Step Size Matters in Deep Learning

Kamil Nar **S. Shankar Sastry**
Electrical Engineering and Computer Sciences
University of California, Berkeley
{nar,sastry}@eecs.berkeley.edu

Abstract

Training a neural network with the gradient descent algorithm gives rise to a discrete-time nonlinear dynamical system. Consequently, behaviors that are typically observed in these systems emerge during training, such as convergence to an orbit but not to a fixed point or dependence of convergence on the initialization. Step size of the algorithm plays a critical role in these behaviors: it determines the subset of the local optima that the algorithm can converge to, and it specifies the magnitude of the oscillations if the algorithm converges to an orbit. To elucidate the effects of the step size on training of neural networks, we study the gradient descent algorithm as a discrete-time dynamical system, and by analyzing the Lyapunov stability of different solutions, we show the relationship between the step size of the algorithm and the solutions that can be obtained with this algorithm. The results provide an explanation for several phenomena observed in practice, including the deterioration in the training error with increased depth, the hardness of estimating linear mappings with large singular values, and the distinct performance of deep residual networks.

1 Introduction

The depth of a neural network determines the size of the class of functions that it can represent. As the depth is increased, this class of functions expands provided that the new layers are able to express the identity mapping. Therefore, the minimum training error that can be achieved by a network diminishes as its depth is increased. However, the training error of most neural networks degrades in practice once the number of layers exceeds a certain value; and deeper networks start to perform worse than their shallower counterparts, as shown in Figure 1 (He et al., 2016). This deterioration in the training error with increased depth indicates a problem with the method used for training the neural network; namely, a problem with the convergence of the gradient descent algorithm.

When gradient descent is used to minimize a function, say $f : \mathbb{R}^n \to \mathbb{R}$, it leads to a discrete-time dynamical system:

$$x[k+1] = x[k] - \delta \nabla f(x[k]), \tag{1}$$

where $x[k]$ is *the state* of the system, which consists of the parameters updated by the algorithm, and δ is the step size of the algorithm. Every fixed point of the system (1) is called *an equilibrium* of the system, and they correspond to the critical points of the function f.

Unless f is a quadratic function of the parameters, the system described by (1) is either a nonlinear system or a hybrid system that switches from one dynamics to another over time. Consequently, the system (1) can exhibit behaviors that are typically observed in nonlinear and hybrid systems, such as convergence to an orbit but not to a fixed point, or dependence of the convergence on the initialization. The step size of the gradient descent algorithm has a critical effect on these behaviors, as shown in the following two examples.

Preprint. Work in progress.

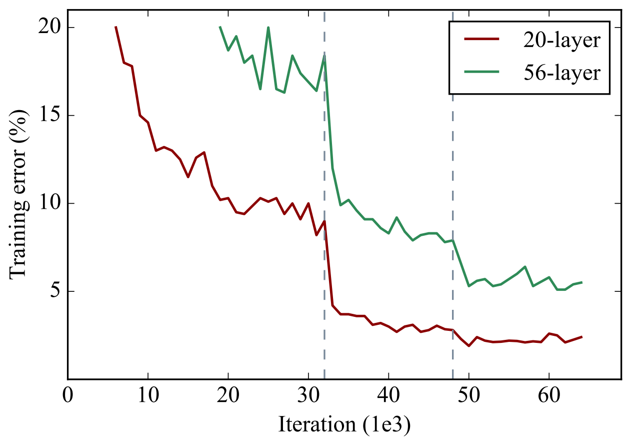

Figure 1: Training error of a 20-layer and a 56-layer neural network on CIFAR-10 dataset. The deeper network leads to a significantly larger training error. **Step size of the gradient descent is divided by 10 at iterations 32k and 48k**; these instants are shown with the dashed lines. Right after the step size is decreased, the training error plummets. It cannot be the case that the parameters are slowly converging to a local optimum before the step size is changed, nor can they be stuck near a local optimum or a saddle point, because decreasing the step size would have further slowed down the convergence in those cases. This behavior can only be explained by the fact that the initial step size is too large for certain regions in the parameter space and the parameters keep oscillating around a local optimum until the step size is changed. Once the step size is decreased, the magnitude of the oscillations around the local optimum diminishes, and so does the training error. The figure is adapted from (He et al., 2016).

Example 1. Convergence to a periodic orbit: Consider the continuously differentiable and convex function $f(x) = \frac{2}{3}|x|^{3/2}$, which has a unique local minimum at the origin. The gradient descent algorithm on this function yields

$$x[k+1] = \begin{cases} x[k] - \delta\sqrt{x[k]}, & x[k] \geq 0, \\ x[k] + \delta\sqrt{-x[k]}, & x[k] < 0. \end{cases}$$

As expected, the origin is the only equilibrium of this system. Interestingly, however, $x[k]$ converges to the origin only when the initial state $x[0]$ belongs to a countable set $\mathcal{S}$:

$$\mathcal{S} = \left\{0, \delta^2, -\delta^2, \dots\right\}.$$

For all other initializations, $x[k]$ converges to an oscillation between $\delta^2/4$ and $-\delta^2/4$. This implies that, if the initial state $x[0]$ is randomly drawn from a continuous distribution, then almost surely, $x[k]$ does not converge to the origin, yet $|x[k]|$ converges to $\delta^2/4$. In other words, with probability 1, the state $x[k]$ does not converge to a fixed point, such as a local optimum or a saddle point, even though the estimation error converges to a finite non-optimal value.

