On the Implicit Bias of Adam

Matias D. Cattaneo[*]
Princeton University
cattaneo@princeton.edu

Jason M. Klusowski[*]
Princeton University
jason.klusowski@princeton.edu

Boris Shigida[*]
Princeton University
bs1624@princeton.edu

Abstract

In previous literature, backward error analysis was used to find ordinary differential equations (ODEs) approximating the gradient descent trajectory. It was found that finite step sizes implicitly regularize solutions because terms appearing in the ODEs penalize the two-norm of the loss gradients. We prove that the existence of similar implicit regularization in RMSProp and Adam depends on their hyperparameters and the training stage, but with a different "norm" involved: the corresponding ODE terms either penalize the (perturbed) one-norm of the loss gradients or, on the contrary, hinder its decrease (the latter case being typical). We also conduct numerical experiments and discuss how the proven facts can influence generalization.

August 31, 2023

1 Introduction

Gradient descent can be seen as a numerical method solving the ordinary differential equation (ODE) $\dot{\boldsymbol{\theta}} = -\nabla E(\boldsymbol{\theta})$, where $E(\cdot)$ is the loss function and $\nabla E(\boldsymbol{\theta})$ denotes its gradient. Starting at $\boldsymbol{\theta}^{(0)}$, it creates a sequence of guesses $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$, which lie close to the solution trajectory $\boldsymbol{\theta}(t)$ governed by aforementioned ODE. Since the step size h is finite, one could search for a modified differential equation $\dot{\tilde{\boldsymbol{\theta}}} = -\nabla \widetilde{E}(\tilde{\boldsymbol{\theta}})$ such that $\boldsymbol{\theta}^{(n)} - \tilde{\boldsymbol{\theta}}(nh)$ is exactly zero, or at least closer to zero than $\boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}(nh)$, that is, all the guesses of the descent lie exactly on the new solution curve or closer compared to the original curve. This approach to analysing properties of numerical method is called backward error analysis in the numerical integration literature (see Chapter IX in [5]).

[1] first used this idea for full-batch gradient descent and found that the modified loss function $\widetilde{E}(\tilde{\boldsymbol{\theta}}) = E(\tilde{\boldsymbol{\theta}}) + (h/4)\|\nabla E(\tilde{\boldsymbol{\theta}})\|^2$ makes the trajectory of the solution to $\dot{\tilde{\boldsymbol{\theta}}} = -\nabla \widetilde{E}(\tilde{\boldsymbol{\theta}})$ approximate the sequence $\{\boldsymbol{\theta}^{(n)}\}_{n=0}^{\infty}$ one order of h better than the original differential equation, where $\|\cdot\|$ denotes the Euclidean norm. In related work, [22] obtained the correction term for full-batch gradient descent up to any chosen order, also studying the global error (uniform in the iteration number) as opposed to the local (one-step) error.

The analysis was later extended to mini-batch gradient descent in [28]. Assume that the training set is split in batches of size B and there are m batches per epoch (so the training set size is mB), the cost function is rewritten $E(\boldsymbol{\theta}) = (1/m)\sum_{k=0}^{m-1} \hat{E}_k(\boldsymbol{\theta})$ with mini-batch costs denoted

[*]Equal contribution

Preprint. Under review.

$\hat{E}_k(\boldsymbol{\theta}) = (1/B)\sum_{j=kB+1}^{kB+B} E_j(\boldsymbol{\theta})$. It was obtained in that work that after one epoch, the mean iterate of the algorithm, averaged over all possible shuffles of the batch indices, is close to the solution to $\dot{\boldsymbol{\theta}} = -\nabla \widetilde{E}_{SGD}(\boldsymbol{\theta})$, where the modified loss is given by

$$\widetilde{E}_{SGD}(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \frac{h}{4m}\sum_{k=0}^{m-1}\left\|\nabla \hat{E}(\boldsymbol{\theta})\right\|^2.$$

More recently, [8] studied the gradient descent with heavy-ball momentum iteration $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - h\nabla E(\boldsymbol{\theta}^{(n)}) + \beta(\boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}^{(n-1)})$, where β is the momentum parameter. They proved that it is close to the continuous trajectory of the piecewise first-order ODE

$$\dot{\tilde{\boldsymbol{\theta}}} = -\frac{1-\beta^{n+1}}{1-\beta}\nabla E(\tilde{\boldsymbol{\theta}}(t)) - \frac{h\gamma_n(1+\beta)}{2(1-\beta)^3}\nabla^2 E(\tilde{\boldsymbol{\theta}}(t))\nabla E(\tilde{\boldsymbol{\theta}}(t)), \quad t \in [nh, (n+1)h],$$

where

$$\gamma_n = \left(1-\beta^{2n+2}\right) - 4(n+1)\beta^{n+1}\frac{(1-\beta)}{(1+\beta)}.$$

This result is stated here in a full-batch setting and is a special case of their main theorem, which includes mini-batches.

In another recent work, [34] introduce a regularization term $\lambda \cdot \|\nabla E(\boldsymbol{\theta})\|$ to the loss function as a way to ensure finding flatter minima, which have been observed empirically to have a smaller test error. The only difference between their term and the first-order correction coming from backward error analysis (up to a coefficient) is that the norm is not squared.

Using backward error analysis to approximate the discrete dynamics with a modified ODE for adaptive algorithms such as RMSProp [30] and Adam [17] (which is an improvement over RMSProp and AdaGrad[4]) is currently missing in the literature. [1] note that "it would be interesting to use backward error analysis to calculate the modified loss and implicit regularization for other widely used optimizers such as momentum, Adam and RMSprop". [28] reiterate that they "anticipate that backward error analysis could also be used to clarify the role of finite learning rates in adaptive optimizers like Adam". In the same context, [8] agree that "RMSProp ... and Adam ..., albeit being powerful alternatives to SGD with faster convergence rates, are far from well-understood in the aspect of implicit regularization". In a similar context, in Appendix G to [22] it is mentioned that "its [Adam's] counter term and discretization error are open questions".

This paper fills the gap in the literature by conducting backward error analysis for (mini-batch, and full-batch as a special case) Adam and RMSProp. Our main contributions are listed below.

