On the Weight Dynamics of Deep Normalized Networks

Christian H.X. Ali Mehmeti-Göpel [1] **Michael Wand** [1]

Abstract

Recent studies have shown that high disparities in effective learning rates (ELRs) across layers in deep neural networks can negatively affect trainability. We formalize how these disparities evolve over time by modeling weight dynamics (evolution of expected gradient and weight norms) of networks with normalization layers, predicting the evolution of layer-wise ELR ratios. We prove that when training with any constant learning rate, ELR ratios converge to 1, despite initial gradient explosion. We identify a "critical learning rate" beyond which ELR disparities widen, which only depends on current ELRs. To validate our findings, we devise a hyper-parameter-free warm-up method that successfully minimizes ELR spread quickly in theory and practice. Our experiments link ELR spread with trainability, a relationship that is most evident in very deep networks with significant gradient magnitude excursions.

1. Introduction

In the past decade, combining neural networks and big data has enabled dramatic breakthroughs (Krizhevsky et al., 2012; OpenAI, 2023), and network *depth* has been a key factor: Compositions of many individual layers provide rich function spaces that empirically appear to be better-aligned with real-world data distributions than any other inductive biases we are aware of today. A fundamental problem of deep networks, maybe easily brushed over as technicality at first sight, is the problem of vanishing and exploding gradients. Propagating signals through a multi-layer networks is not easy: In the forward pass, the magnitude input signals easily increases or decreases, thus leading to an exponential excursion of *signal magnitude*. Similarly, during the backward pass, we easily obtain similar excursion of *gradient magnitudes* (Yang et al., 2019). Further, using deep stacks of layers also easily increase correlations, thereby causing *vanishing dimensionality* (Saxe et al., 2013). Proper Initialization (He et al., 2015a) can reduce the problem; trying to prevent it completely in a simple feed-forward network is challenging though (Pennington et al., 2017).

Modern architectures (He et al., 2016a; Vaswani et al., 2017) thus usually address these issues by combining *residual connections* (He et al., 2016a) and variants of *normalization layers* such as *batch normalization (BN)* (Ioffe and Szegedy, 2015). The former implicitly performs a down-weighting of deep paths, exponentially with depth (Veit et al., 2016), and in combination with normalization layers, this effect is further increased (at initialization) by decreasing the weight of the residual branch (De and Smith, 2020). The central objective of our paper is to understand how the *dynamics* (over training time) of gradient *magnitude* excursions (we do not consider correlations) are affected by normalization layers (BN and the similar).

2. Related Work and Contributions

Understanding of the benefits of BatchNorm *standalone* is not straightforward and still subject to debate. The initial claim of reduced "internal covariate shift" was quickly refuted (Awais et al., 2021) and many alternative explanations were proposed, such as smoothing of the loss surface (Santurkar et al., 2018) or enabling bigger learning rates (Bjorck et al., 2018). Salimans and Kingma (2016) introduced WeightNorm, a method to decouple a layer's length and direction by training them as independent network parameters. They also demonstrated that in weight-normalized networks, gradients are orthogonal to layer weights, allowing update size calculation via the Pythagorean theorem. Hoffer et al. (2018) showed that the "effective step size" in normalized networks is approximately proportional to $\frac{1}{||W||_2^2}$; this shows that scale invariance gives us an additional degree of freedom, as scaling a layer's weights is equivalent to inversely scaling its gradients or learning rate. You et al. (2017) have observed that the ratio $\frac{||\nabla W||_F}{||W||_F}$, which we call effective learning rates (ELR), can vary wildly (up to a factor ~ 250 in AlexNet-BN) between layers after only one step of gradient descent. The authors conjecture that this can create instability in training, especially for large

*Equal contribution [1]Department of Computer Science, Johannes-Gutenberg University, Mainz, Germany. Correspondence to: Christian H.X. Ali Mehmeti-Göpel <chalimeh@uni-mainz.de>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

batch training requiring high learning rates, and propose to re-scale gradients by their effective learning rate. You et al. (2020) have later proposed a modified version of this algorithm for increased performance with transformer models. Brock et al. (2021) have combined a similar re-scaling of the gradients with gradient clipping and are able to train normalizer-free networks using this technique. Bernstein et al. (2020) supported this intuition by showing that in a perturbed gradient descent, an optimization step decreases the loss function if all layer-wise ELRs are bounded by a term that depends on the perturbation angle. Arora et al. (2019) described the auto rate-tuning effect, proving that gradient descent asymptotically converges to a stationary point without manual tuning of learning rates for specific layers, given certain assumptions. Wan et al. (2021) prove that the "angular update" (a measure similar to ELR) of a given normalized layer eventually converges to a constant limit value which does not depend on initial conditions, but rely on weight decay for their demonstration. Interestingly, Li and Arora (2020) show that using weight decay with a constant learning rate schedule is mathematically equivalent to using no weight decay and an exponentially increasing learning rate schedule.

The above-mentioned works show that the learning speeds of different layers do eventually align, but glance over the importance of correct learning rate scheduling in the early training phase, which we believe to be crucial in practice. We find that while convergence is always guaranteed given simplifying assumptions and over an indefinite number of iterations, choosing an excessively high learning rate, especially in the first steps of training, can drastically increase imbalances in layer-wise learning speeds (ELR spread) to a degree where recovery is impossible within a realistic time frame. Furthermore, Li and Arora show a connection between weight decay and warm-up but do not demonstrate how these techniques affect ELR spread.

In this work, we model the dynamic effects solely induced by normalization layers and assume that the layer-wise gradient magnitude excluding normalization effects (base gradient magnitude) remains constant over time. In this setting, we derive a model predicting the evolution of a network's weight dynamics (expected layer-wise gradient and weight norms). In the gradient flow, this behavior reduces to a non-linear ODE with a closed-form solution, where all ELR ratios between layers smoothly converge to 1. When training with higher learning rates, the behavior changes fundamentally, as the layer with the highest ELR can flip even below the layer with the lowest ELR in a single step if a certain *critical learning rate* is exceeded, which in turn increases ELR spread of the network. When training with constant learning rates, ELR spread can increase only during the first step, slowing down convergence, but still eventually converging. From there, we derive a warm-up scheme that is guaranteed to converge in num_layers steps. Empirically, we were able to show that high ELR spreads indeed seem to correlate with low trainability: by using techniques that control ELR spread (gradient normalization and warm-up), we are able to reduce the high (initially exponential in the number of layers) ELR spreads of a 110 layer feedforward network and render the previously un-trainable network trainable. In summary, we create a theoretical framework that shows how the dynamical effects of normalization layers can help counter gradient magnitude excursions in deep neural networks.

3. Auto Rate-Tuning Effect and Its Dynamics

The core observation is that for any layer N that is invariant to scaling in the forward pass $N(\gamma \cdot x) = N(x)$ (e.g. all normalization layers), its gradient scales inversely with its input:

$$\frac{dN}{d\gamma x}(\gamma x) = \frac{1}{\gamma}\frac{dN}{dx}(x). \tag{1}$$

This is a simple consequence of the chain rule and has been shown for BatchNorm by Wu et al. (2018) and for LayerNorm by Xiong et al. (2020). Secondly, Arora et al. (2019) show that since normalization layers are scale-invariant, no gradient can flow in this direction. Hence, weight updates ∇W are orthogonal to the weights W themselves:

$$\langle \nabla W, W \rangle = 0. \tag{2}$$

We now explore how this affects a network's weight dynamics. Intuitively, the weight norm of layers with high gradient norms grows fast and thus down-scales the gradient, leading to auto-regulation: this effect is called *auto rate-tuning*. We would like to point out that in a realistic scenario, the data tensor is multi-dimensional and condition 1 is satisfied along a subset of its dimensions (e.g. the batch, height and width axis for BatchNorm); auto-rate tuning is therefore given along those dimensions.