Example 2. Dependence of convergence on the initialization: Consider the function $f(x) = x^L$ where $L \in \mathbb{N}$ is an even number larger than 2. The gradient descent results in the system

$$x[k+1] = x[k] - \delta L x[k]^{L-1}.$$

The state $x[k]$ converges to the origin if the initial state satisfies $x[0]^{L-2} < (2/L\delta)$ and $x[k]$ diverges if $x[0]^{L-2} > (2/L\delta)$.

These two examples demonstrate:

1. the convergence of training error does not imply the convergence of the gradient descent algorithm to a local optimum or a saddle point,
2. the step size determines the magnitude of the oscillations if the algorithm converges to an orbit but not to a fixed point,
3. the step size influences the convergence of the algorithm differently for each initialization.

Note that **these are consequences of the nonlinear dynamics of the algorithm and not of the (non)convexity of the function to be minimized**. While both of the functions used in the examples are convex, the identical behaviors are observed during the minimization of nonconvex cost functions of neural networks as well. In fact, only these behaviors can provide a satisfactory explanation for a phenomenon observed in Figure 1: right after the step size of the algorithm is decreased, the training error plummets. It cannot be the case that the parameters are slowly converging to an equilibrium right before the step size is changed, nor can they be stuck near a local optimum or a saddle point, because if either were the case, decreasing the step size would have further slowed down the convergence. These sharp falls can only be explained by the fact that the initial step size is too large for some regions in the parameter space, and the parameters are oscillating around a local optimum right before the step size is changed. Once the step size is decreased, the radius of the oscillations around the equilibrium point diminishes, the distance to the equilibrium point in the parameter space falls sharply, and consequently, so does the training error.

While training a deep neural network, the dynamical system created by the gradient descent will usually have multiple equilibria, which coincide with the critical points of the training cost function. Convergence to these equilibria is in general affected unequally by the step size. For example, for a given step size, the algorithm might be able to converge to a subset of the local optima but not to the others independent of the initializations. Therefore, the step size also plays a critical role in understanding why some solutions are more likely to be obtained instead of the others when the gradient descent algorithm is used.

1.1 Our contributions

In this paper, we study the gradient descent algorithm as a discrete-time dynamical system during training deep neural networks, and we show the relationship between the step size of the algorithm and the solutions that can be obtained with this algorithm. In particular, we achieve the following:

1. We highlight that one of the reasons the training error stops decreasing during training is the fact that the step size of the algorithm becomes larger than it should be for certain regions in the parameter space, and the parameters keep oscillating around a local optimum rather than slowly converge to it or be stuck near a saddle point. In particular, for every fixed step size, there is a significant possibility that the algorithm converges to an orbit instead of a critical point of the training cost function.
2. We analyze the Lyapunov stability of the gradient descent algorithm on deep linear networks and find different upper bounds on the step size to enable convergence to each solution. We show that for every step size, the algorithm can converge to only a subset of the local optima, and there are always some local optima that the algorithm cannot converge to independent of the initialization.
3. We show that symmetric positive definite matrices can be estimated with a deep linear network by initializing the weight matrices as the identity, and this initialization allows the use of the largest step size. Conversely, the gradient descent is most likely to converge for an arbitrarily chosen step size if the weight matrices are initialized as the identity.
4. We show that symmetric matrices with negative eigenvalues cannot be estimated with the identity initialization, and the gradient descent converges to the closest positive semidefinite matrix in the Frobenius norm.
5. For 2-layer neural networks with ReLU activations, we obtain an explicit relationship between the step size of the gradient descent and the output of the solution that the algorithm can converge to.

1.2 Related work

It is a well-known problem that the gradient of the training cost function can become disproportionate for different parameters when training a neural network. Several works in the literature tried to address this problem. For example, changing the geometry of optimization was proposed in (Neyshabur et al., 2017) and a regularized descent algorithm was proposed to prevent the gradients from exploding and vanishing during training.

Deep residual networks, which is a specific class of neural networks, yielded exceptional results in practice with their peculiar structure (He et al., 2016). By keeping each layer of the network close to the identity function, these networks were able to attain lower training and test errors as the depth of the network was increased. To explain their distinct behavior, the training cost function of their linear versions was shown to possess some crucial properties (Hardt & Ma, 2016). Later, equivalent results were also derived for nonlinear residual networks under some conditions (Bartlett et al., 2018a).

The effect of the step size on training neural networks was empirically investigated in (Daniel et al., 2016). A step size adaptation scheme was proposed in (Rolinek & Martius, 2018) for the stochastic gradient method and shown to outperform the training with a constant step size. Similarly, some heuristic methods with variable step size were introduced and tested empirically in (Magoulas et al., 1997) and (Jacobs, 1988).

Two-layer linear networks were first studied in (Baldi & Kornik, 1989). The analysis was extended to deep linear networks in (Kawaguchi, 2014), and it was shown that all local optima of these networks were also the global optima. It was discovered in (Hardt & Ma, 2016) that the only critical points of these networks were actually the global optima as long as all layers remained close to the identity function during training. The dynamics of training these networks were also analyzed in (Saxe et al., 2013) and (Gunasekar et al., 2017) by assuming an infinitesimal step size and using a continuous-time approximation to the dynamics.

Lyapunov analysis from the dynamical system theory (Khalil, 2002; Sastry, 1999), which is the main tool for our results in this work, was used in the past to understand and improve the training of neural networks – especially that of the recurrent neural networks (Michel et al., 1988; Matsuoka, 1992; Barabanov & Prokhorov, 2002). State-of-the-art feedforward networks, however, have not been analyzed from this perspective.