- In Theorem 3.1 and Theorem 4.2, we provide a global second-order in h continuous ODE approximation to Adam and RMSProp in the general mini-batch setting. For the full-batch special case, it was shown in prior work [21] that the continuous-time limit of both these algorithms is a (perturbed by ε) signGD flow
$$\dot{\boldsymbol{\theta}} = -\frac{\nabla E(\boldsymbol{\theta})}{|\nabla E(\boldsymbol{\theta})| + \varepsilon}$$
component-wise, where ε is the numerical stability parameter; we make this more precise by finding an additional "bias" term on the right (linearly depending on h).
- We analyze the full-batch case in more detail. We find that the bias term does something different from penalizing the two-norm of the loss gradient as in the case of gradient descent: it either penalizes the perturbed one-norm of the loss gradient, defined as $\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^{p}\sqrt{v_i^2+\varepsilon}$, or, on the contrary, hinders its decrease (depending on hyperparameters and the training stage). See the summary of our theoretical finding for the full-batch case in Section 2. We also obtain the backward error analysis result for heavy-ball momentum gradient descent (which was derived before in [8]) as a special case: see Example 2.3.
- We provide numerical evidence consistent with our results. In particular, we notice that often penalizing the perturbed one-norm appears to improve generalization, and hindering its decrease hurts it. The typical absence of implicit regularization appearing from backward error analysis in RMSProp and Adam (as opposed to GD) becomes one more previously unidentified possible explanation for poorer generalization of adaptive gradient algorithms compared to other methods.

- Consistent with our theoretical results, we notice a phenomenon of rising and falling norm in our experiments, which is described as follows. The training of full-batch Adam first steeply decreases the training loss and the perturbed one-norm of the loss gradient, while increasing the test accuracy. Then, however, the perturbed norm rises, even though the training loss and test accuracy behave as expected: continue to decrease and increase respectively. Later the perturbed norm falls again. It seems that the height of the "hill" the perturbed norm graph rises to mid-training depends on how much the bias term hinders the decrease of this norm. To the best of our knowledge, this has not been noticed previously, though the phenomenon of the two-norm of the loss gradient rising while training convolutional neural networks with SGD may be related.[2]

Related work

Backward error analysis of first-order methods. We provide the history of finding ordinary differential equations approximating different algorithms in the introduction. Recently, there have been other applications of backward error analysis related to machine learning. [18] show that the approximating continuous-time trajectories satisfy conservation laws that are broken in discrete time. [7] use backward error analysis while studying how to discretize continuous-time dynamical systems preserving stability and convergence rates. [27] find continuous-time approximations of discrete two-player differential games.

Approximating gradient methods by differential equation trajectories. [21] prove that the trajectories of Adam and RMSProp are close to signGD dynamics, and investigate different training regimes of these algorithms empirically. SGD is approximated by stochastic differential equations and novel adaptive parameter adjustment policies are devised in [19].

Implicit bias of first-order methods. [29] prove that gradient descent trained to classify linearly separable data in the case of logistic loss converges to the direction of the max-margin vector (the solution to the hard margin SVM). This result has been extended to different loss functions in [24], to stochastic gradient descent in [25] and more generic optimization methods in [9], to the nonseparable case in [13], [14]. This line of research has been generalized to studying implicit biases of linear networks [12], [10], homogeneous neural networks [11], [23], [20]. [32] study the gradient flow of a diagonal linear network with squared loss and show that large initializations lead to minimum 2-norm solutions while small initializations lead to minimum 1-norm solutions. [6] extend this work to the case of non-zero step sizes and mini-batch training (SGD). [31] prove that Adam and RMSProp maximize the margin of homogeneous neural networks.

Generalization of adaptive methods. [3] empirically investigate the edge-of-stability regime of adaptive gradient algorithms and the effect of sharpness (defined as the largest eigenvalue of the hessian) on generalization. [15] introduce a statistic that measures the uniformity of the hessian diagonal and argue that adaptive gradient algorithms are biased towards making this statistic smaller. [16] propose to improve generalization of adaptive methods by switching to SGD in the middle of training.

Convergence of Adam. [26] investigate cases where Adam fails to converge to the optimal solution and argue that this is because of exponential averaging of the gradients only provides a short-term memory, and [33] propose ways of fixing the non-convergence issues by preventing the uncontrolled increase of the effective learning rate (i. e. learning rate divided by the square root of the exponential moving average of the squared gradients).

Notation

We denote the loss of the kth minibatch as a function of the network parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ by $E_k(\boldsymbol{\theta})$, and in the full-batch setting we omit the index and write $E(\boldsymbol{\theta})$. ∇E means the gradient of E, and ∇ with indices means partial derivatives, e. g. $\nabla_{ijs} E$ is a shortcut for $\frac{\partial^3 E}{\partial\theta_i \partial\theta_j \partial\theta_s}$. The norm without indices $\|\cdot\|$ is the two-norm of a vector, $\|\cdot\|_1$ is the one-norm and $\|\cdot\|_{1,\varepsilon}$ is the perturbed one-norm

[2] See, e. g., the presentation http://videolectures.net/deeplearning2015_goodfellow_network_optimization/.

defined as $\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^{p} \sqrt{v_i^2 + \varepsilon}$. (Of course, if $\varepsilon > 0$ the perturbed one-norm is not a norm, but $\varepsilon = 0$ makes it the one-norm.)

2 Implicit bias of full-batch Adam: an informal summary

To avoid ambiguity and to provide the names and notations for hyperparameters, we define the algorithm below.

Definition 2.1. The *Adam* algorithm is an optimization algorithm with numerical stability hyperparameter $\varepsilon > 0$, squared gradient momentum hyperparameter $\rho \in (0, 1)$, gradient momentum parameter $\beta \in (0, 1)$, initialization $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^p$, $\boldsymbol{\nu}^{(0)} = \mathbf{0} \in \mathbb{R}^p$, $\mathbf{m}^{(0)} = \mathbf{0} \in \mathbb{R}^p$ and the following update rule: for each $n \geq 0$, $j \in \{1, \dots, p\}$

$$\begin{aligned} \nu_j^{(n+1)} &= \rho \nu_j^{(n)} + (1-\rho)\big(\nabla_j E_n(\boldsymbol{\theta}^{(n)})\big)^2, \\ m_j^{(n+1)} &= \beta m_j^{(n)} + (1-\beta)\nabla_j E_n(\boldsymbol{\theta}^{(n)}), \\ \theta_j^{(n+1)} &= \theta_j^{(n)} - h \frac{m_j^{(n+1)}/(1-\beta^{n+1})}{\sqrt{\nu_j^{(n+1)}/(1-\rho^{n+1}) + \varepsilon}}. \end{aligned} \tag{1}$$

Remark 2.2 (The ε hyperparameter is inside the square root)**.** Note that the numerical stability hyperparameter $\varepsilon > 0$, which is introduced in these algorithms to avoid division by zero, is inside the square root in our definition. This way we avoid division by zero in the derivative too: the first derivative of $x \mapsto \big(\sqrt{x+\varepsilon}\big)^{-1}$ is bounded for $x \geq 0$. This is useful for our analysis. In the appendix, the original versions of RMSProp and Adam are also tackled, though with an additional assumption which requires that no component of the gradient can come very close to zero in the region of interest (in particular, no component can change the sign). This is true only for the initial period of learning (whereas the theorems below are true for the whole period).

