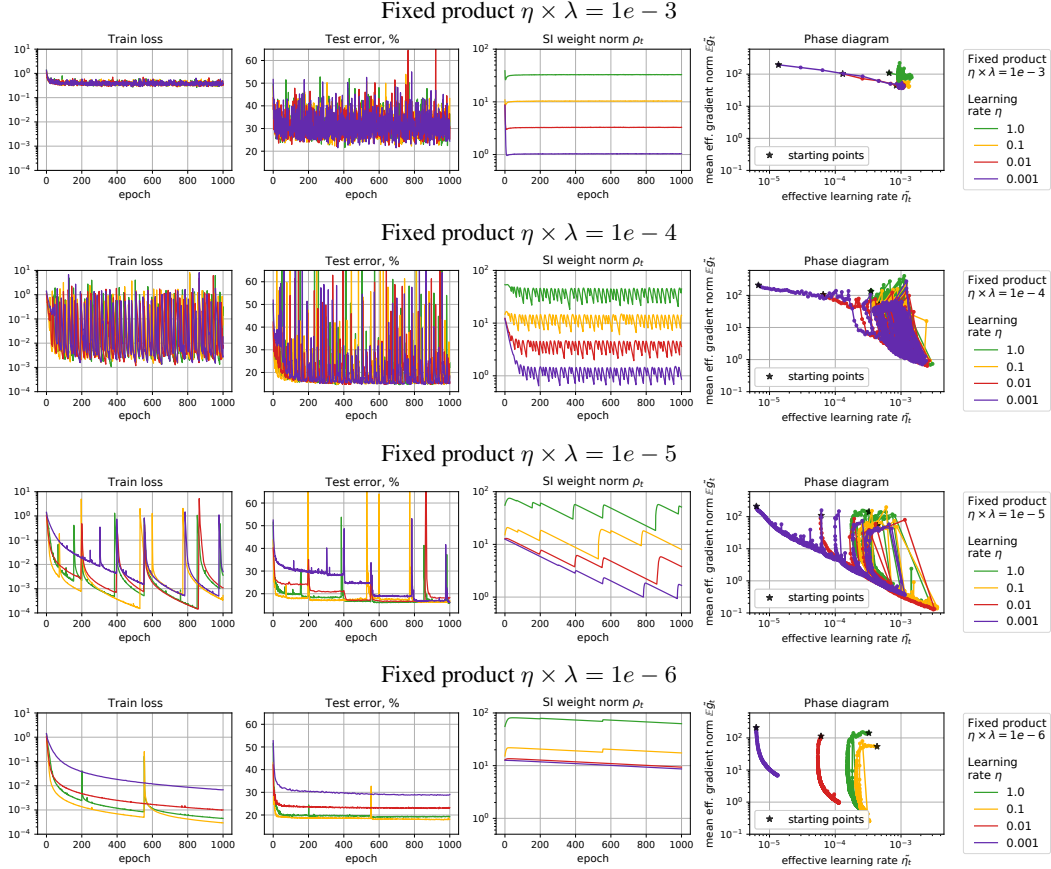
Figure 15: Training dynamics of scale-invariant ConvNet on CIFAR-10 trained with fixed learning rate – weight decay products. Axes limits are the same in each column for convenient comparison.
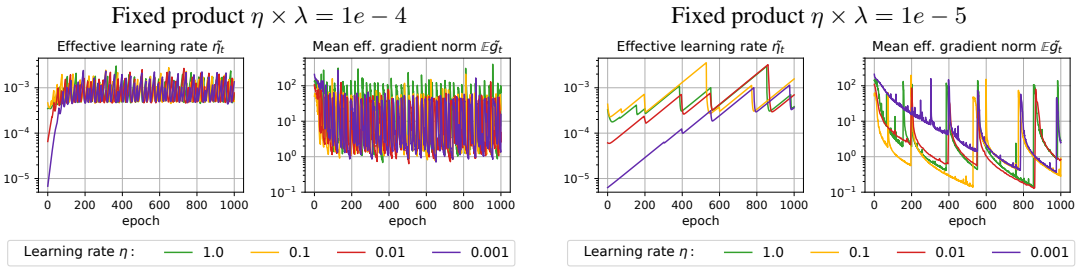

Figure 16: A closer look at dynamics of the effective learning rate and mean effective gradient norm of scale-invariant ConvNet on CIFAR-10 trained with two different fixed learning rate – weight decay products. Axes limits are the same for corresponding metrics for convenient comparison.