We summarize the **major differences** between our contributions and the previous works as follows:

1. We relate the vanishing and exploding gradients that arise during training **feedforward** networks to the Lyapunov stability of the gradient descent algorithm.
2. Unlike the continuous-time analyses given in (Saxe et al., 2013) and (Gunesekar et al., 2017), we study the discrete-time dynamics of the gradient descent with an emphasis on the step size. By doing so, we obtain upper bounds on the step size to be used, and we show that the step size restricts the set of local optima that the algorithm can converge to. Note that these results cannot be obtained with a continuous-time approximation.
3. For deep linear networks with residual structure, (Hardt & Ma, 2016) shows that the gradient of the cost function cannot vanish away from a global optimum. This is not enough, however, to suggest the fast convergence of the algorithm. Given a fixed step size, the algorithm may also converge to an oscillation around a local optimum, as in the case of Example 1. We rule out this possibility and provide a step size so that the algorithm converges to a global optimum with a linear rate.
4. We recently found out that the convergence of the gradient descent algorithm was also studied in (Bartlett et al., 2018b) for symmetric positive definite matrices independently of and concurrently with our preliminary work (Nar & Sastry, 2018). However, unlike (Bartlett et al., 2018b), we explicitly give a step size value for the algorithm to converge with a linear rate, and we emphasize the fact that the identity initialization allows convergence with the largest step size.

2 Upper bounds on the step size for training deep linear networks

Deep linear networks are a special class of neural networks that do not contain nonlinear activations. They represent a linear mapping and can be described by a multiplication of a set of matrices:

$$W_L W_{L-1} \cdots W_2 W_1,$$

where $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ for each $i \in [L] := \{1, 2, \ldots, L\}$. Even though the functions they represent are linear, their training cost is not a quadratic function of the parameters, and therefore, the dynamics of the gradient descent is always nonlinear during training of these networks. For this reason, they provide a simple model to study some of the nonlinear behaviors observed during training.

The training cost functions of the deep linear networks always have multiple optima. In fact, given a cost function $\mathcal{L}\left(W_L W_{L-1} \cdots W_2 W_1\right)$, if the point $\{\hat{W}_i\}_{i\in[L]}$ is a local minimum, then $\{\alpha_i \hat{W}_i\}_{i\in[L]}$ is also a local minimum for every set of scalars $\{\alpha_i\}_{i\in[L]}$ satisfying $\alpha_1\alpha_2\cdots\alpha_L = 1$. Consequently, independent of the function $\mathcal{L}$, none of the local optima is isolated in the parameter space, and the cost function is not strongly convex at any point.

Although multiple local optima attain the same training cost for deep linear networks, the dynamics of the gradient descent algorithm exhibits distinct behaviors around these points. In particular, the step size required to render each of these local optima *stable in the sense of Lyapunov* is very different. Since the Lyapunov stability of a point is a necessary condition for the convergence of the algorithm to that point, the step size that allows convergence to each solution is also different, which is formalized in Theorem 1.

Theorem 1. *Given a nonzero matrix $R \in \mathbb{R}^{n_L \times n_0}$ and a set of points $\{x_i\}_{i\in[N]}$ in $\mathbb{R}^{n_0}$ that satisfy $\frac{1}{N}\sum_{i=1}^N x_i x_i^\top = I$, assume that R is estimated as a multiplication of the matrices $\{W_j\}_{j\in[L]}$ by minimizing the squared error loss*

$$\frac{1}{2N}\sum_{i=1}^N \left\|Rx_i - W_L W_{L-1}\dots W_2 W_1 x_i\right\|_2^2 \tag{2}$$

where $W_j \in \mathbb{R}^{n_j \times n_{j-1}}$ for all $j \in [L]$. Then the gradient descent algorithm can converge to a solution $\{\hat{W}_j\}_{j\in[L]}$ only if the step size δ satisfies

$$\delta \le \frac{2}{\sum_{j=1}^L p_{j-1}^2 q_{j+1}^2} \tag{3}$$

where

$$p_j = \left\|\hat{W}_j \cdots \hat{W}_2 \hat{W}_1 v\right\|, \quad q_j = \left\|u^\top \hat{W}_L \hat{W}_{L-1} \cdots \hat{W}_j\right\| \quad \forall j \in [L],$$

and u and v are the left and right singular vectors of $\hat{R} = \hat{W}_L \cdots \hat{W}_1$ corresponding to its largest singular value.

Considering all the solutions $\{\alpha_i \hat{W}_i\}_{i\in[L]}$ that satisfy $\alpha_1\alpha_2\cdots\alpha_L = 1$, the bound in (3) can be arbitrarily small for some of the local optima. Therefore, given a fixed step size δ, the gradient descent can converge to only a subset of the local optima, and there are always some solutions that the gradient descent cannot converge to independent of the initialization.

Remark 1. Theorem 1 provides a necessary condition for convergence to a specific solution. It does **not** state that given a step size δ, the algorithm converges to a solution which satisfies (3). It only rules out the possibility of converging to a large subset of the local optima. It might be the case, for example, that the algorithm converges to an oscillation around a local optimum which violates (3) even though there are some other local optima which satisfy (3).

As a necessary condition for the convergence to a global optimum, we can also find an upper bound on the step size independent of the weight matrices of the solution, which is given next.

Corollary 1. *For the minimization problem in Theorem 1, the gradient descent with step size δ cannot converge to a global optimum unless the step size δ satisfies*

$$\delta \le \frac{2}{L\rho(R)^{2(L-1)/L}} \tag{4}$$

where $\rho(R)$ is the largest singular value of R.

Remark 2. Corollary 1 shows that, unlike the optimization of the ordinary least squares problem, the step size required for the convergence of the algorithm depends on the parameter to be estimated, R. Consequently, estimating linear mappings with larger singular values requires the use of a smaller step size. Moreover, the step size used during training gives information about the solution obtained if the algorithm converges, which is stated next.