Practitioners do not seem to make a distinction between the version with ε inside vs. outside the square root: tutorials with both versions abound on machine learning related websites. Moreover, the popular Tensorflow variant of RMSProp has ε inside the square root[3] even though in the documentation[4] [17] is cited, where ε is outside. While conducting numerical experiments, we also noted that moving ε inside or outside the square root does not change the behavior of the algorithms qualitatively.

Summary of our main result (in the full-batch case)

Having provided the definition, we are ready to informally describe our theoretical result (in the full-batch special case). Assume $E(\boldsymbol{\theta})$ is the loss, whose partial derivatives up to the fourth order are bounded. Let $\{\boldsymbol{\theta}^{(n)}\}$ be iterations of Adam as defined in Definition 2.1. Our main result for this case is finding an ODE whose solution trajectory $\tilde{\boldsymbol{\theta}}(t)$ is h^2-close to $\{\boldsymbol{\theta}^{(n)}\}$, meaning that for any positive time horizon $T > 0$ there exists a constant $C > 0$ such that for any step size $h \in (0, T)$ we have $\|\tilde{\boldsymbol{\theta}}(nh) - \boldsymbol{\theta}^{(n)}\| \leq Ch^2$ (for n between 0 and $\lfloor T/h \rfloor$). The ODE is written the following way (up to terms that rapidly go to zero as n grows): for the component number $j \in \{1, \dots, p\}$

$$\dot{\tilde{\theta}}_j(t) = -\frac{1}{\sqrt{|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))|^2 + \varepsilon}} \big(\nabla_j E(\tilde{\boldsymbol{\theta}}(t)) + \text{bias}\big) \tag{2}$$

with initial conditions $\tilde{\boldsymbol{\theta}}_j(0) = \boldsymbol{\theta}_j^{(0)}$ for all j, where the bias term is

$$\text{bias} := \frac{h}{2} \left\{ \frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho} + \frac{1+\rho}{1-\rho} \cdot \frac{\varepsilon}{|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))|^2 + \varepsilon} \right\} \nabla_j \big\| \nabla E(\tilde{\boldsymbol{\theta}}(t)) \big\|_{1,\varepsilon}. \tag{3}$$

Depending on hyperparameter values and the training stage, the bias term can take two extreme forms, and during most of the training the reality is usually in between. The extreme cases are as follows.

[3] https://github.com/keras-team/keras/blob/f9336cc5114b4a9429a242deb264b707379646b7/keras/optimizers/rmsprop.py#L190

[4] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/experimental/RMSprop

	ε "small"	ε "large"
$\beta \geq \rho$	$\|\nabla E(\boldsymbol{\theta})\|_1$-penalized	$\|\nabla E(\boldsymbol{\theta})\|_2^2$-penalized
$\rho > \beta$	$-\|\nabla E(\boldsymbol{\theta})\|_1$-penalized	$\|\nabla E(\boldsymbol{\theta})\|_2^2$-penalized

Table 1: Implicit bias of Adam: special cases. "Small" and "large" are in relation to squared gradient components.

- If $\sqrt{\varepsilon}$ is **small** compared to all components of $\nabla E(\tilde{\boldsymbol{\theta}}(t))$, i. e. $\min_j \left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \gg \sqrt{\varepsilon}$, which is the case during the initial learning stage, then

$$\text{bias} = \frac{h}{2}\left\{\frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho}\right\}\nabla_j \left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon}. \tag{4}$$

  For small ε, the perturbed one-norm is indistinguishable from the usual one-norm, and for $\beta > \rho$ it is penalized (in much the same way as the squared two-norm is implicitly penalized in the case of GD), but for $\rho > \beta$ its decrease is actually hindered by this term (so the bias is opposite to penalization). The ODE in (2) can be approximately rewritten as

$$\dot{\tilde{\theta}}_j(t) = -\frac{\nabla_j \widetilde{E}(\tilde{\boldsymbol{\theta}}(t))}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|}, \qquad \widetilde{E}(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \frac{h}{2}\left\{\frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho}\right\}\left\|\nabla E(\boldsymbol{\theta})\right\|_1. \tag{5}$$

- If $\sqrt{\varepsilon}$ is **large** compared to all gradient components, i. e. $\max_j \left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \ll \sqrt{\varepsilon}$, which may happen during the later learning stage, the fraction with ε is the numerator in (3) approaches one, the dependence on ρ cancels out, and

$$\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon} \approx \sum_{i=1}^{p}\sqrt{\varepsilon}\left(1 + \frac{\left|\nabla_i E(\tilde{\boldsymbol{\theta}}(t))\right|^2}{2\varepsilon}\right) = p\sqrt{\varepsilon} + \frac{1}{2\sqrt{\varepsilon}}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|^2. \tag{6}$$

  In other words, $\|\cdot\|_{1,\varepsilon}$ becomes $\|\cdot\|^2/(2\sqrt{\varepsilon})$ up to an additive constant (which is "eaten" by the gradient):

$$\text{bias} = \frac{h}{4\sqrt{\varepsilon}}\frac{1+\beta}{1-\beta}\nabla_j \left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|^2.$$

  The form of the ODE in this case is

$$\dot{\tilde{\theta}}_j(t) = -\nabla_j \widetilde{E}(\tilde{\boldsymbol{\theta}}(t)), \qquad \widetilde{E}(\boldsymbol{\theta}) = \frac{1}{\sqrt{\varepsilon}}\left(E(\tilde{\boldsymbol{\theta}}(t)) + \frac{h}{4\sqrt{\varepsilon}}\frac{1+\beta}{1-\beta}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|^2\right). \tag{7}$$

These two extreme cases are summarized in Table 1. In Figure 1, we use the one-dimensional ($p = 1$) case to illustrate what kind of term is being implicitly penalized.

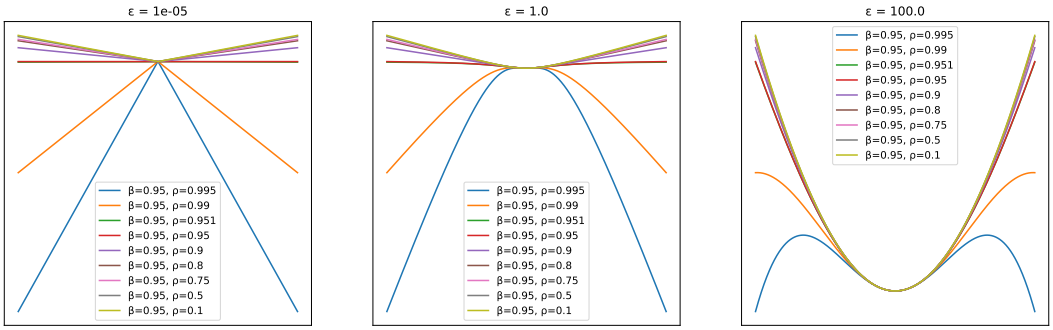

Figure 1: The graphs of $x \mapsto \int_0^x \left\{\frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho} + \frac{1+\rho}{1-\rho}\cdot\frac{\varepsilon}{y^2+\varepsilon}\right\}\mathrm{d}\sqrt{\varepsilon+y^2}$ for different β and ρ.

