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

Anonymous Author(s)
Affiliation
Address
email

Abstract

Despite the conventional wisdom that using batch normalization with weight decay may improve neural network training, some recent works show their joint usage may cause instabilities at the late stages of training. Other works, in contrast, show convergence to the equilibrium, i.e., the stabilization of training metrics. In this paper, we study this contradiction and show that instead of converging to a stable equilibrium, the training dynamics converge to consistent periodic behavior. That is, the training process regularly exhibits instabilities which, however, do not lead to complete training failure, but cause a new period of training. We rigorously investigate the mechanism underlying this discovered periodic behavior both from an empirical and theoretical point of view and show that this periodic behavior is indeed caused by the interaction between batch normalization and weight decay.

1 Introduction

Normalization approaches, such as batch or layer normalization, have become key techniques for successful training of modern deep neural networks [12, 2, 22, 19, 25]. Despite much recent work [3, 20, 9, 26], it is still not completely understood how normalization influences the training process. We investigate the interplay between batch normalization (BN) [12], one of the most popular normalization schemes for convolutional neural networks, and weight decay (WD), a commonly used regularization technique for conventional training with stochastic gradient descend (SGD).

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 easier 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. 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* [23, 10, 27, 16, 17, 24, 18], 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.

On the one hand, Li et al. [17] 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

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

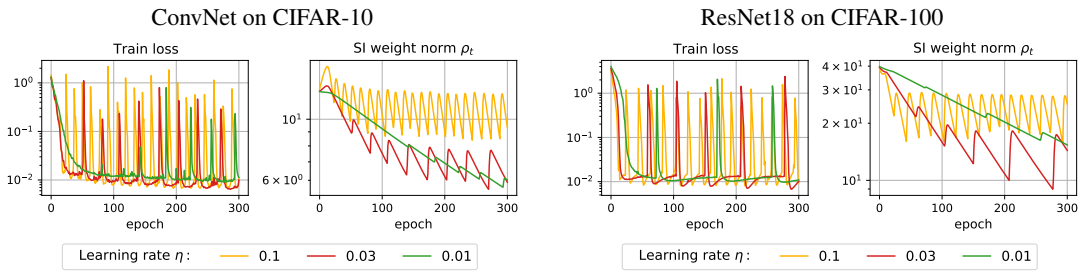

Figure 1: Periodic behavior of ConvNet on CIFAR-10 and ResNet18 on CIFAR-100 trained using SGD with weight decay of 0.001 and different learning rates. All weights are trainable.

converge to a constant value. A number of other works hold a similar equilibrium view [23, 5, 24]. On the other hand, several works [15–17] underline that using WD may cause approaching the origin (zero scale-invariant weights) which results in training *instability*. Particularly, Li et al. [15] reveal that approaching the origin in weight-normalized neural networks leads to numerical overflow in gradients and the subsequent training failure. Li and Arora [16] also underline that scale-invariant functions are ill-conditioned near 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. [17] 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 of them are true only to some extent. Specifically, we show that 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. Examples of this behavior are provided in 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 analysing the underlying mechanism for fully scale-invariant neural networks trained with standard constant learning rate SGD (Section 3) or GD (Appendix C).
- We provide theoretical grounding for our findings by generalizing previous results on the equilibrium condition, analysing the necessary conditions for destabilization of training, and relating the frequency of destabilization to choice of hyperparameters (Section 4).
- We conduct a rigorous empirical study of this periodic behavior and show its presence even in more practical scenarios with momentum, augmentation, and neural networks incorporating non scale-invariant weights (Sections 5 and 6).

2 Methodology

Investigating the effects caused by batch normalization is simpler when *all* the weights of the neural network are scale-invariant, and following [16, 17], we also focus on this setup. We consider ResNet18 and a simple 3-layer batch-normalized convolutional neural network (ConvNet)[1] on CIFAR-10 [13] and CIFAR-100 [14] datasets, and rely on the approach of Li and Arora [16] to achieve scale invariance, i.e., we insert additional BN layers and fix the non scale-invariant weights to be constant. Particularly, 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 multiply the weights of the (fixed to random) last layer by 10. The influence of this coefficient is shown in Appendix F.

To further facilitate the simplicity of the setup when discussing the nature of the periodic behavior, we use SGD with the fixed learning rate and do not use momentum or data augmentation. Later on,

[1] Both architectures in the implementation of `https://github.com/g-benton/hessian-eff-dim`.