Corollary 2. *Assume that the gradient descent with step size δ has converged to a local optimum $\hat{R} = \hat{W}_L \dots \hat{W}_1$ for the minimization problem in Theorem 1. Then the largest singular value of $\hat{R}$ satisfies*

$$\rho(\hat{R}) \le \left(\frac{2}{L\delta}\right)^{L/(2L-2)}.$$

3 Identity initialization allows the largest step size for estimating symmetric positive definite matrices

Corollary 1 provides only a necessary condition for the convergence of the algorithm, and the bound (4) is not tight for every estimation problem. However, if the matrix to be estimated is symmetric and positive definite, the algorithm can converge to a solution with step sizes close to (4), which requires a specific initialization of the weight parameters.

Theorem 2. *Assume that $R \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix, and given a set of points $\{x_i\}_{i\in[N]}$ which satisfy $\frac{1}{N}\sum_{i=1}^N x_i x_i^\top = I$, the matrix R is estimated as a multiplication of the square matrices $\{W_j\}_{j\in[L]}$ by minimizing*

$$\frac{1}{2N}\sum_{i=1}^N \|Rx_i - W_L \dots W_1 x_i\|_2^2.$$

If the weight parameters are initialized as $W_i[0] = I$ for all $i \in [L]$ and the step size satisfies

$$\delta \le \min\left\{\frac{1}{L}, \frac{1}{L\rho(R)^{2(L-1)/L}}\right\},$$

then each W_i converges to $R^{1/L}$ with a linear rate.

Remark 3. Theorem 2 shows that the algorithm converges to a global optimum despite the nonconvexity of the optimization, and it provides a case where the bound (4) is tight. The tightness of the bound implies that for the same step size, most of the other global optima are *unstable in the sense of Lyapunov*, and therefore, the algorithm cannot converge to them independent of the initialization. Consequently, using identity initialization allows convergence to a solution which is most likely to be *stable in the sense of Lyapunov* for an arbitrarily chosen step size.

Remark 4. The fact that the identity initialization allows convergence to a global optimum even for large step sizes is remarkable. Given that the identity initialization on deep linear networks is equivalent to the zero initialization of linear residual networks (Hardt & Ma, 2016), Theorem 2 provides an explanation for the exceptional performance of residual networks as well (He et al., 2016).

When the matrix to be estimated is symmetric but not positive semidefinite, the bound (4) is still tight for some of the global optima. In this case, however, the eigenvalues of the estimate cannot attain negative values if the weight matrices are initialized with the identity.

Theorem 3. *Let $R \in \mathbb{R}^{n\times n}$ in Theorem 2 be a symmetric matrix such that the minimum eigenvalue of R, $\lambda_{\min}(R)$, is negative. If the weight parameters are initialized as $W_i[0] = I$ for all $i \in [L]$ and the step size satisfies*

$$\delta \le \min\left\{\frac{1}{1-\lambda_{\min}(R)}, \frac{1}{L}, \frac{1}{L\rho(R)^{2(L-1)/L}}\right\},$$

then the estimate $\hat{R} = \hat{W}_L \cdots \hat{W}_1$ converges to the closest positive semidefinite matrix to R in Frobenius norm.

4 Effect of step size on training two-layer networks with ReLU activations

In Section 2, we analyzed the relationship between the step size of the gradient descent algorithm and the solutions that can be obtained by training deep linear networks. A similar relationship exists for nonlinear networks as well. The following theorem, for example, provides an upper bound on the step size for the convergence of the algorithm when the network has two layers and ReLU activations.

Theorem 4. *Given a set of points $\{x_i\}_{i\in[N]}$ in $\mathbb{R}^n$, let a function $f : \mathbb{R}^n \to \mathbb{R}^m$ be estimated by a two-layer neural network with ReLU activations by minimizing the squared error loss:*

$$\min_{W,V} \frac{1}{2}\sum_{i=1}^N \|Wg(Vx_i - b) - f(x_i)\|_2^2,$$

where $g(\cdot)$ is the ReLU function, $b \in \mathbb{R}^r$ is the fixed bias vector, and the optimization is only over the weight parameters $W \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$. If the gradient descent algorithm with step size δ converges to a solution $(\hat{W}, \hat{V})$, then the estimate $\hat{f}(x) = \hat{W}g(\hat{V}x - b)$ satisfies

$$\max_{i\in[N]} \|x_i\|_2 \|\hat{f}(x_i)\|_2 \le \frac{1}{\delta}.$$

Theorem 4 shows that if the algorithm is able to converge with a large step size, then the estimate $\hat{f}(x)$ must have a small magnitude for large values of $\|x\|$.

Similar to Corollary 1, the bound given by Theorem 4 is not necessarily tight. Nevertheless, it highlights the effect of the step size on the convergence of the algorithm. To demonstrate that small changes in the step size could lead to significantly different solutions, we generated a piecewise continuous function $f : [0, 1] \to \mathbb{R}$ and estimated it with a two-layer network by minimizing

$$\sum\nolimits_{i=1}^{N} |Wg(Vx_i - b) - f(x_i)|^2$$

with two different step sizes $\delta \in \{2 \cdot 10^{-4}, 3 \cdot 10^{-4}\}$, where $W \in \mathbb{R}^{1\times 20}$, $V \in \mathbb{R}^{20}, b \in \mathbb{R}^{20}$, $N = 1000$ and $x_i = i/N$ for all $i \in [N]$. The initial values of W, V and the constant vector b were all drawn from independent standard normal distributions; and the vector b was kept the same for both of the step sizes used. As shown in Figure 2, training with $\delta = 2 \cdot 10^{-4}$ converged to a fixed solution, which provided an estimate $\hat{f}$ close the original function f. In contrast, training with $\delta = 3 \cdot 10^{-4}$ converged to an oscillation and not to a fixed point. That is, after sufficient training, the estimate kept switching between $\hat{f}_{\text{odd}}$ and $\hat{f}_{\text{even}}$ at each iteration of the gradient descent.