F Influence of learning rate and weight decay on the periodic behavior of scale-invariant networks

F.1 Fixed learning rate – weight decay product

In this section, we discuss the effect of the learning rate – weight decay product on the training process. Figure 15 visualizes training progress for different values of the product (plot rows) and variable ratio of two specified hyperparameters (different lines in each row). We observe that training converges to similar consistent behavior with the fixed learning rate – weight decay product. Specifically, the frequency of the periods, the minimal achieved train loss and test error, and the ranges of the effective
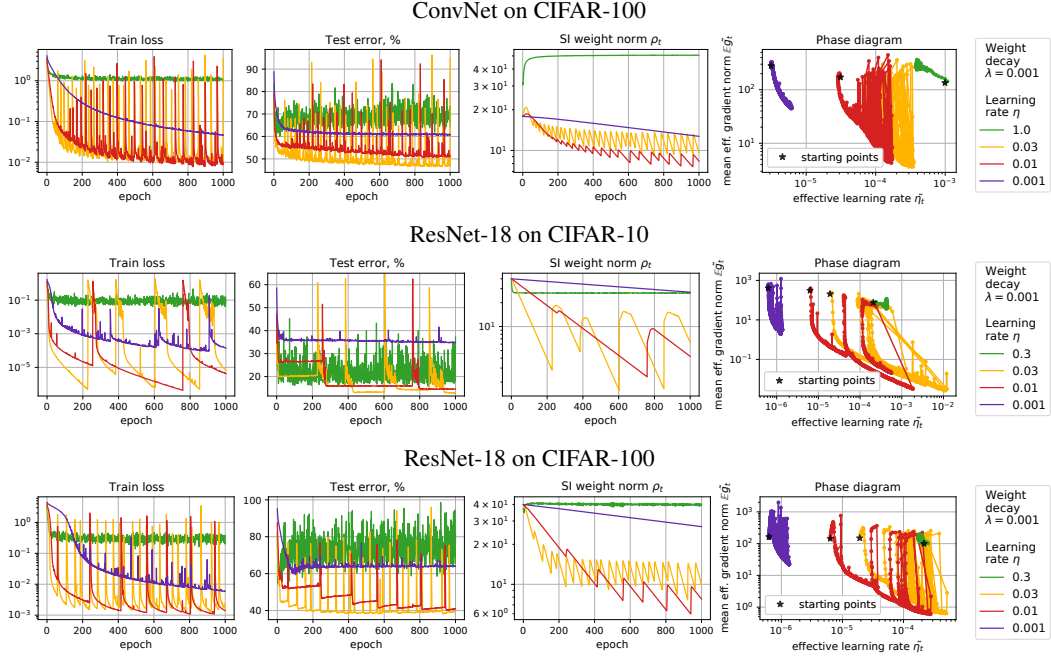
Figure 17: Training dynamics of scale-invariant networks trained with fixed weight decay and different learning rates.

gradient norm and the effective learning rate are similar across different lines in one row. The last-mentioned ranges are visualized in more detail for selected setups in Figure 16. The described empirical results agree with Remark 3 in Appendix A.6. Mainly, the remark states that with a fixed learning rate – weight decay product and bounded effective gradient norm, training converges to a bounded effective learning rate, and the effective learning rate bounds depend only on the effective gradient norm bounds. In practice, we observe that the last-mentioned bounds are similar across different ratios of weight decay and learning rate (see Figure 16). Thus, the effective learning rate bounds are also similar across different ratios (see Figure 16).

However, although the characteristics of the *consistent* periodic behavior are similar across different ratios of the learning rate and the weight decay when their product is fixed, the length of the *warm-up* stage may vary. The reason is that we use the standard initialization for all networks, i.e., the same initial weight norm for all combinations of hyperparameters. At the same time, given different ratios of weight decay and learning rate, the weight norm converges to different ranges (see Figure 15 and Proposition 3). The final weight norm may substantially differ from the initial weight norm, and the larger the difference, the longer the warm-up stage.