3.1. Sufficient Conditions for Auto Rate-Tuning / Correct Placement of Normalization Layers

A necessary condition for auto rate-tuning of a linear layer L is the invariance of the network's output with respect to re-scaling the weights in L (Arora et al., 2019). We deduct that any type of normalization layer (e.g. BatchNorm, LayerNorm) induces auto-rate tuning and that placing a normalization layer directly after every linear layer, as it is the case in most convolutional networks, is sufficient to achieve scale-invariance. In Transformer models, this was initially not the case, and we conjecture that this could explain the improvements when adding additional normalization layers in the feedforward blocks (Shleifer et al., 2021) or query/key blocks (Henry et al., 2020). Arora et al. also note that the

scale invariance property is not disrupted by positive homogeneous functions of degree 1. We infer the following classification of commonly used layers:

Auto-tuning passes through	Breaks auto-tuning
Linear layers w/o bias	Linear layers w. bias
Homogeneous nonlin. of deg. 1	Other nonlinearities
Dropout	MaxPool

If a residual connection is placed in-between a linear layer and the next normalization layer, it can break auto rate-tuning; this is the case e.g. in a ResNet v2 (He et al., 2016b).

3.2. Training Dynamics Induced by Auto Rate-Tuning

To model training dynamics, we assume that weights of a given layer, as well as their gradients, are random matrices where entries are normally distributed with zero mean and a time-dependent standard deviation that is uniform in each layer. We parameterize training time $t \in \mathbb{R}$ such that $t_i = i \cdot \lambda^2$ after i optimization steps with a constant learning rate $\lambda > 0$. In this notation, gradient descent updates can be written as $W(t_{i+1}) = W(t_i + \lambda^2) = W(t_i) - \lambda \nabla W(t_i)$. The updates preserve zero norm and uniform variance of all entries in W. Assuming the independence of all entries in the weight and gradient matrices, we can deduce the following update rule from the orthogonality condition (2):

$$||W(t_{i+1})||_F^2 = ||W(t_i)||_F^2 + \lambda^2 ||\nabla W(t_i)||_F^2. \quad (3)$$

Condition (1) implies that gradient updates are inversely proportional to the current layer weights. We now assume that the "base gradient" of a layer, meaning the gradient magnitude excluding normalization induced scaling effects, is constant during training i.e.

$$\mathbb{E}(||W(t_i)||_F \cdot ||\nabla W(t_i)||)_F = c, \quad (4)$$

for a constant $c \in \mathbb{R}$ at all times-steps t_i. We discuss the limitations of this assumption in Section 4.1.2. Using shorthand $\sigma^2(t_i) := \mathbb{E}(\|W(t_i)\|_F^2)$ and $\sigma(t_i) = \sqrt{\sigma^2(t_i)}$, we obtain:

$$\sigma^2(t_{i+1}) = \sigma^2(t_i) + \frac{\lambda^2 c^2}{\sigma^2(t_i)}, \quad (5)$$

for a constant base gradient $c > 0$ depending only on layer depth and initial weights norm that we assume to be strictly positive $\sigma^2(0) > 0$. We call this the **discrete model**.

3.3. Gradient Norms at Initialization

Feed-forward networks: The dynamics of Eq. 12 apply to all normalized layers equally, but the initial gradient norms $\|\nabla W_i(0)\|_F$ differ substantially across layers $i \leq L$: Yang et al. (2019) show that in feedforward networks with Batch Normalization, the gradient norm at initialization grows as:

$$c_i \sim \alpha^{L-i}, \quad (6)$$

with $\alpha := \sqrt{\pi/(\pi-1)} \approx 1.21$ for ReLU activations and He. initialization. See also Luther (2020) for a simplified derivation.

ResNets: When considering residual networks, as per the multivariate chain rule, the gradient of residual blocks is additive instead of multiplicative (He et al., 2016b). Additionally, frequency-dependent signal-averaging further dampens gradients in a ResNet (Ali Mehmeti-Göpel et al., 2021). It follows from the consideration for fully-connected network above and He et al. (2016b) Eq. 5 that for a residual network using ReLU units:

$$c_i \sim 1 + \lfloor \frac{L-i}{s} \rfloor \alpha^s, \quad (7)$$

where s is the number of ReLU units in a residual block.

3.4. Auto Rate-Tuning Affects Each Layer Separately

In this section, we establish that the dynamic re-scaling of gradients explored above applies to each layer independently and does not affect layers above or below, showing that a simple layer-wise view is sufficient.

Proposition 3.1 (Every Layer Auto-Tunes Separately)**.** *Consider a concatenation of a linear layer $L(x, W) = x^T W$ followed by a normalization layer N. Then, the derivative wrt. the input remains the same when layer weights are scaled by a factor γ:*

$$\frac{dN}{dx}(x, \gamma W) = \frac{dN}{dx}(x, W). \quad (8)$$

The proof can be found in the Appendix Section A.

3.5. Effective Learning Rates and Their Ratios

To account for scale variance induced by normalization layers, we are interested in the update size of the weight direction $\widehat{W} := \frac{W}{\|W\|_2}$. Similarly to van Laarhoven (2017a), by approximating $\|W(t_{i+1})\|_2 \approx \|W(t_i)\|_2$, we can write:

$$\widehat{W}(t_{i+1}) - \widehat{W}(t_i) \approx \frac{W(t_{i+1}) - W(t_i)}{\|W(t_i)\|_2} \sim \frac{\nabla W(t_i)}{\|W(t_i)\|_2}. \quad (9)$$

It is therefore imperative to consider the ratio from gradient-to-weight norm as measure of change in the layer's weights.

Definition 3.2 (Effective Learning Rate)**.** We define the effective learning rate E of a layer with weight norm σ^2 and base gradient c as:

$$E(t_i) := \mathbb{E}\left(\frac{||\nabla W(t_i)||_F}{||W(t_i)||_F}\right) = \frac{c}{\sigma^2(t_i)}. \quad (10)$$

As all effective learning rates can simply be globally re-scaled by adjusting the learning rate, we are interested in the evolution of layer-wise ratios of effective learning rates.

Definition 3.3 (Effective Learning Rate Ratios)**.** We define the effective learning rate ratio R_{jk} of two layers j and k with weight norms σ_j^2, σ_k^2 and base gradients c_j, c_k at a given time step t_i as:

$$R_{jk}(t_i) \coloneqq \frac{E_j}{E_k}(t_i) = \frac{c_j \sigma_k^2}{c_k \sigma_j^2}(t_i). \tag{11}$$

3.6. Analysis in the Gradient Flow

In this section, we show that in the gradient flow, weight dynamics have a closed-form solution and all ELR ratios converge smoothly to 1.

Theorem 3.4 (Closed-Form Solution)**.** *In the gradient flow ($\lambda \to 0$), Eq. 5 has the following closed form solution:*

$$\sigma^2(t) = \sqrt{2c^2 t + k_0}. \tag{12}$$

*with $k_0 = 4$, assuming He initialization (He et al., 2015b). We will further call this the **continuous model**.*

Proof. Starting from Eq. 5, we can utilize that $t_{i+1} = t + \lambda^2$ to drop the index and solve for the difference quotient:

$$\frac{\sigma^2(t+\lambda^2) - \sigma^2(t)}{\lambda^2} = \frac{c^2}{\sigma(t)^2}. \tag{13}$$

In the limit $\lambda^2 \to 0$, this yields the gradient flow that can be expressed as a nonlinear first order differential equation :

$$\frac{d\sigma^2}{dt} = \frac{c^2}{\sigma^2} \tag{14}$$

The exact positive solution to the differential equation is given by:

$$\sigma^2(t) = \sqrt{2c^2 t + k_0}. \tag{15}$$

Assuming He initialization, the expected initial squared weight norm is 2 for layer width n. Thus, $2 = \sigma^2(0) = \sqrt{k_0}$ and therefore $k_0 = 4$. □