Example 2.3 (Backward Error Analysis for GD with Heavy-ball Momentum)**.** Assume ε is very large compared to all squared gradient components during the whole training process, so that the form of the ODE is approximated by (7). Since Adam with a large ε and after a certain number of iterations approximates SGD with heavy-ball momentum with step size $h\frac{1-\beta}{\sqrt{\varepsilon}}$, linear step size change (and corresponding time change) gives exactly the equations in Theorem 4.1 of [8]. Taking $\beta = 0$ (no momentum), we get the implicit regularization of GD from [1].

This overview also applies to RMSProp by setting $\beta = 0$. See Section 4 for the formal result.

3 ODE approximating mini-batch Adam trajectories: full statement

We only make one assumption, which is standard in the literature: the loss for each mini-batch is 4 times continuously differentiable partial derivatives up to order 4 of each mini-batch loss E_k are bounded by constants, i. e. there exists a positive constant M such that for $\boldsymbol{\theta}$ in the region of interest

$$\sup_k \sup_{\boldsymbol{\theta}} \left\{ \sup_i |\nabla_i E_k(\boldsymbol{\theta})| \vee \sup_{i,j} |\nabla_{ij} E_k(\boldsymbol{\theta})| \vee \sup_{i,j,s} |\nabla_{ijs} E_k(\boldsymbol{\theta})| \vee \sup_{i,j,s,r} |\nabla_{ijsr} E_k(\boldsymbol{\theta})| \right\} \leq M. \quad (8)$$

We now state the main result for mini-batch Adam, whose proof is in the supplemental appendix.

Theorem 3.1. *Assume (8) holds. Let $\{\boldsymbol{\theta}^{(n)}\}$ be iterations of Adam as defined in Definition 2.1, $\tilde{\boldsymbol{\theta}}(t)$ be the continuous solution to the piecewise ODE*

$$\begin{aligned} \dot{\tilde{\theta}}_j(t) = &- \frac{M_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))}{R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))} \\ &+ h \left(\frac{M_j^{(n)}(\tilde{\boldsymbol{\theta}}(t)) \big(2P_j^{(n)}(\tilde{\boldsymbol{\theta}}(t)) + \bar{P}_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))\big)}{2R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))^3} - \frac{2L_j^{(n)}(\tilde{\boldsymbol{\theta}}(t)) + \bar{L}_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))}{2R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))} \right). \end{aligned} \quad (9)$$

for $t \in [nh, (n+1)h]$ with the initial condition $\tilde{\boldsymbol{\theta}}(0) = \boldsymbol{\theta}^{(0)}$, where

$$\begin{aligned} R_j^{(n)}(\boldsymbol{\theta}) &:= \sqrt{\sum_{k=0}^{n} \rho^{n-k}(1-\rho)(\nabla_j E_k(\boldsymbol{\theta}))^2/(1-\rho^{n+1}) + \varepsilon}, \\ M_j^{(n)}(\boldsymbol{\theta}) &:= \frac{1}{1-\beta^{n+1}} \sum_{k=0}^{n} \beta^{n-k}(1-\beta)\nabla_j E_k(\boldsymbol{\theta}), \\ L_j^{(n)}(\boldsymbol{\theta}) &:= \frac{1}{1-\beta^{n+1}} \sum_{k=0}^{n} \beta^{n-k}(1-\beta) \sum_{i=1}^{p} \nabla_{ij} E_k(\boldsymbol{\theta}) \sum_{l=k}^{n-1} \frac{M_i^{(l)}(\boldsymbol{\theta})}{R_i^{(l)}(\boldsymbol{\theta})}, \\ \bar{L}_j^{(n)}(\boldsymbol{\theta}) &:= \frac{1}{1-\beta^{n+1}} \sum_{k=0}^{n} \beta^{n-k}(1-\beta) \sum_{i=1}^{p} \nabla_{ij} E_k(\boldsymbol{\theta}) \frac{M_i^{(n)}(\boldsymbol{\theta})}{R_i^{(n)}(\boldsymbol{\theta})}, \\ P_j^{(n)}(\boldsymbol{\theta}) &:= \frac{1}{1-\rho^{n+1}} \sum_{k=0}^{n} \rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta}) \sum_{i=1}^{p} \nabla_{ij} E_k(\boldsymbol{\theta}) \sum_{l=k}^{n-1} \frac{M_i^{(l)}(\boldsymbol{\theta})}{R_i^{(l)}(\boldsymbol{\theta})}, \\ \bar{P}_j^{(n)}(\boldsymbol{\theta}) &:= \frac{1}{1-\rho^{n+1}} \sum_{k=0}^{n} \rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta}) \sum_{i=1}^{p} \nabla_{ij} E_k(\boldsymbol{\theta}) \frac{M_i^{(n)}(\boldsymbol{\theta})}{R_i^{(n)}(\boldsymbol{\theta})}. \end{aligned}$$

Then, for any fixed positive time horizon $T > 0$ there exists a constant C such that for any step size $h \in (0, T)$ we have

$$\left\| \tilde{\boldsymbol{\theta}}(nh) - \boldsymbol{\theta}^{(n)} \right\| \leq Ch^2, \quad n = 0, 1, \ldots, \lfloor T/h \rfloor. \quad (10)$$

Remark 3.2 (Backward error analysis of Adam in the full-batch setting)**.** In the full-batch setting $E_k \equiv E$, the terms in Theorem 3.1 simplify to

$$R_j^{(n)}(\boldsymbol{\theta}) = \sqrt{|\nabla_j E(\boldsymbol{\theta})|^2 + \varepsilon}, \qquad M_j^{(n)}(\boldsymbol{\theta}) = \nabla_j E(\boldsymbol{\theta}),$$

$$L_j^{(n)}(\boldsymbol{\theta}) = \left[\frac{\beta}{1-\beta} - \frac{(n+1)\beta^{n+1}}{1-\beta^{n+1}}\right] \nabla_j \|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}, \qquad \bar{L}_j^{(n)}(\boldsymbol{\theta}) = \nabla_j \|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon},$$

$$P_j^{(n)}(\boldsymbol{\theta}) = \left[\frac{\rho}{1-\rho} - \frac{(n+1)\rho^{n+1}}{1-\rho^{n+1}}\right] \nabla_j E(\boldsymbol{\theta}) \nabla_j \|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}, \qquad \bar{P}_j^{(n)}(\boldsymbol{\theta}) = \nabla_j E(\boldsymbol{\theta}) \nabla_j \|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}.$$

If the iteration number n is large, (9) rapidly becomes as described in (2) and (3).