We note, however, that, according to Proposition 4 in Appendix A.1, if we fixed the direction of initialization (i.e., the point on the unit sphere) and then appropriately rescaled it (proportionally to the square root of the learning rate), the training dynamics would be exactly the same for different ratios of learning rate and weight decay, given their product is unchanged, including the warm-up stage.

F.2 Fixed weight decay and different learning rates

Figure 17 supplements Figure 3 and shows how the learning rate affects the periodic behavior for different dataset-architecture pairs when the weight decay is fixed. For CIFAR-100, we had to increase the ConvNet's width factor up to 64 and the last layer's weight norm up to 20 to ensure the network is able to learn the train dataset and achieve low train loss. The general picture is the same as described in Section 6: the periodic behavior is absent for too low or too high learning rates and present for a range of learning rate values, which also allow lower test error. Interestingly, for ResNet on CIFAR-10 with the learning rate of 0.03, phase A is noisy and quite long because of the relatively
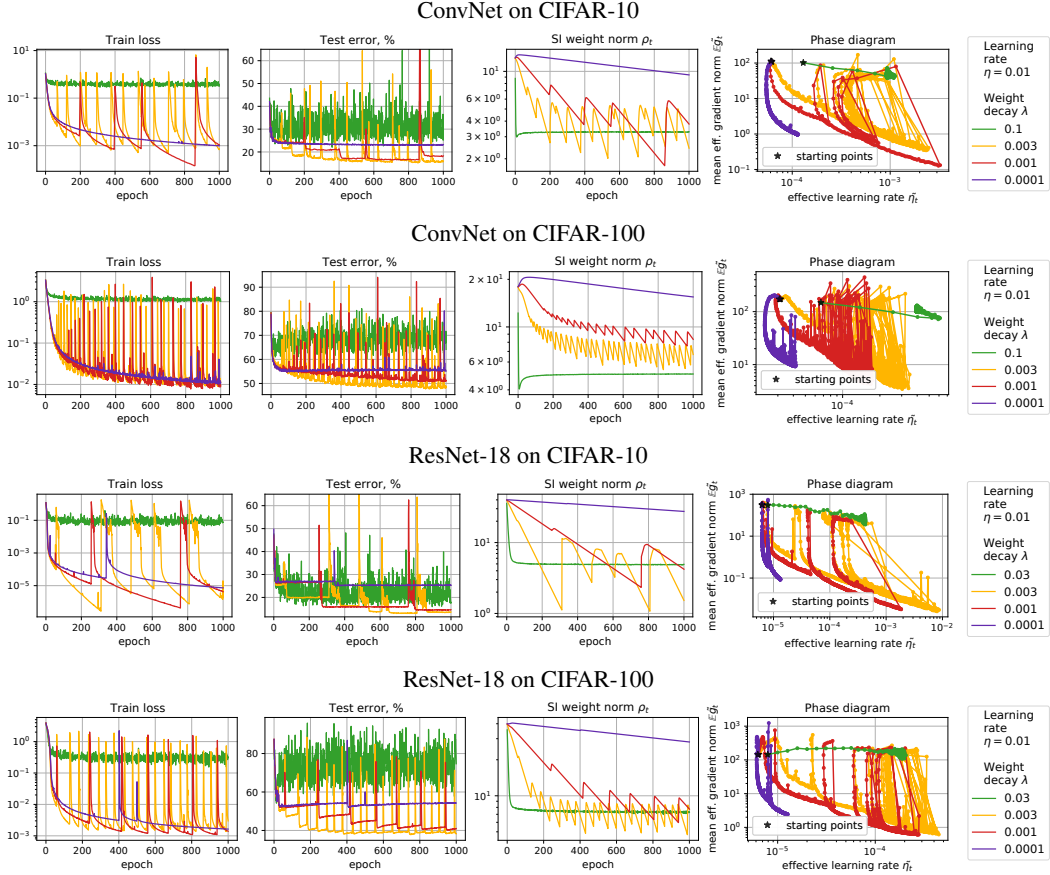
Figure 18: Training dynamics of scale-invariant networks trained with the fixed learning rate and different weight decays.

high learning rate, but training still proceeds to phase B, while for larger learning rate, training gets stuck at high train loss.