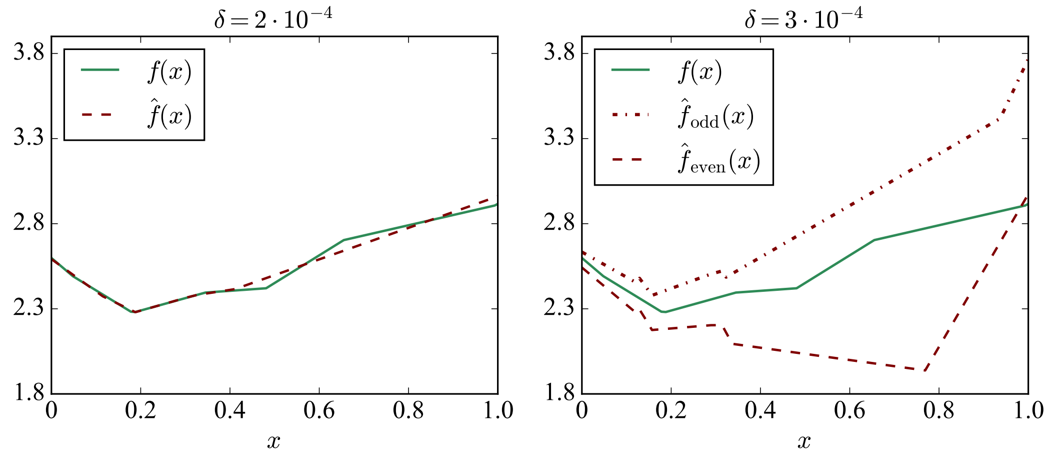

Figure 2: Estimates of the function f obtained by training a two-layer neural network with two different step sizes. **[Left]** When the step size of the gradient descent is $\delta = 2 \cdot 10^{-4}$, the algorithm converges to a fixed point, which provides an estimate $\hat{f}$ close to f. **[Right]** When the step size is $\delta = 3 \cdot 10^{-4}$, the algorithm converges to an oscillation and not to a fixed solution. That is, after sufficient training, the estimate keeps switching between $\hat{f}_{\text{odd}}$ and $\hat{f}_{\text{even}}$ at each iteration.

5 Discussion

When gradient descent is used to minimize a function, typically only three possibilities are considered: convergence to a local optimum, to a global optimum, or to a saddle point. In this work, we considered the fourth possibility: the gradient descent may not converge at all – even in the deterministic setting. The training error may not reflect the oscillations in the dynamics, or when a stochastic optimization method is used, the oscillations in the training error might be wrongly attributed to the stochasticity of the algorithm. We underlined that, if the training error of an algorithm converges to a non-optimal value, that does not imply the algorithm is stuck near a bad local optimum or a saddle point; it might simply be the case that the algorithm has not converged at all.

We showed that step size of the gradient descent influences the dynamics of the algorithm substantially. It renders some of the local optima unstable in the sense of Lyapunov, and the algorithm cannot converge to these points independent of the initialization. It also determines the magnitude of the oscillations if the algorithm converges to an orbit around an equilibrium point in the parameter space.

In Corollary 2 and Theorem 4, we showed that the step size required for convergence to a specific solution depends on the solution itself. This reveals that some solutions, such as linear functions with large singular values, are harder to converge to. Given that there exists a relationship between the Lipschitz constants of the estimated functions and their generalization error (Bartlett et al., 2017), this result could provide a better understanding of the generalization of deep neural networks.

References

[1] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, Vol. 2, pp. 53–58, 1989.

[2] N. E. Barabanov and D. V. Prokhorov. Stability analysis of discrete-time recurrent neural networks. *IEEE Transactions on Neural Networks*, Vol. 13, No. 2, pp. 292–303, 2002.

[3] P. L. Bartlett, S. Evans, and P. Long. Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv*:1804.05012 [cs.LG], 2018a.

[4] P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, 2017.

[5] P. L. Bartlett, D. P. Helmbold, and P. Long. Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. *arXiv*:1802.06093 [cs.LG], 2018b.

[6] C. Daniel, J. Taylor, and S. Nowozin. Learning step size controllers for robust neural network training. In *AAAI Conference on Artificial Intelligence*, 2016.

[7] S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6152–6160, 2017.

[8] M. Hardt and T. Ma. Identity matters in deep learning. *arXiv*:1611.04231 [cs.LG], 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[10] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, Vol. 1, pp. 295–307, 1988.

[11] K. Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

[12] H. K. Khalil. *Nonlinear Systems, 3rd Edition.* Prentice Hall, 2002.

[13] G. D. Magoulas, M. N. Vrahatis and G. S. Androulakis. Effective backpropagation training with variable stepsize. *Neural Networks*, Vol. 10, No. 1, pp. 69–82, 1997.

[14] K. Matsuoka. Stability conditions for nonlinear continuous neural networks with asymmetric connection weights. *Neural Networks*, Vol. 5, No. 3, pp. 495–500, 1992.

[15] A. N. Michel, J. A. Farrell, and W. Porod. Stability results for neural networks. In *Neural Information Processing Systems*, pp. 554–563, 1988.