Theorem 3.5 (Convergence to Fixed Point)**.** *In the gradient flow ($\lambda \to 0$), all effective learning rate ratios eventually converge given enough time, i.e. for any layer pair $j, k \leq L$:*

$$\lim_{i \to \infty} R_{jk}(t_i) = 1. \tag{16}$$

Proof. We consider two arbitrary layers j and k with respective weight norms σ_j^2, $\sigma_k^2 > 0$ and base gradients c_j, $c_k > 0$. Using the formulae for gradient norm (Eq. 14) and weight norm (Eq. 12) in the continuous model, we write:

$$\frac{E_j}{E_k}(t) = \frac{c_j}{\sigma_j^2(t)} \cdot \frac{\sigma_k^2(t)}{c_k} = \frac{c_j \sqrt{2c_k^2 t + k_0}}{c_k \sqrt{2c_j^2 t + k_0}} \overset{t \to \infty}{\longrightarrow} 1. \tag{17}$$

□

3.7. Analysis for Bigger Learning Rates

In this section, we characterize the evolution of ELR ratios for non-infinitesimal, scheduled learning rates $\lambda(t_i)$, now relying solely on the discrete model. If $\lambda(t_i)$ is constant, we find the asymptotic behavior to be the same as in the gradient flow, where ELRs ratios converge in the time limit. On the contrary to the continous model, ratios can (temporarily) widen when surpassing a certain critical learning rate.

Theorem 3.6 (Convergence to Fixed Point)**.** *In the time limit and for a constant learning rate $\lambda(t_i) = \lambda$, all effective learning rate ratios converge. For any layer pair $j, k \leq L$:*

$$\lim_{i \to \infty} R_{jk}(t_i) = 1. \tag{18}$$

The proof can be found in Appendix Section A. The main idea is that by substituting $x_i \coloneqq \frac{\sigma_j^2(t_i)}{c_j \lambda}$ and $y_i \coloneqq \frac{\sigma_k^2(t_i)}{c_k \lambda}$, we can rewrite Eq. 4 for two distinct layers j and k as two sequences obeying the same recurrence relation and consequently bound the expression.

Proposition 3.7 (Ratios Shrink)**.** *Let $j, k \leq L$ be any layer pair.*

1. *If $R_{jk}(t_i) > 1$, the ratio R_{jk} is then strictly lower in the next time step, i.e. $R_{jk}(t_{i+1}) < R_{jk}(t_i)$.*
2. *If $R_{jk}(t_i) < 1$, the ratio R_{jk} is then strictly greater in the next time step, i.e. $R_{jk}(t_{i+1}) > R_{jk}(t_i)$.*

Proof. We start by showing the first proposition. We can reformulate the expression $\frac{E_j}{E_k}(t_{i+1}) < \frac{E_j}{E_k}(t_i)$ using the definition of the effective learning rate as the following equivalent expression:

$$\frac{c_j^2 \sigma_k^4}{\sigma_j^4 c_k^2}(t_{i+1}) < \frac{c_j^2 \sigma_k^4}{\sigma_j^4 c_k^2}(t_i). \tag{19}$$

We simplify this expression and take the square root. Since σ are variances and thus non-negative, the following expression is also equivalent:

$$\frac{\sigma_k^2}{\sigma_j^2}(t_i) - \frac{\sigma_k^2}{\sigma_j^2}(t_{i+1}) > 0. \tag{20}$$

We expand the second term using Eq. 5:

$$\frac{\sigma_k^2}{\sigma_j^2}(t_{i+1}) = \frac{\sigma_k^2 + \frac{c_k^2 \lambda^2}{\sigma_k^2}}{\sigma_j^2 + \frac{c_j^2 \lambda^2}{\sigma_j^2}}(t_i) = \frac{\sigma_j^2 \sigma_k^4 + \sigma_j^2 c_k^2 \lambda^2}{\sigma_j^4 \sigma_k^2 + \sigma_k^2 c_j^2 \lambda^2}(t_i). \tag{21}$$

Substituting this term on the left hand side of Eq. 20 and combining the terms yields:

$$\frac{\sigma_k^2}{\sigma_j^2}(t_i) - \frac{\sigma_j^2 \sigma_k^4 + \sigma_j^2 c_k^2 \lambda^2}{\sigma_j^4 \sigma_k^2 + \sigma_k^2 c_j^2 \lambda^2}(t_i) = \frac{\sigma_k^4 c_j^2 \lambda^2 - \sigma_j^4 c_k^2 \lambda^2}{\sigma_j^2 (\sigma_j^4 \sigma_k^2 + \sigma_k^2 c_j^2 \lambda^2)}(t_i). \tag{22}$$

Since the denominator is strictly positive. Eq. 20 is therefore equivalent to:

$$\sigma_k^4 c_j^2 \lambda^2(t_i) > \sigma_j^4 c_k^2 \lambda^2(t_i), \tag{23}$$

which is in turn equivalent to $\frac{E_j}{E_k}(t_i) > 1$ by definition. The second proposition can be shown analogously. □

A consequence of this proposition is that a given ratio R_{jk} diminishes during every step, except for when it flips, i.e. $R_{jk}(t_i) > 1$ and $R_{jk}(t_{i+1}) < 1$. In Appendix Section A, we show that when training with constant learning rates, this can only happen during the first step. Now, we would like to find the precise learning rate where the ratio flips.

Definition 3.8 (Flipping Ratio)**.** We define the "flipping ratio" κ_{jk} of two layers j and k at a given time step t_i as:

$$\kappa_{jk}(t_i) := \frac{\sigma_j \sigma_k}{\sqrt{c_j c_k}}(t_i) = \sqrt{\frac{1}{E_j E_k}}(t_i). \tag{24}$$

Proposition 3.9 (Flipping Conditions)**.** *Let $j, k \leq L$ be any layer pair with w.l.o.g. be $R_{jk}(t_i) > 1$.*

1. *The effective learning rate ratio does not flip between time steps t_i and t_{i+1} i.e. $R_{jk}(t_{i+1}) > 1$ if and only if $\lambda(t_i) < \kappa_{jk}(t_i)$.*
2. *The effective learning rate ratio does flip between time steps t_i and t_{i+1} i.e. $R_{jk}(t_{i+1}) < 1$ if and only if $\lambda(t_i) > \kappa_{jk}(t_i)$.*
3. *The ratio $\frac{E_j}{E_k}$ has reached a stationary point at a given time step t_i, i.e. $R_{jk}(t_j) = R_{jk}(t_{j+1})$ for all $j \geq i$ if and only if $\lambda(t_i) = \kappa_{jk}(t_i)$.*

Proof. We start by showing the first proposition. Using the definition of R_{jk} and Eq. 21, we can write:

$$R_{jk}(t_{i+1}) = \frac{c_j \sigma_k^2}{c_k \sigma_j^2}(t_{i+1}) = \frac{\sigma_j^2 \sigma_k^4 c_j + \sigma_j^2 c_j c_k^2 \lambda^2}{\sigma_j^4 \sigma_k^2 c_k + \sigma_k^2 c_j^2 c_k \lambda^2}(t_i). \tag{25}$$

Thus, the condition $R_{jk}(t_{i+1}) > 1$ is equivalent to:

$$\left(\sigma_j^2 \sigma_k^4 c_j + \sigma_j^2 c_j c_k^2 \lambda^2\right)(t_i) > \left(\sigma_j^4 \sigma_k^2 c_k + \sigma_k^2 c_j^2 c_k \lambda^2\right)(t_i) \tag{26}$$

$$\Leftrightarrow \left(\sigma_j^2 c_j c_k^2 \lambda^2 - \sigma_k^2 c_j^2 c_k \lambda^2\right)(t_i) > \left(\sigma_j^4 \sigma_k^2 c_k - \sigma_j^2 \sigma_k^4 c_j\right)(t_i) \tag{27}$$

$$\Leftrightarrow \lambda^2 (\sigma_j^2 c_j c_k^2 - \sigma_k^2 c_j^2 c_k)(t_i) > \sigma_j^4 \sigma_k^2 c_k - \sigma_j^2 \sigma_k^4 c_j (t_i) \tag{28}$$

$$\Leftrightarrow c_j c_k \lambda^2 (\sigma_j^2 c_k - \sigma_k^2 c_j)(t_i) > \sigma_j^2 \sigma_k^2 (\sigma_j^2 c_k - \sigma_k^2 c_j)(t_i) \tag{29}$$