4 ODE approximating mini-batch RMSProp trajectories: full statement

We also study the properties of RMSProp using the same arguments as in Theorem 3.1. Up to rapidly vanishing terms, the resulting ODE is the same as for Adam with $\beta = 0$. See Remark 4.3 below.

Definition 4.1. The *RMSProp* algorithm is an optimization algorithm with numerical stability parameter $\varepsilon > 0$, squared gradient momentum parameter $\rho \in (0,1)$, initialization $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^p$, $\boldsymbol{\nu}^{(0)} = \mathbf{0} \in \mathbb{R}^p$ and the following update rule: for each $n \geq 0$, $j \in \{1, \ldots, p\}$

$$\begin{aligned} \nu_j^{(n+1)} &= \rho \nu_j^{(n)} + (1-\rho)\big(\nabla_j E_n(\boldsymbol{\theta}^{(n)})\big)^2, \\ \theta_j^{(n+1)} &= \theta_j^{(n)} - \frac{h}{\sqrt{\nu_j^{(n+1)} + \varepsilon}} \nabla_j E_n(\boldsymbol{\theta}^{(n)}). \end{aligned} \tag{11}$$

We now state the main result for mini-batch RMSProp, whose proof is in the supplemental appendix.

Theorem 4.2. *Assume (8) holds. Let $\{\boldsymbol{\theta}^{(n)}\}$ be iterations of RMSProp as defined in Definition 4.1, $\tilde{\boldsymbol{\theta}}(t)$ be the continuous solution to the piecewise ODE*

$$\begin{aligned} \dot{\tilde{\theta}}_j(t) = &-\frac{\nabla_j E_n(\tilde{\boldsymbol{\theta}}(t))}{R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))} \\ &+ h\left(\frac{\nabla_j E_n(\tilde{\boldsymbol{\theta}}(t))\big(2P_j^{(n)}(\tilde{\boldsymbol{\theta}}(t)) + \bar{P}_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))\big)}{2R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))^3} - \frac{\sum_{i=1}^p \nabla_{ij} E_n(\tilde{\boldsymbol{\theta}}(t)) \frac{\nabla_i E_n(\tilde{\boldsymbol{\theta}}(t))}{R_i^{(n)}(\tilde{\boldsymbol{\theta}}(t))}}{2R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))}\right). \end{aligned} \tag{12}$$

for $t \in [nh, (n+1)h]$ with the initial condition $\tilde{\boldsymbol{\theta}}(0) = \boldsymbol{\theta}^{(0)}$, where

$$\begin{aligned} R_j^{(n)}(\boldsymbol{\theta}) &:= \sqrt{\sum_{k=0}^n \rho^{n-k}(1-\rho)(\nabla_j E_k(\boldsymbol{\theta}))^2 + \varepsilon}, \\ P_j^{(n)}(\boldsymbol{\theta}) &:= \sum_{k=0}^n \rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta}) \sum_{i=1}^p \nabla_{ij} E_k(\boldsymbol{\theta}) \sum_{l=k}^{n-1} \frac{\nabla_i E_l(\boldsymbol{\theta})}{R_i^{(l)}(\boldsymbol{\theta})}, \\ \bar{P}_j^{(n)}(\boldsymbol{\theta}) &:= \sum_{k=0}^n \rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta}) \sum_{i=1}^p \nabla_{ij} E_k(\boldsymbol{\theta}) \frac{\nabla_i E_n(\boldsymbol{\theta})}{R_i^{(n)}(\boldsymbol{\theta})}. \end{aligned}$$

Then, for any fixed positive time horizon $T > 0$ there exists a constant C such that for any step size $h \in (0, T)$ we have

$$\big\|\tilde{\boldsymbol{\theta}}(nh) - \boldsymbol{\theta}^{(n)}\big\| \leq Ch^2, \quad n = 0, 1, \ldots, \lfloor T/h \rfloor.$$

Remark 4.3 (Backward error analysis of RMSProp in the full-batch setting)**.** In the full-batch setting $E_k \equiv E$, the terms in Theorem 4.2 simplify to

$$\begin{aligned} R_j^{(n)}(\boldsymbol{\theta}) &= \sqrt{|\nabla_j E(\boldsymbol{\theta})|^2 (1-\rho^{n+1}) + \varepsilon}, \\ P_j^{(n)}(\boldsymbol{\theta}) &= \sum_{k=0}^n \rho^{n-k}(1-\rho)\nabla_j E(\boldsymbol{\theta}) \sum_{i=1}^p \nabla_{ij} E(\boldsymbol{\theta}) \sum_{l=k}^{n-1} \frac{\nabla_i E(\boldsymbol{\theta})}{\sqrt{|\nabla_i E(\boldsymbol{\theta})|^2 (1-\rho^{l+1}) + \varepsilon}}, \\ \bar{P}_j^{(n)}(\boldsymbol{\theta}) &= (1-\rho^{n+1})\nabla_j E(\boldsymbol{\theta}) \sum_{i=1}^p \nabla_{ij} E(\boldsymbol{\theta}) \frac{\nabla_i E(\boldsymbol{\theta})}{\sqrt{|\nabla_i E(\boldsymbol{\theta})|^2 (1-\rho^{n+1}) + \varepsilon}}. \end{aligned}$$

If the iteration number n is large, (12) rapidly becomes as described in (2) and (3) with $\beta = 0$.

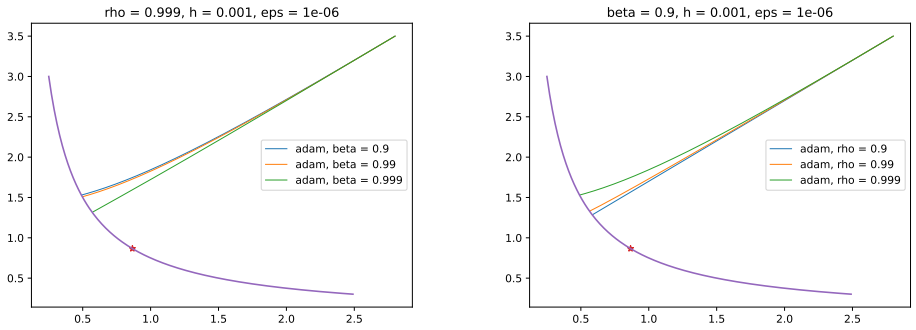

Figure 2: Increasing β moves the trajectory of Adam towards the regions with smaller one-norm of the gradient (if ε is sufficiently small); increasing ρ does the opposite. The violet line is the line of global minima, and the cross denotes the limiting point of minimal one-norm of the gradient. All Adam trajectories start at $(2.8, 3.5)$.