[16] K. Nar and S. Shankar Sastry. Residual networks: Lyapunov stability and convex decomposition. a̅rXiv:1803.08203 [cs.LG], 2018.

[17] B. Neyshabur, R. Tomioka, R. Salakhutdinov, and N. Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv*:1705.03071 [cs.LG], 2017.

[18] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer: New York, NY, 1999.

[19] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv*:1312.6120 [cs.NE], 2013.

[20] M. Rolinek and G. Martius. L4: Practical loss-based stepsize adaptation for deep learning. *arXiv*:1802.05074 [cs.LG], 2018.

Appendix

A Proof of Theorem 1 and Corollary 1

Lemma 1. *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric and positive semidefinite. Then, $\langle A, B\rangle \geq 0$.*

Proof. We can write B as $B = \sum_{i=1}^{n} \lambda_i u_i u_i^\top$, where $\lambda_i \geq 0$ for all $i \in [n]$ and $u_i^\top u_j = 0$ if $i \neq j$. Then,

$$\langle A, B\rangle = \text{trace}\{AB\} = \text{trace}\left\{A \sum\nolimits_{i=1}^{n} \lambda_i u_i u_i^\top\right\} = \sum\nolimits_{i=1}^{n} \lambda_i u_i^\top A u_i \geq 0. \qquad \blacksquare$$

Lemma 2. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ be a linear map defined as $f(X) = \sum_{i=1}^{L} A_i X B_i$, where $A_i \in \mathbb{R}^{m\times m}$ and $B_i \in \mathbb{R}^{n\times n}$ are symmetric positive semidefinite matrices for all $i \in [L]$. Then, for every nonzero $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, the largest eigenvalue of f satisfies*

$$\lambda_{\max}(f) \geq \frac{1}{\|u\|_2^2 \|v\|_2^2} \sum\nolimits_{i=1}^{L} (u^\top A_i u)(v^\top B_i v).$$

Proof. First, we show that f is symmetric and positive semidefinite. Given two matrices $X, Y \in \mathbb{R}^{m\times n}$, we can write

$$\langle X, f(Y)\rangle = \text{trace}\left\{\sum\nolimits_i X^\top A_i Y B_i\right\} = \text{trace}\left\{\sum\nolimits_i B_i Y^\top A_i X\right\} = \langle Y, f(X)\rangle,$$

$$\langle X, f(X)\rangle = \text{trace}\left\{\sum\nolimits_i X^\top A_i X B_i\right\} = \sum\nolimits_i \langle X^\top A_i X, B_i\rangle \geq 0,$$

where the last inequality follows from Lemma 1. This shows that f is symmetric and positive semidefinite. Then, for every nonzero $X \in \mathbb{R}^{m\times n}$, we have

$$\lambda_{\max}(f) \geq \frac{1}{\langle X, X\rangle} \langle X, f(X)\rangle.$$

In particular, given two nonzero vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$,

$$\lambda_{\max}(f) \geq \frac{1}{\langle uv^\top, uv^\top\rangle} \langle uv^\top, f(uv^\top)\rangle = \frac{1}{\|u\|_2^2 \|v\|_2^2} \sum\nolimits_{i=1}^{L} (u^\top A_i u)(v^\top B_i v). \qquad \blacksquare$$

Proof of Theorem 1. The cost function (2) in Theorem 1 can be written as

$$\frac{1}{2}\text{trace}\left\{(W_L \cdots W_1 - R)^\top (W_L \cdots W_1 - R)\right\}.$$

Let E denote the error in the estimate, i.e. $E = W_L \cdots W_1 - R$. The gradient descent yields

$$W_i[k+1] = W_i[k] - \delta W_{i+1}^\top[k] \cdots W_L^\top[k] E[k] W_1^\top[k] \cdots W_{i-1}^\top[k] \quad \forall i \in [L]. \tag{5}$$

By multiplying the update equations of $W_i[k]$ and subtracting R, we can obtain the dynamics of E as

$$E[k+1] = E[k] - \delta \sum\nolimits_{i=1}^{L} A_i[k] E[k] B_i[k] + o(E[k]), \tag{6}$$

where $o(\cdot)$ denotes the higher order terms, and

$$A_i = W_L W_{L-1} \cdots W_{i+1} W_{i+1}^\top \cdots W_{L-1}^\top W_L^\top \quad \forall i \in [L],$$

$$B_i = W_1^\top W_2^\top \cdots W_{i-1}^\top W_{i-1} \cdots W_2 W_1 \quad \forall i \in [L].$$

Lyapunov's indirect method of stability (Khalil, 2002; Sastry, 1999) states that given a dynamical system $x[k+1] = F(x[k])$, its equilibrium x^* is stable in the sense of Lyapunov only if the linearization of the system around x^*

$$(x[k+1] - x^*) = (x[k] - x^*) + \left.\frac{\partial F}{\partial x}\right|_{x=x^*} (x[k] - x^*)$$

does not have any eigenvalue larger than 1 in magnitude. By using this fact for the system defined by (5)-(6), we can observe that an equilibrium $\{\hat{W}_j\}_{j\in[L]}$ with $\hat{W}_L \cdots \hat{W}_1 = \hat{R}$ is stable in the sense of Lyapunov only if the system

$$\left(E[k+1] - \hat{R} + R\right) = \left(E[k] - \hat{R} + R\right) - \delta \sum\nolimits_{i=1}^{L} A_i\Big|_{\{\hat{W}_j\}} \left(E[k] - \hat{R} + R\right) B_i\Big|_{\{\hat{W}_j\}}$$

does not have any eigenvalue larger than 1 in magnitude, which requires that the mapping