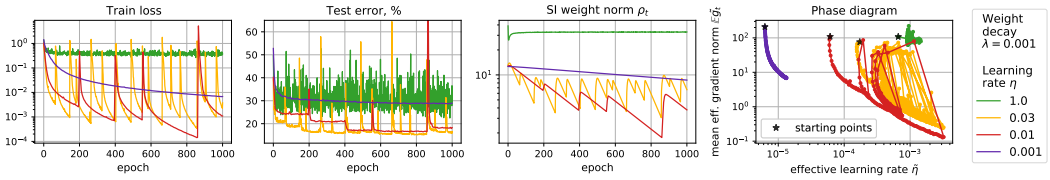

Figure 2: Periodic behavior of scale-invariant ConvNet on CIFAR-10.

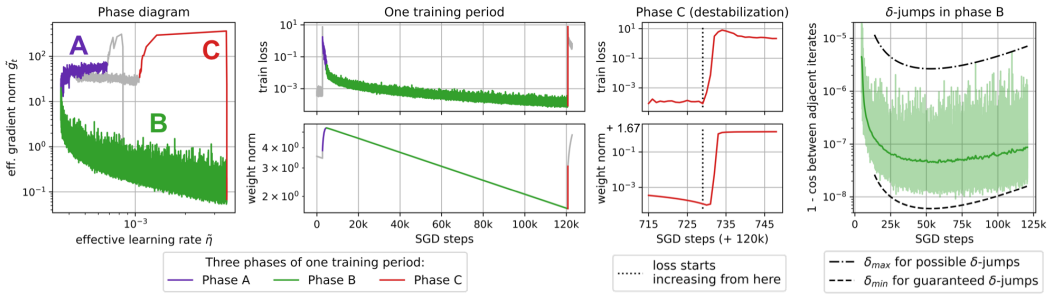

Figure 3: A closer look at one training period for scale-invariant ConvNet on CIFAR-10 trained using SGD with weight decay of 0.001 and the learning rate of 0.01. Three phases of the training period are highlighted. Here the train loss and the effective gradient norm computed over a mini-batch are logged after each SGD step. The rightmost plot compares empirically observed cosine distance between weights at adjacent SGD steps with theoretically derived bounds in Section 4.1. Cosine distance is presented along with the smoothed trend.

we discuss the influence of using a more practical setup on our findings. 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, $\{10^{-k}, 3 \cdot 10^{-k}\}_{k=1,2,3}$ and choose the most representative ones for each visualization, since it is difficult to distinguish many periodic functions on one plot. For scale-invariant neural networks, training with a fixed weight decay – learning rate product converges to a similar behavior, regardless their ratio: we show it empirically and discuss it from theoretical point of view in Appendix E.1; the same was noticed in [16, 17]. Thus, in the main text, we provide only the results for the varied learning rate and the fixed weight decay of 0.001. Results for the varied weight decay are presented in Appendix E.3.

At each epoch of training, 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 in the scale-invariant space. Particularly, we log the *effective learning rate* and the (mean over mini-batches) norm of *effective gradients*, i.e., the learning rate and gradients reproducing the SGD update on the unit sphere [10, 18]. We plot these two metrics over two axes of the *phase diagram* to visualize their simultaneous dynamics that serve as a starting point to understanding the mechanism underlying the periodic behavior.

The rest of the work is organized as follows. In Section 3, we explain the reasons for the occurrence of the discovered periodic behavior. Section 4 provides the theoretical grounding for the periodic behavior. After that, Section 5 contains the empirical investigation of the periodic behavior and Section 6 shows whether the periodic behavior is present in the more practical setting.

3 Periodic behavior and its underlying mechanism

As discussed in the previous section, the majority of our experiments focus on the setting with a fully scale-invariant neural network trained with standard SGD. Figure 2 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 E.2, we show the presence of the periodic behavior for other dataset-architecture pairs. Moreover, the same periodic behavior is also present for full-batch gradient descend, see Appendix C.

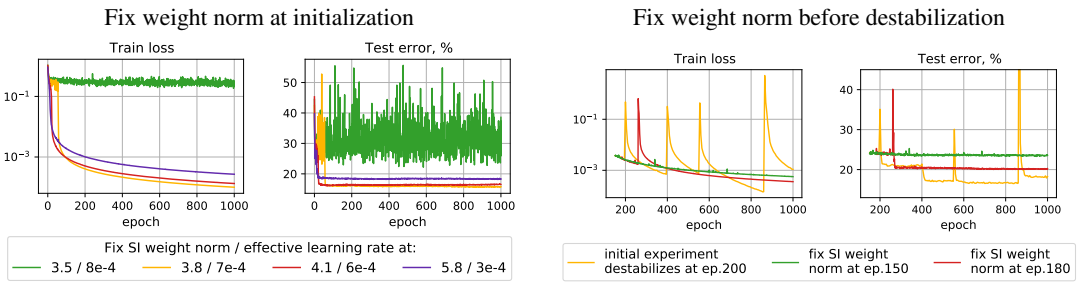

Figure 4: The absence of the periodic behavior for training with the fixed weight norm. Scale-invariant ConvNet on CIFAR-10 trained using SGD with weight decay of 0.001 and learning rate of 0.01. Left pair: the weight norm is fixed at random initialization of different scales. Right pair: the weight norm is fixed at some epoch of regular training before destabilization.