F.3 Fixed learning rate and different weight decays

Figure 18 shows the periodic behavior when the learning rate is fixed, and the weight decay is varied for different dataset-architecture pairs. The general observations are the same as when the learning rate is varied with the fixed weight decay. Notably, the periodic behavior is absent for too low or too high weight decay coefficients and present for a range of weight decay values, which also allow reaching lower test error. Further, using a larger weight decay increases the frequency of the periods.

G Influence of the last layer weight matrix norm

In scale-invariant neural networks, we fix the weights of the last layer. Moreover, we renormalize the weight matrix to the specified weight norm, which becomes a new hyperparameter. This hyperparameter determines the level of the neural network's confidence in its predictions, and, in the main text, we set it to a large value (10) to achieve high confidence and to make our setup closer to the conventional neural network training (when all parameters are trained). In this section, we discuss the influence of the specified hyperparameter on periodic behavior.

Figure 19 shows results for ConvNet on CIFAR-10 and ResNet on CIFAR-100 and different values of the last layer's weight norm. The lowest presented last layer's weight norms are close to the norms obtained at random initialization without rescaling. Using low last layer's weight norm leads to low network's confidence which prohibits reaching low train loss and may result in the absence of
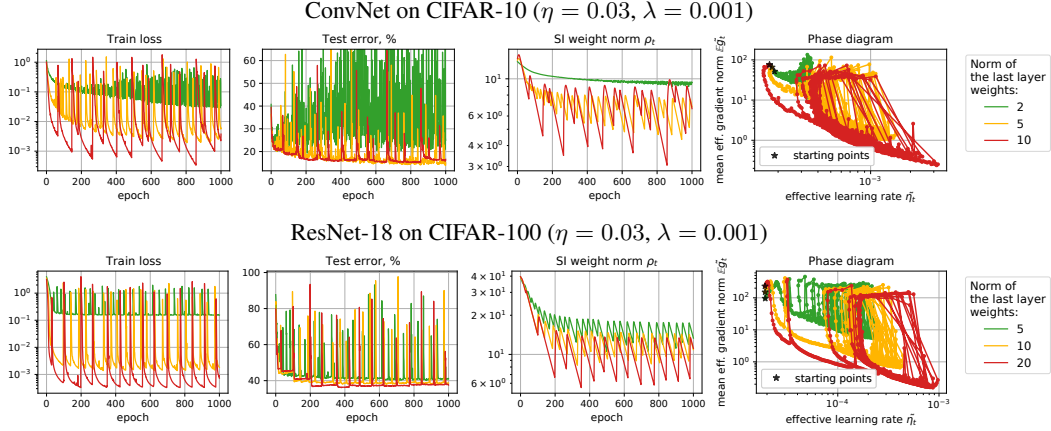
Figure 19: Influence of the last layer weight matrix norm on the periodic behavior.
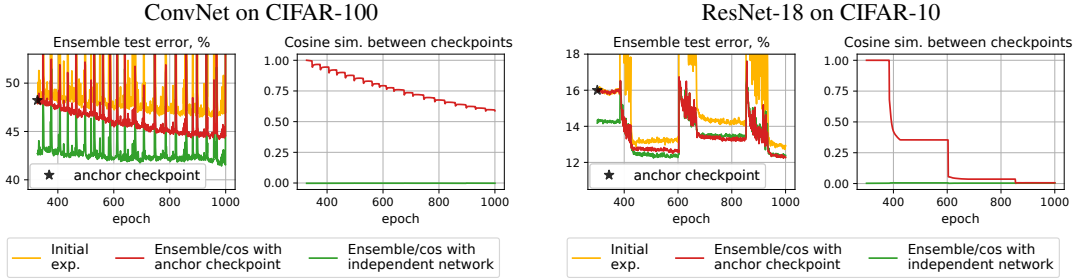

Figure 20: Similarity in the weight space (cosine sim.) and in the functional space (ensemble test error) for different checkpoints of training scale-invariant ConvNet on CIFAR-100 (left pair) and ResNet on CIFAR-10 (right pair) using SGD with weight decay of 0.001 and learning rate of 0.03.

the periodic behavior. In the main text, we use larger values of the last layer's weight norm, which circumvents this issue.