$$f(\tilde{E}) = \sum\nolimits_{i=1}^{L} A_i\Big|_{\{\hat{W}_j\}} \tilde{E} B_i\Big|_{\{\hat{W}_j\}} \tag{7}$$

does not have any real eigenvalue larger than $(2/\delta)$. Let u and v be the left and right singular vectors of $\hat{R}$ corresponding to its largest singular value, and let p_j and q_j be defined as in the statement of Theorem 1. Then, by Lemma 2, the mapping f in (7) does not have an eigenvalue larger than $(2/\delta)$ only if

$$\sum\nolimits_{i=1}^{L} p_{i-1}^2 q_{i+1}^2 \leq \frac{2}{\delta},$$

which completes the proof. ∎

Proof of Corollary 1. Note that

$$q_{i+1} p_i = \|u^\top W_L W_{L-1} \cdots W_{i+1}\|_2 \|W_i \cdots W_2 W_1 v\|_2 \geq \|u^\top W_L \cdots W_1 v\|_2 = \rho(R).$$

As long as $\rho(R) \neq 0$, we have $p_i \neq 0$ for all $i \in [L]$, and therefore,

$$p_{i-1}^2 q_{i+1}^2 \geq \frac{p_{i-1}^2}{p_i^2} \rho(R)^2. \tag{8}$$

Using inequality (8), the bound in Theorem 1 can be relaxed as

$$\delta \leq 2\left(\sum\nolimits_{i=1}^{L} \frac{p_{i-1}^2}{p_i^2}\rho(R)^2\right)^{-1}. \tag{9}$$

Since $\prod_{i=1}^{L}(p_i/p_{i-1}) = \rho(R) \neq 0$, we also have the inequality

$$\sum\nolimits_{i=1}^{L} \frac{p_{i-1}^2}{p_i^2}\rho(R)^2 \geq \sum\nolimits_{i=1}^{L} \frac{\rho(R)^2}{\left(\rho(R)^{1/L}\right)^2} = L\rho(R)^{2(L-1)/L},$$

and the bound in (9) can be simplified as

$$\delta \leq \frac{2}{L\rho(R)^{2(L-1)/L}}.$$ ∎

B Proof of Theorem 2

Lemma 3. *Let $\lambda > 0$ be estimated as a multiplication of the scalar parameters $\{w_i\}_{i\in[L]}$ by minimizing $\frac{1}{2}(w_L \cdots w_2 w_1 - \lambda)^2$ via gradient descent. Assume that $w_i[0] = 1$ for all $i \in [L]$. If the step size δ is chosen to be less than or equal to*

$$\delta_c = \begin{cases} L^{-1}\lambda^{-2(L-1)/L} & \textit{if } \lambda \in [1,\infty), \\ (1-\lambda)^{-1}(1-\lambda^{1/L}) & \textit{if } \lambda \in (0,1), \end{cases}$$

then $|w_i[k] - \lambda^{\frac{1}{L}}| \leq \beta(\delta)^k |1 - \lambda^{\frac{1}{L}}|$ for all $i \in [L]$, where

$$\beta(\delta) = \begin{cases} 1 - \delta(\lambda - 1)(\lambda^{1/L} - 1)^{-1} & \textit{if } \lambda \in (1,\infty), \\ 1 - \delta L \lambda^{2(L-1)/L} & \textit{if } \lambda \in (0,1]. \end{cases}$$

Proof. Due to symmetry, $w_i[k] = w_j[k]$ for all $k \in \mathbb{N}$ for all $i,j \in [L]$. Denoting any of them by $w[k]$, we have

$$w[k+1] = w[k] - \delta w^{L-1}[k](w^L[k] - \lambda).$$

To show that $w[k]$ converges to $\lambda^{1/L}$, we can write

$$w[k+1] - \lambda^{1/L} = \mu(w[k])(w[k] - \lambda^{1/L}),$$

where

$$\mu(w) = 1 - \delta w^{L-1} \sum_{j=0}^{L-1} w^j \lambda^{(L-1-j)/L}.$$

If there exists some $\beta \in [0, 1)$ such that

$$0 \leq \mu(w[k]) \leq \beta \text{ for all } k \in \mathbb{N}, \tag{10}$$

then $w[k]$ is always larger or always smaller than $\lambda^{1/L}$, and its distance to $\lambda^{1/L}$ decreases by a factor of β at each step. Since $\mu(w)$ is a monotonic function in w, the condition (10) holds for all k if it holds only for $w[0] = 1$ and $\lambda^{1/L}$, which gives us δ_c and $\beta(\delta)$. ■

Proof of Theorem 2. There exists a common invertible matrix $U \in \mathbb{R}^{n\times n}$ that can diagonalize all the matrices in the system created by the gradient descent: $R = U\Lambda_R U^\top$, $W_i = U\Lambda_{W_i} U^\top$ for all $i \in [L]$. Then the dynamical system turns into n independent update rules for the diagonal elements of Λ_R and $\{\Lambda_{W_i}\}_{i\in[L]}$. Lemma 3 can be applied to each of the n systems involving the diagonal elements. Since δ_c in Lemma 3 is monotonically decreasing in λ, the bound for the maximum eigenvalue of R guarantees linear convergence. ■

C Proof of Theorem 3

Lemma 4. *Assume that $\lambda < 0$ and $w_i[0] = 1$ is used for all $i \in [L]$ to initialize the gradient descent algorithm to solve*

$$\min_{(w_1,\ldots,w_L)\in\mathbb{R}^L} \frac{1}{2}\left(w_L \ldots w_2 w_1 - \lambda\right)^2.$$

Then, each w_i converges to 0 unless $\delta > (1-\lambda)^{-1}$.