The observed periodic behavior occurs because of the interaction between batch normalization and weight decay, and particularly, because of their competing influence on the weight norm. On one hand, weight decay aims at decreasing the weight norm. On the other hand, using batch normalization makes the weights scale-invariant, resulting in the gradient of the loss being orthogonal to the radius connecting current weight and zero weights. Thus, shifting weights towards the loss's gradient direction always increases the weight norm. These two forces alternately outweigh each other for quite long periods of training, resulting in the periodic behavior.

Let us examine a single period in greater detail by analysing Figure 3, which 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 outweight weight decay. This results in an 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 outweighted by weight decay. Thus, the weight norm starts decreasing, and the effective learning rate starts increasing, i.e., we move along phase B of the phase diagram. 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, being multiplied by a high effective learning rate, result in 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 3 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 note that the essential difference between phases A and B is the "large" and "small" gradient norm of the loss. In the majority of our experiments, large loss gradients outweigh the weight decay's gradients in phase A resulting in the increasing weight norm and allowing clear distinction between two phases. However, in some experiments presented in Appendix E.2, loss gradients are permanently outweighed by the weight decay's gradients, including phase A, but still there is the visible transition from "large" to "small" loss gradient norm, i.e., from phase A to phase B. Moreover, this transition correlates with achieving near-zero train error.

We also conduct 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 4 (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 also repeat this ablation fixing the weight norm at some epoch preceding destabilization in the experiment where we observe the periodic behavior. Particularly, we use as initial experiment the one with the learning rate of 0.01 from Figure 2 and fix the weight norm at the 150-th and 180-th epochs, preceding the destabilization at epoch 200. Figure 4 (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 training process has already became unstable. However, after converging from this increase, train loss never destabilizes again.

4 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. [17] taking into account the discovered periodic behavior. To do so, we study the properties of the optimization of an arbitrary scale-invariant function $f(x)$ with (S)GD[2] with learning rate η and weight decay of strength λ:

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

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.

In the following reasoning we will use two fundamental properties holding for (stochastic) gradients of an arbitrary scale-invariant function $f(x)$ (see, e.g., Lemma 1.3 in Li and Arora [16]):

$$\begin{cases} \langle \nabla f(x), x \rangle = 0, & \text{(2a)} \\ \nabla f(\alpha x) = \dfrac{1}{\alpha}\nabla f(x),\ \alpha > 0. & \text{(2b)} \end{cases}$$

Based on these properties, we obtain the dynamics of the parameters norm induced by Eq. (1) which we also leverage in our analysis (derivation of this equation 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.

4.1 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, which may indicate training destabilization.

Definition 1 *We say that dynamics* (1) *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. Our next 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 $0 \le \ell \le L$ 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* (1) *with bounded effective gradients* $0 \le \ell \le \tilde{g}_t \le L$*, then for sufficiently small* δ *and assuming* $(1 - \eta\lambda) \lessapprox 1$*, the following approximate conditions on the* δ*-jump hold:*

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

Remark 1 *Note that 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.*

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

The rightmost plot of Figure 3 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 upper and lower bounds on the cosine distance corresponding to the possible and guaranteed δ-jumps derived from Eq. (4a) and (4b), respectively: $\delta_{\max} = \frac{\eta^2 L^2}{2\rho_t^4}$, $\delta_{\min} = \frac{\eta^2 \ell^2}{2\rho_t^4}$. We chose ℓ and L as smooth functions locally bounding the effective gradient norm in phase B (see Appendix D for details). We can clearly see that both bounds along with the cosine metric itself 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 the stochasticity. For full-batch GD, this effect is more prominent, see Appendix C.

Next, we formulate a proposition concerning how hyperparameters η and λ affect the time of occurrence of δ-jumps and hence the frequency of the periods, as destabilization of training dynamics is closely connected with δ-jumps. Proof can be found in Appendix A.5.