Since we assumed $R_{jk}(t_i) = \frac{E_j}{E_k}(t_i) = \frac{\sigma_k^2 c_j}{\sigma_j^2 c_k}(t_i) > 1$, it follows that $\left(\sigma_j^2 c_k - \sigma_k^2 c_j\right)(t_i) < 0$ and thus we invert the sign of the inequality when dividing by this quantity and we obtain the following equivalent condition:

$$\lambda^2(t_i) < \frac{\sigma_j^2 \sigma_k^2}{c_j c_k}(t_i). \tag{30}$$

All quantities are non-negative, therefore taking the square root preserves equivalence and we obtain the sought condition. The other propositions can be shown analogously. □

Since we are interested in reducing the highest overall ratio $R_{h\ell}(t_i)$ where ℓ, h are the layers with the lowest respective highest effective learning rate, we call $\kappa_{\ell h}(t_i)$ the *critrical learning rate*. When using higher learning rates than this value, $E_h(t_i)$ flips below $E_\ell(t_i)$ during the next step, thus (for high λ considerably) increasing total ELR spread. In the following we will come to understand that in practice, a more conservative choice is advisable; for this reason, we propose the *subcritical*, but still provably fast warm-up scheme below.

Corollary 3.10 (Subcritical Warm-Up)**.** *Given a network with $L > 0$ layers, if we schedule the learning rate as $\lambda(t_i) = \kappa_{hh'}(t_i)$, where we chose $h, h' \leq L$ at each step to be the two layers with the highest effective learning rates, then no ratio R_{jk} for any $j, k \leq L$ ever flips between a time step and the next and all pairs of effective learning rate ratios R_{jk} for any $j, k \leq L$ converge to 1 in L steps.*

Proof. Let h, h' be the two layers with the highest effective learning rates at time step t_i. By Proposition 3.9, if we chose $\lambda(t_i) = \kappa_{hh'}(t_i)$, we have $R_{hh'} = 1$ at the next time step t_{i+1} and for all further time steps. Since h, h' are the two layers with the highest effective learning rate at time step t_i and we chose $\lambda(t_i)$ to be equal to their flipping ratio $\kappa_{hh'}(t_i)$, we have:

$$\lambda(t_i) = \kappa_{hh'}(t_i) \leq \kappa_{jk}(t_i) \tag{31}$$

for all other layers $j, k \leq L$. Therefore, by Lemma 3.9, no ratio R_{jk} will ever flip between a time step and the next for any $j, k \leq L$. If we repeat this process L times, all pairwise learning rate ratios converge to 1. □

3.8. Simulating Warm-Up Schedulers and Criticality

In Figure 1, in order to visualize the concept of criticality, we simulated the evolution of effective learning rates and weight norms for popular learning rate schedulers with our discrete model (ref. Eq. 5), assuming initially exponentially exploding gradients (ref. Eq. 6). We also indicated $\lambda(t)$ along with the critical learning rate $\kappa_{\ell h}(t)$. As predicted by our analysis in Section 3.7, whenever $\lambda(t) > \kappa_{\ell h}(t)$, we see

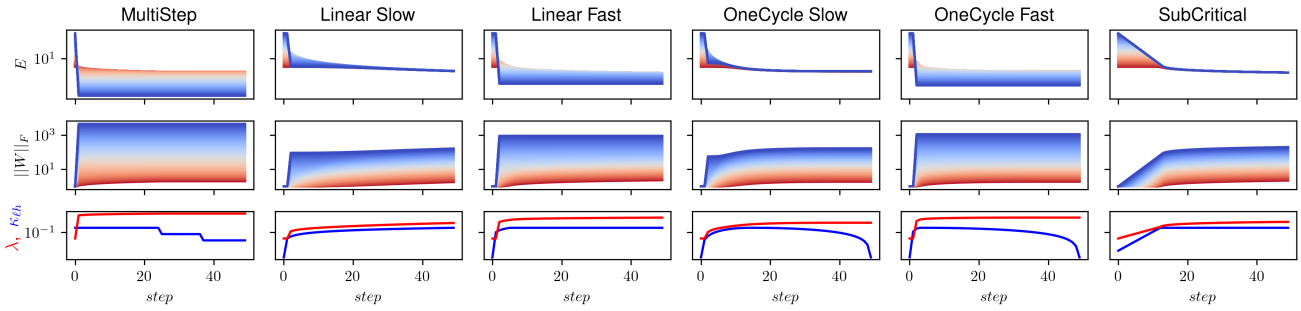

Figure 1. Simulated evolution of layer-wise effective learning rates **(top)**, weight norms **(middle)**, learning rates $\lambda(t)$ and critical LRs $\kappa_{\ell h}(t)$ **(bottom)** for popular learning rate schedulers. All y-axes are in logarithmic scale. Blue color corresponds to the lower layers.

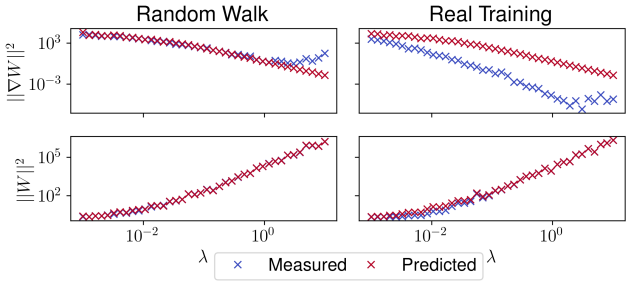

Figure 2. Short term evolution (after 10 steps) of predicted vs. measured weight and gradient norms of the lowest layer of a ResNet56 NoShort trained with random gradients **(left)** and real gradients **(right)** for various λ.

that the highest effective learning rate flips below the lowest, which in turn increases the ELR spread and consequently the convergence time. We conclude that whether a warm-up scheme succeeds in quickly reducing ELR spread highly depends on the chosen hyper-parameters.

4. Experimental Validation

In this experimental Section, we will first check the limitations of the assumption about constant base gradients and validate the predictivity of our model. Then, we will compare the predicted critical learning rate to an empirical value extracted from real training runs. Finally, we confirm that high ELR spreads correlate with network trainability in practice.

4.1. Experimental Setup

4.1.1. Architectures, Datasets and Training Protocols

We chose ResNet v1 (He et al., 2016a) with ("Short") and without ("NoShort") residual connections as examples of standard architectures. We chose a ResNet v1 as opposed to a v2 since in the former, the "correct" placement of normalization layers (ref. Section 3.1) is given without modifying the architecture. Theory predicts that a high number of layers and not using residual connections increase the strength of the observed effect (ref. Section 3.3). We therefore use 56 and 110 layer networks: Without residual connections, the former is deep but still trainable and the latter is mostly un-trainable with basic constant LR training. The final layer of a ResNet v1 is not scale-invariant and we therefore exclude it from our analysis. For computer vision tasks, we work with standard image classification datasets of variable difficulty: CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and ILSVRC 2012 (called ImageNet in the following) (Deng et al., 2009). We use the most basic training setting possible (vanilla SGD) and disable all possible factors that influence weight dynamics: momentum, weight decay, affine BatchNorm parameters and bias on linear layers (for a discussion, please refer to Appendix Section C). We further use different kinds of learning rate scheduling with and without warm-up; further details about the architectures and training process can be found in the Appendix.

4.1.2. Measuring ELR Spread

In our experiments, we need a measure for ELR spreads that is relative to the network's mean ELR.

Definition 4.1 (Relative Logarithmic ELR Spread)**.** We define the Relative Logarithmic ELR Spread as:

$$S_{rel} := \text{std}(\ln(E)), \tag{32}$$

computed across the layers of the network and usually averaged over all channels and the entire training process.