H   Minima achieved at different training periods

Figure 20 supplements Figure 6 for analyzing the weight/functional similarity of optima achieved at different training periods. The general observations are the same as in Section 6. Interestingly, the ensemble of two models spawned by optima from different periods can reach the error of two independent networks ensemble for both architectures on the CIFAR-10 dataset and does not reach one on the CIFAR-100 dataset (in given epochs budget).

I   Practical modifications

Figure 21 supplements Figure 7 and shows the presence of the periodic behavior in a more practical setting, i.e., with trainable non-scale-invariant parameters, momentum, and data augmentation, for ConvNet on CIFAR-100 and ResNet on CIFAR-10 and CIFAR-100. For a more detailed discussion, see Section 7 in the main text.

We also consider training neural networks with a more sophisticated optimizer, Adam [13], and show the presence of the periodic behavior for ConvNet on CIFAR-10 in Figure 23.

In order to show that our results extrapolate to other normalization approaches besides batch normalization, we train ConvNet on CIFAR-10 using layer normalization [2] and instance normalization [24] and demonstrate the presence of the periodic behavior in this setting in Figure 22.
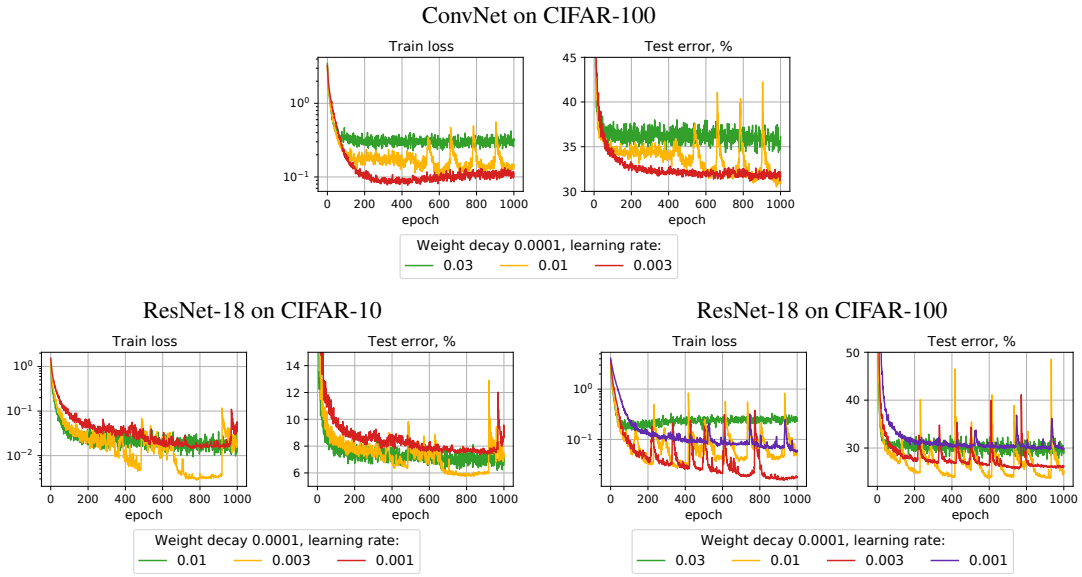
Figure 21: Training dynamics of networks trained with more practical modifications, i.e., with learnable non-scale-invariant parameters, momentum, and augmentation (all modifications together).
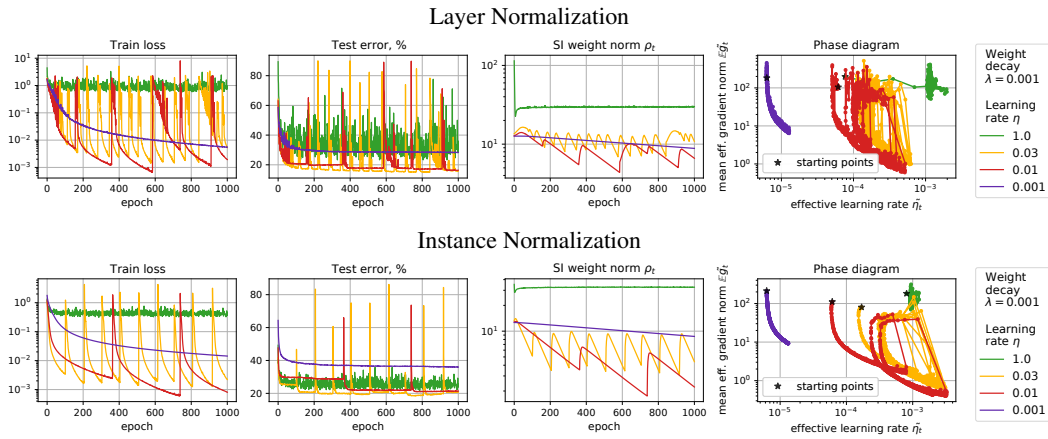