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 $\eta\lambda$.*

4.2 Generalization of the equilibrium condition

We now generalize the equilibrium condition presented in Li et al. [17] and characterize globally the behavior of the parameter norm in the following proposition. 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. [17] and Wan et al. [24] 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 global bounds ℓ, L. A similar assumption is made, e.g., in Theorem 1 in Wan et al. [24]. We discuss it in more detail (including the non-fulfillment case) in Appendix A.7.

Proposition 3 generalizes the results of Li et al. [17] who claimed that the effective learning rate $\tilde{\eta}_t = \eta/\rho_t^2$ [10] 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. [24] as we provide the exact limiting interval for ρ_t^2, not just bound its variance.

5 Empirical study of the periodic behavior

After discussing the reasons for the occurrence of the periodic behavior, we now further analyse its properties. In particular, we investigate: how hyperparameters affect the periodic behavior, how the

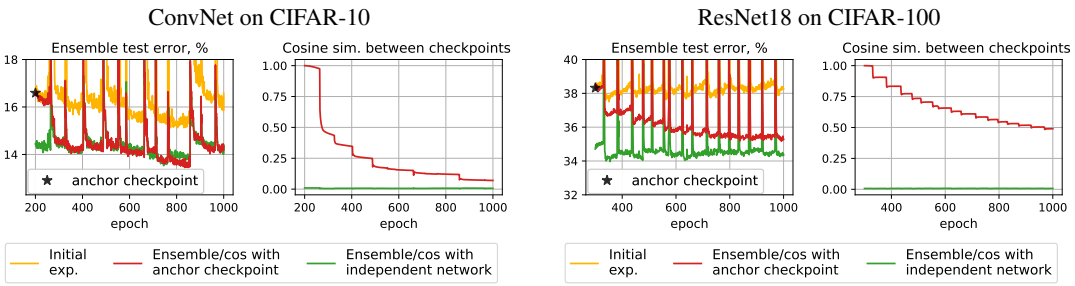

Figure 5: Similarity in the weight space (cosine sim.) and in the functional space (ensemble test error) for different checkpoints of training scale-invariant ConvNet on CIFAR-10 (left pair) and ResNet on CIFAR-100 (right pair) using SGD with weight decay of 0.001 and learning rate of 0.03.

periodic behavior evolves over epochs, and how minima achieved in different training periods differ both in parameter and functional space.

Influence of hyperparameters. We investigate the influence of two key training hyperparameters: learning rate and weight decay, but since the training dynamics mostly depend on their product (see discussion in Section 2 and Appendix E.1), we only vary the learning rate. The results for ConvNet on CIFAR-10 are given in Figure 2, the results for other dataset-architecture pairs are given in Appendix E.2 and the results on the variable weight decay are given in Appendix E.3. Our first observation is that with higher learning rates, the 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. (2b). This observation also agrees with Proposition 3 in Section 4. 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 rate, the shorter the periods, which agrees with Corollary 1 in Section 4. When using a learning rate which is too high, e.g., 1 in Figure 2, 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 which is too low, e.g., 0.001 in Figure 2, leads to extremely slow 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 using which results in the lowest test error. In Appendix E.3, we show that varying the 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 analyse how the discovered periodic behavior evolves over epochs. As was discussed in the previous paragraph, with higher learning rates, the consistent periodic behavior occurs at larger weight norms. However, the initialization may have substantially different norm. Thus we observe a *warm-up stage* in some plots of Figures 1 and 2, when the beginning of training is spent on moving towards the appropriate norm of the scale-invariant weights. Expectedly, this warm-up stage is longer for lower learning rates. Reaching the appropriate weight norm initiates a consistent periodic behavior. During the warm-up stage, SGD can still exhibit regular destabilization happening at the higher weight norms than in the stage of consistent periodic behavior. We hypothesize the reason is that at the early stage of training, SGD converges to less stable basins, i.e., with larger effective gradients, in which destabilization happens at larger norms of the scale-invariant parameters. We notice that test error decreases after each of the warm-up destabilization episodes and reaches a lower level compared to training with a fixed effective learning rate, as can be seen in Figure 4 (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 at 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]. The setup is as follows. We select

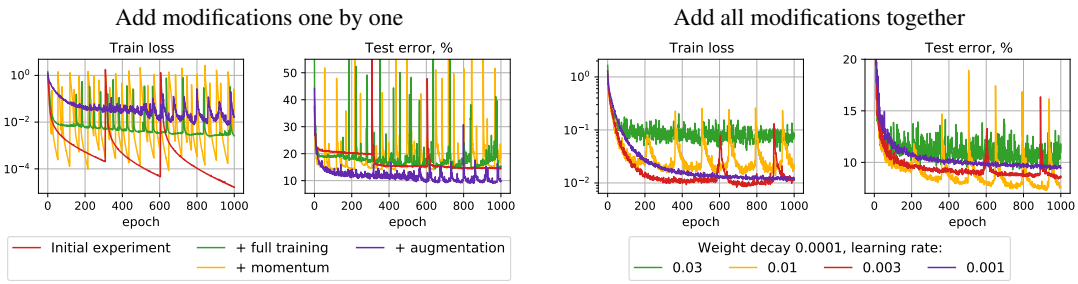

Figure 6: Periodic behavior of ConvNet (of increased width) on CIFAR-10 trained with the more practical modifications. Left: weight decay of 0.001, learning rate of 0.01.