5 Discussion

First conclusion. Recall that from [8] the ODE approximating the dynamics of full-batch heavy-ball momentum GD is close to

$$\dot{\boldsymbol{\theta}} = \frac{1}{1-\beta}\nabla E(\boldsymbol{\theta}) + \underbrace{h\frac{1+\beta}{4(1-\beta)^3}\nabla \left\|\nabla E(\boldsymbol{\theta})\right\|^2}_{\text{regularization}}.$$

The first-order term regularizes the training process by penalizing the two-norm of the gradient of the loss. We can conclude with high confidence that *this* kind of regularization is typically absent in RMSProp (if ε is small) and Adam with $\rho > \beta$ (if ε is small). This may partially explain why these algorithms generalize worse than their non-adaptive counterparts.

Second conclusion. However, the bias term in (3) does contain a kind of "norm" which is the perturbed one-norm $\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^{p}\sqrt{v_i^2+\varepsilon}$. If $\sqrt{\varepsilon}$ is small compared to gradient components, which is usually true except at the end of the training, we can conclude from (5) with moderate confidence that it is only in the case $\beta > \rho$ that the perturbed norm *is* penalized, and decreasing ρ or increasing β moves the trajectory towards regions with lower "norm".

Third conclusion. There is currently no theory that would indicate that penalizing the (perturbed) one-norm of the gradient improves generalization. However, reasoning by analogy (with the case of the two-norm), we can conjecture with lower confidence that at least in some stable regimes of training increasing β and decreasing ρ should improve the test error.

6 Illustration: simple bilinear model

We now analyze the effect of the first-order term for Adam in the same model as [1] and [8] have studied. Namely, assume the parameter $\boldsymbol{\theta} = (\theta_1, \theta_2)$ is 2-dimensional, and the loss is given by $E(\boldsymbol{\theta}) := 1/2(y - \theta_1\theta_2 x)^2$, where x, y are fixed scalars $x = 2$, $y = 3/2$. The loss is minimized on the hyperbola $\theta_1\theta_2 = y/x$. We graph the trajectories of Adam in this case: Figure 2 shows that increasing β forces the trajectory to the region with smaller 1-norm of the gradient of the loss $\|\nabla E(\boldsymbol{\theta})\|_1$, and increasing ρ does the opposite. Figure 3 shows that increasing the learning rate moves Adam towards the region with smaller $\|\nabla E(\boldsymbol{\theta})\|_1$ if $\beta > \rho$ (just like in the case of gradient descent, except the norm is different if ε is small compared to gradient components), and does the opposite if $\rho > \beta$. All these observations are exactly what Theorem 3.1 predicts.

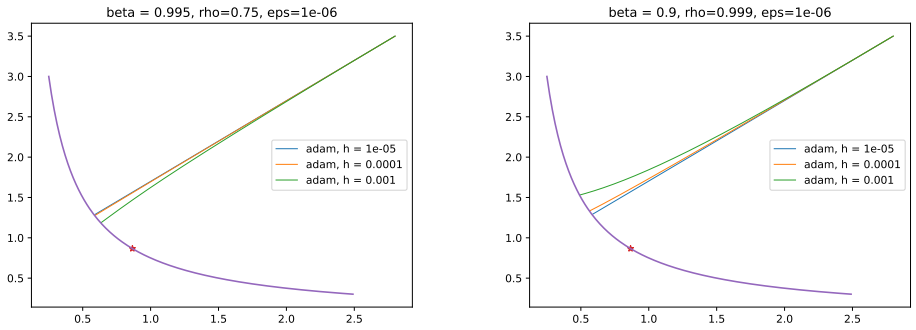

Figure 3: The setting is the same as in Figure 2. Increasing the learning rate moves the Adam trajectory towards the regions with smaller one-norm of the gradient if β is significantly larger than ρ and does the opposite if ρ is larger than β.

7 Numerical experiments

We offer some preliminary empirical evidence of how the first-order term shows up in deep neural networks.

[21] divides training regimes of Adam into three categories: the spike regime when ρ is sufficiently larger than β, in which the training loss curve contains very large spikes and the training process is obviously unstable; the (stable) oscillation regime when ρ is sufficiently close to β, in which the loss curve contains fast and small oscillations; the divergence regime when β is sufficiently larger than ρ, in which the optimization diverges. We of course exclude the last regime. Since it is very unlikely that an unstable Adam trajectory is close to the piecewise ODE emerging from backward error analysis, we exclude this regime as well, and confine ourselves to considering the oscillation regime (in which ρ and β do not have to be equal, but should not be too far apart). This is the regime [21] recommend to use in practice.

We train Resnet-50 on the CIFAR-10 dataset with full-batch Adam and calculate the quantity $\|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}$ at the first point the training loss drops below 0.01. Figure 4 shows that in the stable oscillation regime increasing ρ seems to increase the perturbed one-norm. This is consistent with backward error analysis (the smaller ρ, the more this "norm" is penalized). We also observe that increasing ρ seems to decrease the test accuracy (see the same figure). The opposite effect was noticed in [3], which we think is the case for the spike regime (where the trajectory of Adam is definitely far from the piecewise ODE trajectory at the later stages of training): it is intuitively plausible that increasing the number and magnitude of the spikes should increase the test accuracy by reducing overfitting.

The left part of Figure 5 shows that increasing β seems to decrease the perturbed one-norm. This is consistent with backward error analysis (the larger β, the more this norm is penalized). Similarly, the right part shows that increasing β seems to increase the test accuracy, if anything. Note that the effective learning rate does not depend on β (as is the case for gradient descent with heavy-ball momentum), so we compare training with the same effective learning rate. The picture confirms the finding in [8] (for momentum gradient descent) that increasing the momentum parameter almost always improves the test accuracy.