4.1.3. Random Walk

In the past section, we modeled exclusively the dynamics induced by normalization assuming constant base gradients (ref. Eq 4), meaning that the layer-wise expected gradient magnitude excluding normalization effects is constant

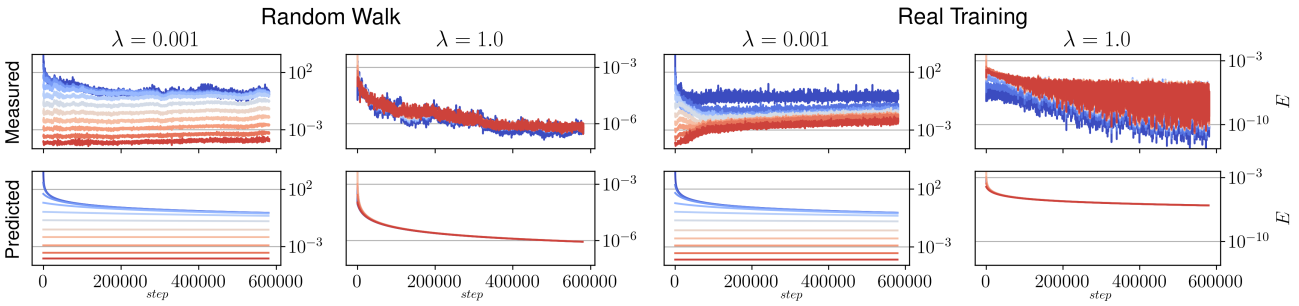

Figure 3. Long term evolution of predicted vs. measured layer-wise effective learning rates for a ResNet56 NoShort trained with random gradients **(left)** and real gradients **(right)**. Blue color corresponds to the lower layers.

over time. This is obviously not strictly true in a practical setting: apart from the obvious factors mentioned above (momentum, affine parameters etc.), the derivative of non-linear layers and the objective function itself changes when varying inputs or weights. During training, gradient norms tend to shrink as the objective function saturates (Lee et al., 2019). To verify that mostly learning effects are responsible for fluctuations in base gradients, we observe how weight dynamics evolve during a random walk.

Definition 4.2 (Random Walk)**.** During each training step, before applying the gradients computed in the backward-pass, we replace every layer's gradient by a random vector of similar norm which is also orthogonal to the layer's weights. Please refer to Algorithm 1 for a formal description.

Algorithm 1 Random Walk

Let e_ℓ denote the number of elements of the weight vector W_ℓ and $\langle \cdot, \cdot \rangle$ the dot product.

for each gradient descent step i **do**
  for each layer ℓ **do**
    Compute $\nabla W_\ell(t_i)$
    $\sigma \leftarrow \sqrt{\|\nabla W_\ell(t_i)\|_2^2 / e_\ell}$
    $R \leftarrow \mathcal{N}(0, \sigma^2)^{e_\ell}$
    $V \leftarrow W(t_i)$
    $\nabla W_\ell(t_i) \leftarrow R - \frac{\langle R,V \rangle}{\langle V,V \rangle} V$ // orthogonalize
  end for
end for

4.2. Model Validation and Limitations

To validate our theory, we measure the initial gradient and weight norms of a network and extrapolate their evolution using our discrete model (Eq. 5). We then compare the predicted weight/gradient norms to the empirically measured values after a given number of steps. We will first see that for a feedforward network with ReLU activations, it is already enough to exclude learning effects (random walk scenario) for our model to be long-term predictive. When including them, as expected, gradients are lower than predicted but the main takeaway qualitatively still holds: ELR spreads diminish over time, given that a certain learning rate is not exceeded.

Short-Term Validation : In Figure 2, we compare the measured weight/gradient norm of the lowest layer of a ResNet56 NoShort after training on Cifar10 for 10 steps to the values predicted using our discrete model on the initial values. In a random walk (left), predictions are quite accurate up to $\lambda \approx 1$ and get slightly inaccurate for higher λ, presumably due to numerical issues. As for real training (right), we see that gradients are notably smaller than expected after 10 steps. For the following, it is crucial to note that the difference between the predicted and measured values is not a constant ratio but instead increases in λ.

Long Term Validation : We conducted a similar experiment for only two different learning rates $\lambda \in \{0.001, 1\}$ over 3000 epochs and visualize the measured/predicted ELR of all layers in Figure 3. We see that in the random walk scenario, our prediction is remarkably accurate. In real training and for $\lambda = 1$, our model predicts this learning rate to be critical, but in reality it is super-critical as the gradients of the lower layers (blue) significantly undershoot with regard to the prediction and their ELR flips below the highest layers (red); further training does not seem to be able to recover the high ELR spread. Since training with any subcritical learning rate reduces ELR spread, we will see that it is sufficient to use a slightly lower λ than the predicted critical value to avoid an increase in ELR spread.

Predicting Criticality: In Figure 4, we train a ResNet110 NoShort for a single epoch using various constant learning rates on Cifar10 in a random walk and a real training scenario, tracking the evolution of ELR spreads. We plot ELR spreads at initialization and after one epoch, averaging measurements over 10 runs for each datapoint. First, we note that as predicted, up to a certain learning rate, ELR spreads are always lowered by training. Next, we indicate

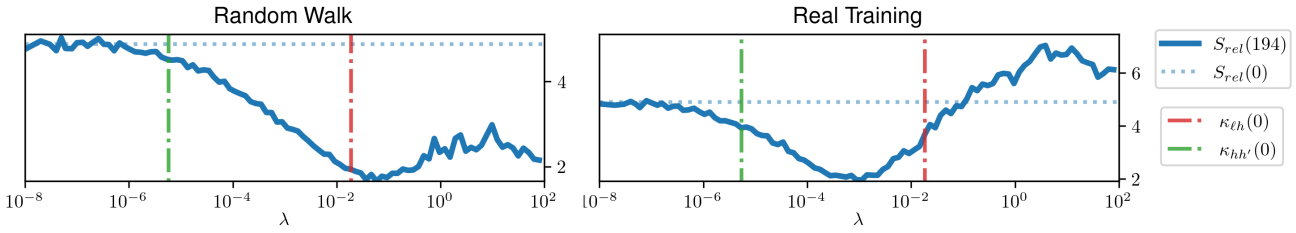

Figure 4. Relative spread after one epoch **(solid blue)**, relative spread at initialization **(dotted blue)** and the critical **(red)** / subcritical **(green)** learning rate at initialization of a ResNet110 NoShort with random gradients (**left**) and real gradients (**right**).

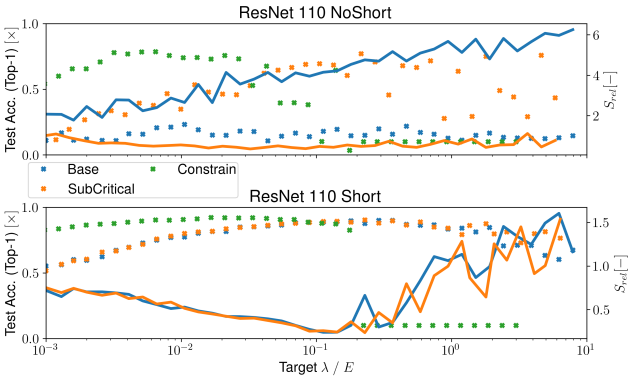

Figure 5. Test accuracies and relative spreads of a ResNet110 (No)Short trained on Cifar10 using regular, warm-up and constrained ELR training protocols for different target (E)LRs.

the predicted (sub)critical learning rate at initialization: as per Proposition A.1, if a learning rate is subcritical in step 0, it should also not increase spreads during later steps. Consequently, we expect the runs with $\lambda \approx \kappa_{\ell h}(0)$ to have the lowest S_{rel} value after training. In Figure 4, we see that this is indeed the case for the random walk. In real training, the qualitative behavior is similar but the curve is shifted to the left as gradients are smaller. We also note that in real training, when using super-critical learning rates ($\lambda > 10^{-3}$), ELRs do not seem to converge anymore, presumably to auto-rate tuning effects becoming too weak compared to fluctuations in base gradient magnitude caused by training.