Figure 22: Training dynamics of scale-invariant ConvNet with other normalization approaches on CIFAR-10.
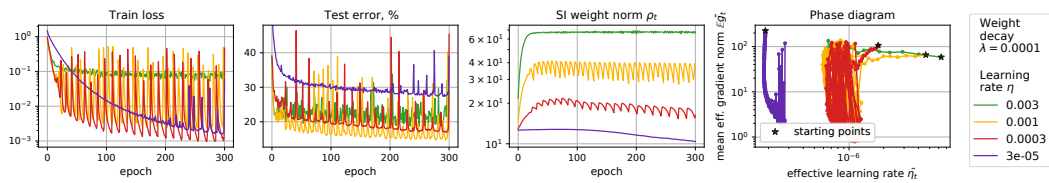

Figure 23: Training dynamics of scale-invariant ConvNet on CIFAR-10 trained using Adam.

J   Comparison with previous works

In this section, we compare our experimental setup with that of the prior art and point out the main factors for why previous experiments mostly do not show periodic behavior.

As we stated in Section 7, periodic behavior is not usually observed when training normalized neural networks due to a relatively small epochs budget and usage of learning rate schedules. Moreover, some hyperparameters settings can make periods too slow or even unreachable, which both hinder observation of the periodic behavior in practice (see, e.g., the smallest and the highest learning rate curves in Figure 3). Finally, the use of data augmentation and/or models that are too simple to learn a given dataset does not allow even the first period to be completed within a reasonable time frame. These are the key reasons periodic behavior was mainly not reported in the literature previously. Below we discuss the particular aspects of several most related works.

One of the works closest to ours, Li et al. [19], discovers the unstable behavior of full-batch GD training of scale-invariant networks and at the same time reports convergence to a constant equilibrium when training with SGD. We suppose that the experiments of Li et al. [19] with full-batch GD depict exactly our periodic behavior. Speaking of SGD experiments, we suspect that, despite a large epochs budget, Li et al. [19] did not encounter periodic behavior in most of their experiments due to data augmentation, different hyperparameters settings, and learning rate schedules. In other words, they mainly observed a prolonged phase A in their experiments without reaching the end of even the first period, which may seem like convergence to a stable equilibrium.

Wan et al. [26], who also study the convergence of scale-invariant parameters dynamics to the equilibrium (which, however, is now *dynamical*, i.e., depends on the behavior of effective gradients, in par with our work), did not find periods in their experiments as well. This can also be attributed to data augmentation and learning rate schedules but most importantly to short training, which does not allow finishing phase A of the first period, as seen by the increasing effective gradients norm throughout training in Figure 2 therein.

As mentioned in the main text, Li et al. [17] discovered that training weight-normalized neural networks with improperly selected weight decay may become unstable and even result in training failure since the numerical gradient updates are beyond the representation of float. This is the extreme case of destabilization in phase C when scale-invariant parameters approach the origin too close and the gradients blow up so that training is already unable to recover due to numerical issues. In our experiments, such situations did not occur, however, we hypothesize that they can be encountered when training very large networks equipped with both weight normalization and feature normalization, which may amplify the destabilization effect of approaching the origin. Other experiments of Li et al. [17] did not reveal the periodic behavior for the same reasons as above: data augmentation, insufficient training duration, and learning rate schedules.