We also train Resnet-101 on CIFAR-10 with full-batch Adam, investigating how increasing β influences the training process. Figure 6 shows the training loss curves and the perturbed one-norm curve (the graphs of $\|\nabla E\|_{1,\varepsilon}$ as functions of the epoch number). Note that the training loss decreases monotonically to zero, the larger β the faster. The "norm" decreases, then rises again, and then decreases further until convergence. Throughout most of the training, the larger β the smaller the "norm" (so the norm behaves with respect to β in the opposite way the training loss does). The "hills" of the "norm" curves are higher with smaller β and almost unnoticeable when $\beta = \rho$. (This should be treated with caution: "hills" are not fully explained by $\rho > \beta$.) This is completely consistent with backward analysis because the larger ρ with respect to β, the more $\|\nabla E\|_{1,\varepsilon}$ is prevented from

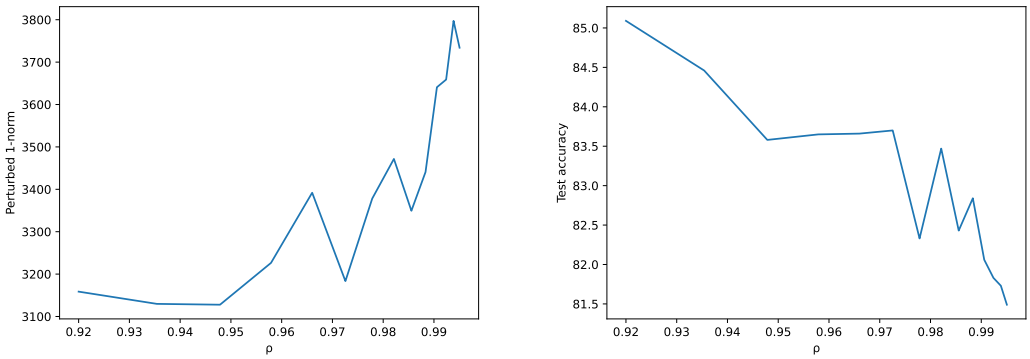

Figure 4: Resnet-50 on CIFAR-10 trained with full-batch Adam. The test accuracy seems to fall as ρ increases (in the stable "small oscillations" regime of training). The hyperparameters are as follows: $h = 7.5 \cdot 10^{-5}$, $\varepsilon = 10^{-8}$, $\beta = 0.99$. The test accuracies plotted here are maximal after more than 3600 epochs (they become almost constant much earlier). The perturbed norms are calculated at the same epoch number 900. (It is fair to compare Adam with different parameters at one epoch since the effective learning rates are the same.)

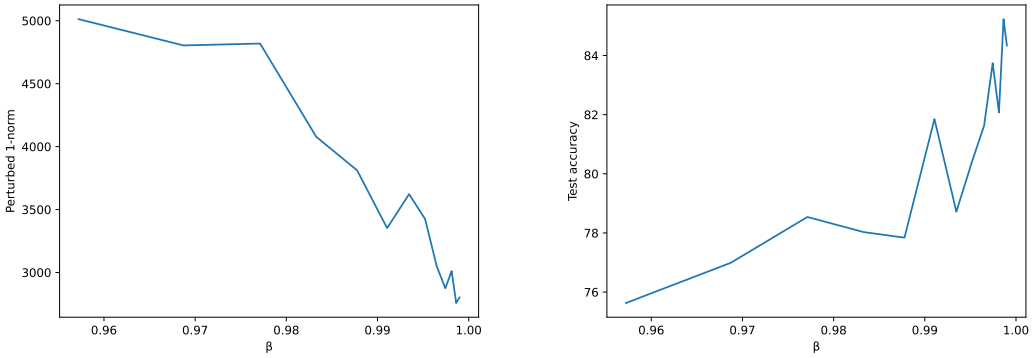

Figure 5: Resnet-50 on CIFAR-10 trained with full-batch Adam. The perturbed one-norm seems to fall as β increases (in the stable oscillation regime of training), and the test accuracy seems to rise. The hyperparameters are as follows: $h = 10^{-4}$, $\rho = 0.999$, $\varepsilon = 10^{-8}$. Both metrics are calculated when the loss first drops below the threshold 0.1.

falling by the bias term in (10). The last picture in Figure 6 shows how penalizing the "norm" seems to correspond to increasing the test accuracy in this case. Further evidence on how the perturbed one-norm behaves during training is available in Figure 7, where we train Resnet-101 on CIFAR-100 with increasing ρ. We see that the "hills" are there even if $\beta > \rho$, but their height seems to be larger for larger ρ.

8 Limitations and future directions

We think that backward error analysis applied to real-world machine learning optimization tasks has some limitations, some of them general to the whole literature and some of them specific to adaptive algorithms.

First, the assumption similar to (8) is either explicitly or implicitly present in all previous work on backward error analysis of gradient-based machine learning algorithms, as far as we know. This relatively weak assumption is not true if at least one activation function in the neural network is ReLU: the loss is not even differentiable (though it is very common to ignore this). Moreover, there