4.3. ELR Spread and Trainability

In this section, we want to show empirically that networks with high ELR spreads correlate with low trainability and that lowering spreads using various methods can restore trainability. For this, we chose an experimental setting where ELR spreads are large: we train a ResNet110 (No)Short on Cifar10. For all runs, we use a simple multi-step learning rate decay. In Figure 5 (top), we see that for the "NoShort" networks in regular training without warm-up ("base"), spreads (averaged over the training run) are very high and trainability is very low. Using skip connections (bottom), spreads are much lower and the network is able to train.

4.3.1. Subcritical Warm-Up

As we have seen in Figure 4 (right), because of learning effects present in real training, the more conservative choice of using the subcritical learning rate for warm-up seems like a more sensible value to avoid overshooting in practice but still guarantees fast convergence in theory (ref. Corollary 3.10). Further, since we are using BatchNorm, we obtain channel-wise ELR values and use the maximum of these values as our layer-wise value.

4.3.2. Constraining Layer-Wise ELRs

Another possibility of controlling ELR spread is scaling each layer's gradients before each step so that layer-wise effective learning rates are constrained to be constant:

$$\nabla W \leftarrow \nabla W \cdot \frac{E_{goal}}{E + \epsilon}, \tag{33}$$

for a given constant goal effective learning rate E_{goal} and a small ϵ we chose as $\epsilon = 10^{-5}$. A similar mechanic was used in the popular LARS optimizer (You et al., 2017). To prevent increasing weight norms W from overflowing, we additionally divide all layer weights by the maximum layer weight $\widehat{W} = max(||W||_F)$ over all layers before every step. This should not change the network function since normalization layers are scale invariant and gradients are normalized. Alternatively, one could re-scale the gradients by $\frac{1}{\sqrt{1+\lambda^2}}$ after every optimization step, as described by Bernstein et al. (2020).

4.3.3. Evaluation

In Figure 5 (left), we see that both techniques lower ELR spread across layers which correlates with the previously untrainable ResNet110 NoShort becoming trainable, despite its initially exponentially exploding gradients. Although not a proof of a general causal connection between ELR spread and trainability, the fact that an untrainable network

becomes trainable with the intervention made yields some compelling evidence supporting such a hypothesis. For the network with skip-connections (right), we can observe the same effects but less pronounced, which is expected since the initial spread is linear and not exponential in the number of layers as shown in Section 3.3. In Appendix Section B, we repeat this experiment on the Cifar100 dataset and draw similar conclusions.

Finally, we train a ResNet101 (No)Short on ImageNet for 50 epochs using three different warm-up schedulers: OneCycle (Smith and Topin, 2017), sub-critical warm-up and no warm-up; we use the exact same cool-down phase (cosine) for all schedulers. We use default hyper-parameters for the OneCycle scheduler that work well in training a ResNet101 Short in a short amount of epochs on this dataset. In Table 1, we see that indeed warm-up lowers S_{rel} and correlates with increased performance, but the hyper-parameters used for OneCycle that work well with the residual network still result in significant spreads for the non-residual network with higher initial spreads. This confirms that warm-up should be scheduled as a function of current ELRs. Although subcritical warm-up uses very few warm-up steps, it yields comparable or better results than our preset OneCycle warm-up. Using the ELR-constrain method to prescribe a global ELR similar to the OneCycle run, we see that we are able to train the network without residual connections; using warm-up additionally decreases performance, showing that warm-up presents no benefits in a setting with no ELR spread.

Table 1. Test accuracies and relative spread of a ResNet101 (No)Short trained on ImageNet using different types of warm-up / ELR-constrain; *RES* indicates residual connections and *CTN* whether the ELR-constrain method was used.

RES	CTN	W. TYPE	W. STEPS	ACC.	S_{rel}
NO	NO	NONE	0	08.50	3.96
NO	NO	ONECYCLE	64060	22.82	1.96
NO	NO	SUBCRITICAL	9	41.83	0.70
NO	YES	NONE	0	47.99	-
NO	YES	ONECYCLE	64060	45.61	-
YES	NO	NONE	0	72.85	0.29
YES	NO	ONECYCLE	64060	72.83	0.31
YES	NO	SUBCRITICAL	3	73.06	0.27

5. Discussion and Future Work

In past work, high spreads in effective learning rates have been conjectured to negatively affect trainability, but to our best knowledge, no formal model exists that describes their time-based evolution in early training phases for scheduled learning rates. Under the assumption of constant gradient magnitudes beyond normalization effects, we derived a simple model from first principles that describes the evolution of expected weight/gradient norms and consequently effective learning rates during training. Under our model's assumption, we were able to prove that when training long enough using *any* constant learning rate, all ratios of layer-wise effective learning rates eventually converge to the same value. Problems can still arise in the first step(s) if the learning rate $\lambda(t_i)$ is bigger than the critical value $\kappa_{\ell h}(t_i)$ (which depends on current weight/gradient norms), momentarily increasing the disparity between layer-wise effective learning rates. We consider this theoretical model of normalization-induced dynamic effects to be our main contribution.

In a series of empirical experiments, we have shown that although we exclusively model norm-induced dynamics (scale-invariant linear layers) and assume that the expected gradient norm of other layers (objective function, nonlinear layers) does not change over time, our main takeaway still holds when training a deep convolutional ReLU network on real data: training reduces effective learning rate spread up to a certain *critical learning rate*. By using live gradient values at each step and using a slightly more conservative learning rate choice than predicted, we were able to design a hyper-parameter-free warm-up scheduler that is able to quickly reduce effective learning rate spreads in practice. In an (extreme) setting with exponentially exploding initial gradients, we show that reducing ELR spreads using warm-up or by normalizing gradients to prescribe a constant effective learning rate correlates with the network's trainability being restored.

Our analysis applies to all *normalized* networks, i.e. architectures where the network function is invariant wrt. scaling in weight matrices, which is usually the case in most normalized feedforward architectures. Unfortunately, unlike most other traditional MLPs/CNNs/ResNets, the weight matrices of attention blocks are not scale-invariant and thus the inverse scaling property (Eq. 1) and orthogonality (Eq. 2), which our model relies on, are violated. Moreover, modifying the architecture (i.e. adding additional normalization layers) would fundamentally impact its way of working (e.g. attention cannot be unlearnt anymore). Preliminary results show that for architectures containing higher degree nonlinearities (e.g. Transformer models), base gradients can vary much more compared to simple feedforward ReLU networks, therefore limiting the applicability of our model as is. If the order of the fluctuations of the base gradient exceeds that of the auto-rate tuning effects, the effect vanishes. We could envision extending our model to include an error analysis for non-constant base gradients, estimating when this is the case.

Acknowledgments

The authors acknowledge funding from the Emergent AI Center funded by the Carl-Zeiss-Stiftung. The authors would like to thank Daniel Franzen and Jan Disselhoff for their helpful discussions. The authors would also like to express their gratitude to the HPC working group of the Johannes-Gutenberg University Mainz for sharing their compute power in times of need.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Christian H. X. Ali Mehmeti-Göpel, David Hartmann, and Michael Wand. Ringing relus: Harmonic distortion analysis of nonlinear feedforward networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=TaYhv-q1Xit.

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rkxQ-nA9FX.

Muhammad Awais, Md. Tauhid Bin Iqbal, and Sung-Ho Bae. Revisiting internal covariate shift for batch normalization. *IEEE Trans. Neural Networks Learn. Syst.*, 32(11):5082–5092, 2021. doi: 10.1109/TNNLS.2020.3026784. URL https://doi.org/10.1109/TNNLS.2020.3026784.

Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f4b31bee138ff5f7b84ce1575a738f95-Abstract.html.