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 main measurement. For comparison, we 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 of the anchor checkpoint, and measure the similarity between this new checkpoint and the checkpoints of the original experiment.

The results for ConvNet on CIFAR-10 and ResNet18 on CIFAR-100 are presented in Figure 5, the results for other dataset-architecture pairs are given in Appendix E.2. 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 ResNet, 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 independent network ensemble. Still, both similarities continue decreasing. For ConvNet, 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 neighbouring training periods are substantially different but their similarity is usually higher than the similarity of two independently trained networks.

6 Why is periodic behavior usually not observed in practice?

In the previous sections, we conducted experiments with scale-invariant neural networks trained with the simplest version of SGD. This allowed us to analyse the periodic behavior of train loss in details. 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 typically trained using more advanced procedures, e.g., SGD with momentum, data augmentation and a learning rate schedule. At the same time, to the best of our knowledge, the periodic behavior is usually not observed in practice. In this section, we investigate whether using these more advanced techniques preserves the periodic behavior, and if so, why is periodic behavior usually not observed in practice.

The setup of the experiment is as follows. We select some initial experiment and add modifications one by one, in order to see their effects more clearly. We also present the results for training with all modifications turned on. The plots for ConvNet on CIFAR-10 are given in Figure 7 and for other setups — in Appendix H. In this experiment, we use the 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 as well as the parameters of BN layers, since they all are not scale-invariant. We now consider the conventional procedure that implies training all the weights of the neural network. 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 affection is connected to the trainable last layer which automatically adjusts prediction confidence. In Appendix F, we show that variable prediction confidence results in different period 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 frequently used for training convolutional neural networks. We observe that using momentum does not break up the periodic behavior and increases the frequency of periods. This agrees with the commonly observed effect that momentum speeds up training [21], i.e., momentum accelerates phases A and B. Interestingly, momentum does not prevent destabilization.

Data augmentation. We next consider training on the dataset augmented 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 large train loss. As a result, the train loss outweighs weight decay 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 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, augmentation and unfrozen scale-variant parameters used simultaneously and show the presence of the periodic behavior. So, why is periodic behavior usually not observed in practice? The reasons are twofold. First of all, the interplay between different modifications narrows the range of hyperparameter values for which the periodic behavior is present. When non scale-invariant parameters are trained, training can converge to low test error only with certain values of weight decay. Moreover, with data augmentation, the periodic behavior occurs only with relatively high learning rates (with lower learning rates, the training is too slow to converge to the periodic behavior in 1000 epochs), while with momentum, using too high learning rates may result in training failure in phase A. In sum, the periodic behavior is present only for a narrow range of learning rates. Second, practical setup implies using learning rate schedules and a limited number of epochs, which do not preserve the periodic behavior.

7 Conclusion

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

Limitations and negative societal impact We discuss only conventional training of convolutional neural networks and do not consider other practical setups. However, we believe that our findings extrapolate to other normalization schemes as well. We also focus on a particular source of instability induced by BN and WD, yet, there are other factors that may make training unstable [7]. This is an interesting direction for future research. To the best of our knowledge, our work does not have any direct negative societal impact. However, during conducting the research we had to spend a lot of 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. C., Hinder, O., and Sidford, A. (2017). Lower bounds for finding stationary points ii: First-order methods. *arXiv preprint arXiv:1711.00841*.

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

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

[15] 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.

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

[17] 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.

[18] 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*.

[19] 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.

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

[21] 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.

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

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

[24] 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*.

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

[26] 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*.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We provide experimental results for various hyperparameter combinations which itself provides the evidence for the consistence of the discovered effect.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 2.
   (b) Did you mention the license of the assets? [Yes] See Appendix B. CIFAR datasets are distributed under MIT license and code is under Apache-2.0 License license (we provide the link to the GitHub in Section 2.)
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide our code in Supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We train networks on the CIFAR datasets which are widely used in NeurIPS papers.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We train networks on the CIFAR datasets which are widely used in NeurIPS papers.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]