Proof. We can write the update rule for any weight w_i as

$$w[k+1] = w[k]\left(1 - \delta\sigma w^{L-2}[k]\left(w^L[k] - \lambda\right)\right)$$

which has one equilibrium at $w^* = \lambda^{1/L}$ and another at $w^* = 0$. If $0 < \delta \leq 1/\sigma(1-\lambda)$ and $w[0] = 1$, it can be shown by induction that

$$0 \leq 1 - \delta\sigma w^{L-2}[k]\left(w^L[k] - \lambda\right) < 1$$

for all $k \geq 0$. As a result, $w[k]$ converges to 0. ■

Proof of Theorem 3. Similar to the proof of Theorem 2, the system created by the gradient descent can be decomposed into n independent systems of the diagonal elements of the matrices Λ_R and $\{\Lambda_{W_i}\}_{i\in[L]}$. Then, Lemma 3 and Lemma 4 can be applied to the systems with positive and negative eigenvalues of R, respectively. ■

D Proof of Theorem 4

To find a necessary condition for the convergence of the gradient descent algorithm to $(\hat{W}, \hat{V})$, we analyze the local stability of that solution in the sense of Lyapunov. Since the analysis is local and the function g is fixed, for each point x_i we can use a matrix G_i that satisfies $G_i(\hat{V}x_i - b) = g(\hat{V}x_i - b)$. Note that G_i is a diagonal matrix and all of its diagonal elements are either 0 or 1. Then, we can write the cost function around an equilibrium as

$$\frac{1}{2}\sum_{i=1}^N \text{trace}\left\{[WG_i(Vx_i - b) - f(x_i)]^\top [WG_i(Vx_i - b) - f(x_i)]\right\}.$$

Denoting the error $WG_i(Vx_i - b) - f(x_i)$ by e_i, the gradient descent gives

$$W[k+1] = W[k] - \delta\sum_{i=1}^N e_i[k](V[k]x_i - b)^\top G_i^\top,$$

$$V[k+1] = V[k] - \delta \sum\nolimits_{i=1}^{N} G_i^\top W[k]^\top e_i[k] x_i^\top .$$

Let e denote the vector $(e_1^\top \ \dots \ e_N^\top)^\top$. Then we can write the update equation of e_j as

$$\begin{aligned} e_j[k+1] &= e_j[k] - \delta W[k] G_j \sum\nolimits_i G_i^\top W[k]^\top e_i[k] x_i^\top x_j \\ &\quad -\delta \sum\nolimits_i e_i[k](V[k]x_i - b)^\top G_i^\top G_j (V[k]x_j - b) + o(e[k]). \end{aligned}$$

Similar to the proof of Theorem 1, the equilibrium $(\hat{W}, \hat{V})$ can be stable in the sense on Lyapunov only if the system

$$e_j[k+1] = e_j[k] - \delta \sum\nolimits_i \hat{W} G_j G_i^\top \hat{W}^\top e_i[k] x_i^\top x_j - \delta \sum\nolimits_i e_i[k] (\hat{V} x_i - b)^\top G_i^\top G_j (\hat{V} x_j - b) \quad (11)$$

does not have any eigenvalue larger than 1 in magnitude. Note that the linear system in (11) can be described by a symmetric matrix, whose eigenvalues cannot be larger in magnitude than the eigenvalues of its sub-blocks on the diagonal, in particular those of the system

$$e_j[k+1] = e_j[k] - \delta \hat{W} G_j G_j^\top \hat{W}^\top e_j[k] x_j^\top x_j - \delta e_j[k] (\hat{V} x_j - b)^\top G_j^\top G_j (\hat{V} x_j - b). \quad (12)$$

The eigenvalues of the system (12) are less than 1 in magnitude only if the eigenvalues of the system

$$h(u) = \hat{W} G_j G_j^\top \hat{W}^\top u x_j^\top x_j + u (\hat{V} x_j - b)^\top G_j^\top G_j (\hat{V} x_j - b)$$

are less than $(2/\delta)$. This requires that for all $j \in [N]$ for which $\hat{f}(x_j) \neq 0$,

$$\begin{aligned} \frac{2}{\delta} &\geq \frac{\langle \hat{f}(x_j), h(\hat{f}(x_j)) \rangle}{\langle \hat{f}(x_j), \hat{f}(x_j) \rangle} \\ &= \frac{1}{\|\hat{f}(x_j)\|^2} \left(\|G_j^\top \hat{W}^\top \hat{f}(x_j)\|^2 \|x_j\|^2 + \|\hat{f}(x_j)\|^2 \|G_j (\hat{V} x_j - b)\|^2 \right) \\ &\geq \frac{1}{\|\hat{f}(x_j)\|^2} \frac{\|(\hat{V} x_j - b)^\top G_j^\top G_j^\top \hat{W}^\top \hat{f}(x_j)\|^2}{\|(\hat{V} x_j - b)^\top G_j^\top\|^2} \|x_j\|^2 + \|G_j (\hat{V} x_j - b)\|^2 \\ &= \frac{1}{\|G_j (\hat{V} x_j - b)\|^2} \|\hat{f}(x_j)\|^2 \|x_j\|^2 + \|G_j (\hat{V} x_j - b)\|^2 \\ &\geq 2 \|\hat{f}(x_j)\| \|x_j\|. \end{aligned}$$

As a result, Lyapunov stability of the solution $(\hat{W}, \hat{V})$ requires

$$\frac{1}{\delta} \geq \max_i \|\hat{f}(x_i)\| \|x_i\|. \qquad \blacksquare$$