Johan Bjorck, Carla P. Gomes, Bart Selman, and Kilian Q. Weinberger. Understanding batch normalization. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7705–7716, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/36072923bfc3cf47745d704feb489480-Abstract.html.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071. PMLR, 2021. URL http://proceedings.mlr.press/v139/brock21a.html.

Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33:19964–19975, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015a. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015b. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016*

IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016a. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016b. doi: 10.1007/978-3-319-46493-0_38.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4246–4253. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://doi.org/10.18653/v1/2020.findings-emnlp.379.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2164–2174, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a0160709701140704575d499c997b6ca-Abstract.html.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8570–8581, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0d1a9651497a38d8b1c3871c84528bd4-Abstract.html.

Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rJg8TeSFDH.

Kyle Luther. Why batch norm causes exploding gradients. Blog post, 2020. URL https://kyleluther.github.io/2020/02/18/batchnorm-exploding-gradients.html.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d9fc0cdb67638d50f411432d0d41d0ba-Paper.pdf.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational*

Mathematics and Mathematical Physics, 4(5):1–17, 1964. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(64)90137-5. URL https://www.sciencedirect.com/science/article/pii/0041555364901375.

PyTorch. BatchNorm2d; PyTorch 2.1 documentation — pytorch.org. https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html. [Accessed 23-11-2023].

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, page 901, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/ed265bc903a5a097f61d3ec064d96d2e-Abstract.html.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2488–2498, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. *CoRR*, abs/2110.09456, 2021. URL https://arxiv.org/abs/2110.09456.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. URL http://arxiv.org/abs/1708.07120.

Ryszard Szwarc. Convergence analysis of a quotient of two sequences $x_{n+1}^2 = x_n^2 + \frac{c}{x_n^2}$. Mathematics Stack Exchange. URL https://math.stackexchange.com/q/4820434. URL:https://math.stackexchange.com/q/4820434 (version: 2023-12-05).

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017a. URL http://arxiv.org/abs/1706.05350.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017b. URL http://arxiv.org/abs/1706.05350.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 550–558, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Ruosi Wan, Zhanxing Zhu, Xiangyu Zhang, and Jian Sun. Spherical motion dynamics: Learning dynamics of normalized neural network using SGD and weight decay. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6380–6391, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/326a8c055c0d04f5b06544665d8bb3ea-Abstract.html.

Xiaoxia Wu, Rachel Ward, and Léon Bottou. Wngrad: Learn the learning rate in gradient descent. *CoRR*, abs/1803.02865, 2018. URL http://arxiv.org/abs/1803.02865.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the

transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR, 2020. URL http://proceedings.mlr.press/v119/xiong20b.html.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SyMDXnCcF7.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD batch size to 32k for imagenet training. *CoRR*, abs/1708.03888, 2017. URL http://arxiv.org/abs/1708.03888.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

A. Additional Proofs

This section contains proofs of theorems as well as additional material complementary to the main section.

Proof of Proposition 3.1. We compute the derivative of the output with regard to the input using the chain rule and relate it to the derivative with unscaled inputs:

$$\frac{dN}{dx}(x,\gamma W) = \frac{dN}{dL}(x,\gamma W)\cdot\frac{dL}{dx}(x,\gamma W) \tag{34}$$

$$= \frac{1}{\gamma}\frac{dN}{dL}(x,W)\cdot\gamma\frac{dL}{dW}(x,W) \tag{35}$$

$$= \frac{dN}{dx}(x,W). \tag{36}$$

Where in Eq. 35, we used the inverse scaling property from Eq. 1. □

Proof of Theorem 3.6. We would like to credit the original author of this proof (Szwarc). By setting $x_i \coloneqq \frac{\sigma_j^2(t_i)}{c_j\lambda}$ and $y_i \coloneqq \frac{\sigma_k^2(t_i)}{c_k\lambda}$, we can rewrite Eq. 4 for layers j and k as two sequences obeying the same recurrence relation:

$$x_{i+1} = x_i + \frac{1}{x_i} \tag{37}$$

$$y_{i+1} = y_i + \frac{1}{y_i}. \tag{38}$$

Raising x_i to the second power yields:

$$x_{i+1}^2 = x_i^2 + 2 + \frac{1}{x_i^2}. \tag{39}$$

This allows us to unroll the recursion as follows :

$$x_i^2 = x_1^2 + 2(i-1) + \frac{1}{u_1^2} + \ldots + \frac{1}{u_{i-1}^2}. \tag{40}$$

As $x_j \geq 2(j-1)$, we can write the following inequality:

$$2(i-1) \leq x_i^2 \leq 2(i-1) + x_1^2 + \frac{1}{x_1^2} + \frac{1}{2} + \frac{1}{4} + \ldots + 2(i-2). \tag{41}$$

By the integral test, it is clear that $\sum_{i=1}^{n-1}\frac{1}{k} \leq ln(n)$ and therefore $\sum_{i=1}^{n-1}\frac{1}{2k} \leq \frac{ln(n)}{2} = ln(\sqrt{n})$. Let be $\gamma \coloneqq u_1^2 + \frac{1}{u_1^2} - 2$, we consider the square root of the expression above:

$$\sqrt{2i-2} \leq x_i \leq \sqrt{2i + log(\sqrt{i-1}) + \gamma}. \tag{42}$$

Since γ is a constant and $\lim_{i\to\infty}\frac{log(i)}{i} = 0$, it follows that:

$$\lim_{i\to\infty}\frac{x_i}{\sqrt{2i}} = 1. \tag{43}$$

and analogously

$$\lim_{i\to\infty}\frac{y_i}{\sqrt{2i}} = 1. \tag{44}$$

We therefore obtain:

$$\lim_{i\to\infty}\frac{x_i}{y_i} = \frac{\sigma_j^2 c_k}{\sigma_k^2 c_j}(t_i) = R_{kj}(t_i) = 1, \tag{45}$$

which is in turn also true for the inverse fraction. □

Proposition A.1 (Ratios Flip at Most Once)**.** *Let $j,k \leq L$ be any layer pair with w.l.o.g. $R_{jk}(t_i) > 1$ and assume a constant learning rate $\lambda(t_i) = \lambda$.*

1. *If effective learning rate ratios do not flip between a given time step and the next, they will never flip at a later time step, i.e. if $R_{jk}(t_{i+1}) > 1$ it follows that $R_{jk}(t_{i+j}) > 1$ for all $j \geq 1$.*

2. *If effective learning rate ratios do flip between a given time step and the next, they will never flip again at a later time step,, i.e. if $R_{jk}(t_{i+1}) < 1$ it follows that $R_{jk}(t_{i+j}) < 1$ for all $j \geq 1$.*

Proof. We start by showing the first statement. Assuming that the effective learning rate ratio does not flip between time steps t_i and t_{i+1}, we know by Lemma 3.9 that $\lambda < \kappa_{jk}(t_i)$. We now just have to show that $\lambda < \kappa_{jk}$ for all successive time steps. Since c_j and c_k are constants and we know by the definition of the discrete process in Eq. 5 that all weight norms $\sigma(t_i)$ are strictly increasing over time, we can write:

$$\lambda < \kappa_{jk}(t_i) = \frac{\sigma_j\sigma_k}{\sqrt{c_j c_k}}(t_i) < \frac{\sigma_j\sigma_k}{\sqrt{c_j c_k}}(t_{i+j}) = \kappa_{jk}(t_{i+j}) \tag{46}$$

for all $j \geq 1$ and thus by Lemma 3.9 the ratio will never flip again.
We now show the second statement. Assuming that the effective learning rate ratio does flip between time steps t_i and t_{i+1}, we know by Lemma 3.9 that $\lambda > \kappa_{jk}(t_i)$. We start by showing that the ratio will not flip for the next time step, which is in turn equivalent to $\lambda < \kappa_{jk}(t_{i+1})$. We can

expand this term as follows:

$$\kappa_{jk}^2(t_{i+1}) = \frac{\sigma_j^2\sigma_k^2}{c_j c_k}(t_{i+1}) \tag{47}$$

$$= \frac{1}{c_j c_k}\left(\sigma_j^2 + \frac{c_j^2\lambda^2}{\sigma_j^2}\right)\left(\sigma_k^2 + \frac{c_k^2\lambda^2}{\sigma_k^2}\right)(t_i) \tag{48}$$

$$= \left(\frac{\sigma_j^2\sigma_k^2}{c_j c_k} + \frac{\sigma_j^2 c_k\lambda^2}{\sigma_k^2 c_j} + \frac{\sigma_k^2 c_j\lambda^2}{\sigma_j^2 c_k} + \frac{c_j c_k\lambda^4}{\sigma_j^2\sigma_k^2}\right)(t_i) \tag{49}$$

$$= \left(\kappa_{jk}^2 + \frac{E_k}{E_j}\lambda^2 + \frac{E_j}{E_k}\lambda^2 + \frac{1}{\kappa_{jk}^2}\lambda^4\right)(t_i) \tag{50}$$

$$> \left(\kappa_{jk}^2 + \frac{E_k}{E_j}\lambda^2 + \frac{E_j}{E_k}\lambda^2 + \frac{1}{\lambda^2}\lambda^4\right)(t_i) \tag{51}$$

$$\geq \lambda^2. \tag{52}$$

We can write Eq. 51 because of the assumption that $\lambda > \kappa_{jk}(t_i)$ and Eq. 52 because all summands are non-negative. We have therefore shown that $\lambda < \kappa_{jk}(t_{i+1})$ and thus the ratio will not flip between time step t_{i+1} and t_{i+2}. By the first proposition shown above, we know it will therefore never flip in future time steps, i.e. $R_{jk}(t_{i+j}) < 1$ for all $j \geq 1$. □

B. Additional Experiments

In Figure 6, we repeated the experiment of Figure 5 on the Cifar100 dataset. Qualitatively, we observe the same effects. The ResNet110 NoShort does not train at all and has high ELR spreads. By using the ELR-constrain or critical-warmup method, we are able to train the network to a significant, but not very good performance. As for the ResNet110 Short, we start to see a difference between runs without warm-up our sub-critical scheduler for high learning rates $\lambda > 10$) where again, a reduced spread results in increased trainability. We conclude that reducing ELR spread correlates with increased trainability, but other factors (e.g. vanishing dimensionalty) explain the gap between short and no-short architectures.

C. Other Factors Influencing Weight Dynamics

As mentioned in the main paper, some techniques commonly used in training influence the evolution of weight dynamics in a way that is not modeled by Eq. 5; in this section we will discuss them.

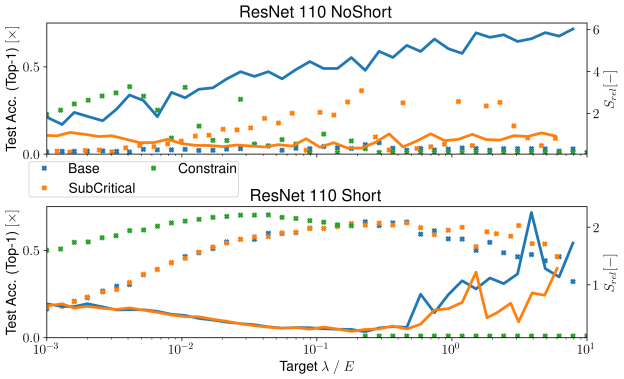

Figure 6. Test accuracies and relative spread of a ResNet110 (No)Short trained on Cifar100 using regular, warm-up and constrained ELR training protocols.

C.1. Weight Decay

The fact that weight decay influences weight dynamics in normalized networks is quite trivial and well-explored in recent literature: (Hoffer et al., 2018) (van Laarhoven, 2017b). In a normalized network, if all weights are reduced by a factor α, this corresponds to an increase of the global learning rate by a factor α, as per Eq 1.

C.2. Momentum

As momentum SGD (Polyak, 1964) modifies each gradient's direction and length before it is applied, it is easy to see that it must influence weight dynamics. It is possible to compute weight dynamics of a network optimized with momentum SGD, but we consider this to be out of scope of this work.

C.3. Affine Normalization Parameters

Normalization layers are usually applied with learnable affine parameters $\gamma \cdot N(x) + \beta$ that are initialized to $\gamma = 1$ and $\beta = 0$ (PyTorch). In the case of a network where normalization layers are followed by ReLUs (this is the case in our experiments), this means that we initialize in the "maximum curvature region" of the nonlinearity but drift away from it during training (Ali Mehmeti-Göpel et al., 2021) leading to gradients dropping further than expected. In Figure 7, we repeated the experiment of Figure 3 using random gradients (a setting that produces a reliable prediction) but add affine BatchNorm parameters to our training protocol. For $\lambda = 0.001$, the prediction is still quite accurate but for $\lambda = 1$, we see that the real gradients are much smaller than predicted.

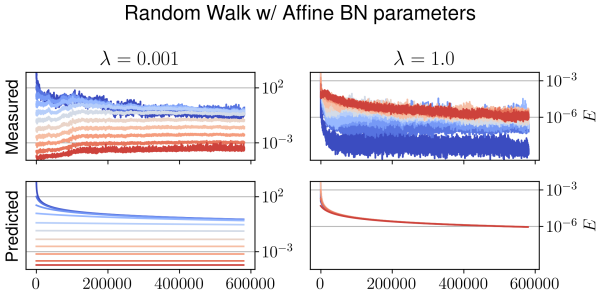

Figure 7. Long term evolution of simulated/real layer-wise effective learning rates for a ResNet56 NoShort trained with random gradients and affine BatchNorm parameters.

D. Architecture and Training Details

As described in the main paper, we used ResNet variants with varying hyper-parameters, with and without skip connections. Architectural details can be found in the tables below.

The experiments in the paper were made on computers running Arch Linux, Python 3.11.5, PyTorch Version 2.1.2+cu121. Various Nvidia GPUS were used ranging from GeForce GTX 1080TI, GeForce GTX 2080Ti RTX 4090.

Table 2. Network architecture and training regime used for the Cifar10/100 task.

ARCHITECTURE	RESNET56/110
BLOCK	BASICBLOCK V1
NUM. BLOCKS	9 9 9 / 18 18 18
NUM. PLANES	16 32 64
SHORTCUT TYPE	A (PADDING)

TRAINING	CIFAR-10 / CIFAR-100
EPOCHS	200
SCHEDULER	MULTISTEP ($\gamma = 0.1$)
MILESTONES	100, 150
LEARNING RATE	VARIABLE
BATCH SIZE	256
OPTIMIZER	SGD
MOMENTUM	0
WEIGHT DECAY	0
AUGMENTATION	RANDOM FLIP
NESTEROV	FALSE

Table 3. Network architecture and training regime used for the ImageNet task.

ARCHITECTURE	RESNET101
BLOCK	BOTTLENECKBLOCK V1
NUM. BLOCKS	3 4 32 3
NUM. PLANES	64 128 256 512
SHORTCUT TYPE	B (1X1-CONV+BN)

TRAINING	IMAGENET
EPOCHS	50
SCHEDULER	ONECYCLE/ NO-WARMUP + COSINE/ SUBCRITICAL + COSINE
MAX. LR	0.4
BATCH SIZE	100
OPTIMIZER	SGD
NESTEROV	FALSE
MOMENTUM	0
WEIGHT DECAY	0
AUGMENTATION	RANDOM FLIP
ONECYCLE ANNEALSTRATEGY	COSINE
ONECYCLE BASEMOMENTUM	0
ONECYCLE CYCLEMOMENTUM	TRUE
ONECYCLE DIVFACTOR	20
ONECYCLE EPOCHSSTART	0.1
ONECYCLE FINALDIVFACTOR	2000
ONECYCLE MAXMOMENTUM	0.0