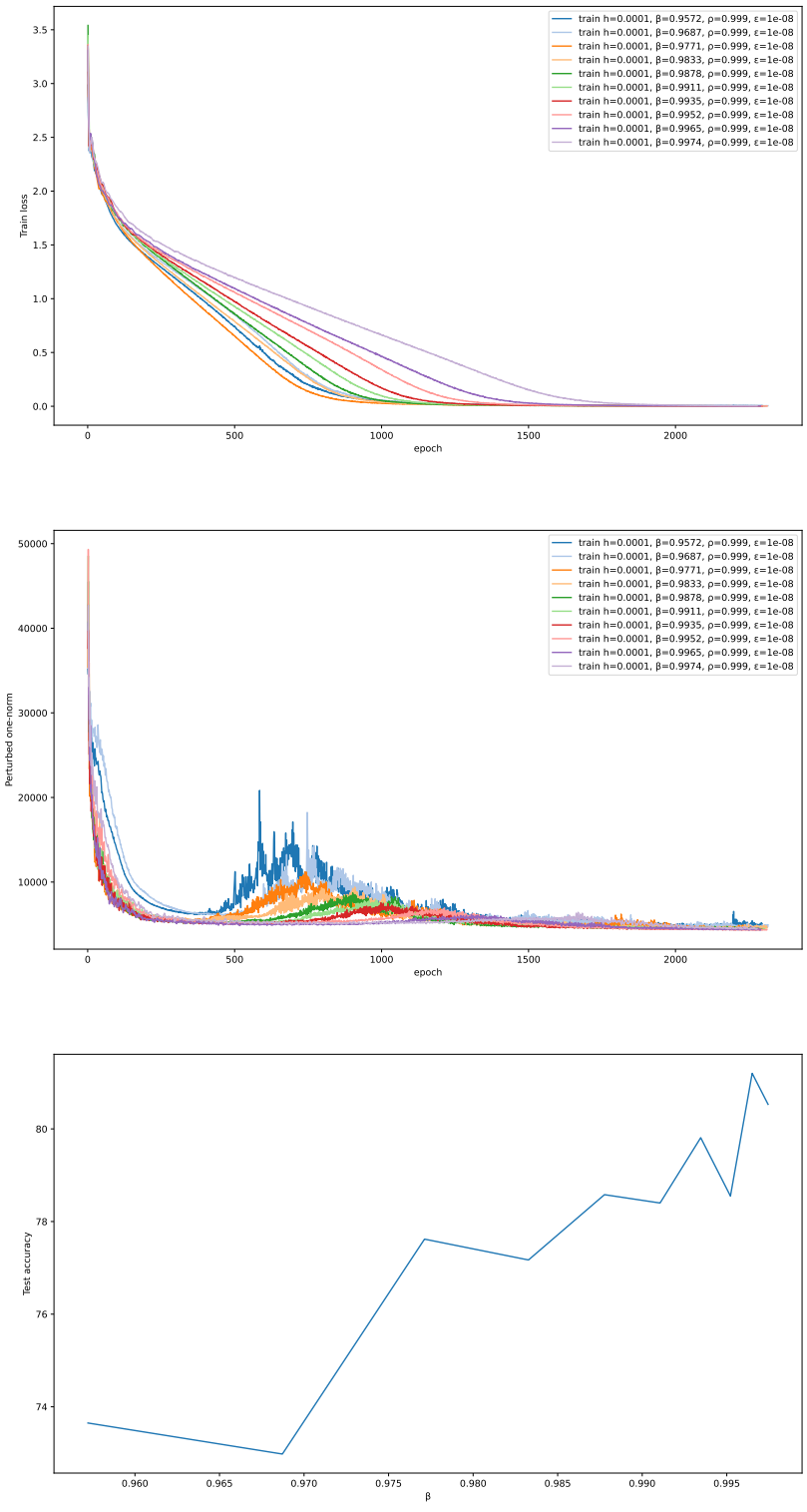

Figure 6: Resnet-101 on CIFAR-10 trained with full-batch Adam. First picture from the top: training loss curves. Second picture: curves plotting $\|\nabla E\|_{1,\varepsilon}$ after each epoch. Third picture: test accuracy the moment loss drops below 0.01 as a function of β. Hyperparameters: $h = 10^{-4}$, $\rho = 0.999$, $\varepsilon = 10^{-8}$.

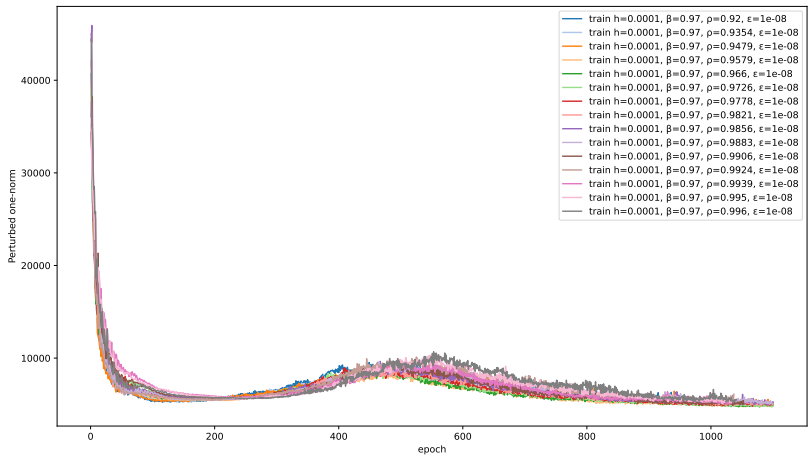

Figure 7: Resnet-101 trained on CIFAR-100 with full-batch Adam. We plot $\|\nabla E\|_{1,\varepsilon}$ after each epoch. Hyperparameters: $h = 10^{-4}$, $\beta = 0.97$, $\varepsilon = 10^{-8}$.

is evidence that large-batch algorithms often operate at the edge of stability ([2], [3]), in which the largest eigenvalue of the hessian can be quite large, making it unclear whether the higher-order partial derivatives can safely be assumed bounded near optimality.

Second, note that the constant in (10) depends on ε and goes to infinity as ε goes to zero. Theoretically, very small ε may mean that the trajectory of the piecewise ODE is only close to the actual Adam trajectory for unrealistically small learning rates, at least at the later stages of learning. (For the initial learning period, this is not a problem.) It is also true of Proposition 1 in [21]: the real trajectory may be far away even from the sign-GD dynamics. This is especially noticeable in the large-spike regime of training (see Section 7 and [21]) which, despite being obviously pretty unstable, can still minimize the training loss well and lead to acceptable test errors.

We believe that these considerations can fruitfully guide future work in this area.

Acknowledgments and Disclosure of Funding

We specially thank Boris Hanin for his insightful comments and suggestions. Cattaneo gratefully acknowledges financial support from the National Science Foundation through DMS-2210561 and SES-2241575. Klusowski gratefully acknowledges financial support from the National Science Foundation through CAREER DMS-2239448, DMS-2054808, and HDR TRIPODS CCF-1934924.

References

[1] David Barrett and Benoit Dherin. "Implicit Gradient Regularization". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=3q5IqUrkcF.

[2] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. "Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=jh-rTtvkGeM.

[3] Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. "Adaptive gradient methods at the edge of stability". In: *arXiv preprint arXiv:2207.14484* (2022).

[4] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[5] Christian Lubich Ernst Hairer and Gerhard Wanner. *Geometric numerical integration*. 2nd ed. Springer-Verlag, Berlin, 2006. ISBN: 3-540-30663-3.

[6] Mathieu Even, Scott Pesme, Suriya Gunasekar, and Nicolas Flammarion. "(S) GD over Diagonal Linear Networks: Implicit Regularisation, Large Stepsizes and Edge of Stability". In: *arXiv preprint arXiv:2302.08982* (2023).

[7] Guilherme França, Michael I Jordan, and René Vidal. "On dissipative symplectic integration with applications to gradient-based optimization". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.4 (2021), p. 043402.

[8] Avrajit Ghosh, He Lyu, Xitong Zhang, and Rongrong Wang. "Implicit regularization in Heavy-ball momentum accelerated stochastic gradient descent". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=ZzdBhtEH9yB.

[9] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. "Characterizing implicit bias in terms of optimization geometry". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1832–1841.

[10] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. "Implicit bias of gradient descent on linear convolutional networks". In: *Advances in neural information processing systems* 31 (2018).

[11] Ziwei Ji and Matus Telgarsky. "Directional convergence and alignment in deep learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17176–17186.

[12] Ziwei Ji and Matus Telgarsky. "Gradient descent aligns the layers of deep linear networks". In: *arXiv preprint arXiv:1810.02032* (2018).

[13] Ziwei Ji and Matus Telgarsky. "Risk and parameter convergence of logistic regression". In: *arXiv preprint arXiv:1803.07300* (2018).

[14] Ziwei Ji and Matus Telgarsky. "The implicit bias of gradient descent on nonseparable data". In: *Conference on Learning Theory*. PMLR. 2019, pp. 1772–1798.

[15] Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. "How Does Adaptive Optimization Impact Local Neural Network Geometry?" In: *arXiv preprint arXiv:2211.02254* (2022).

[16] Nitish Shirish Keskar and Richard Socher. "Improving generalization performance by switching from adam to sgd". In: *arXiv preprint arXiv:1712.07628* (2017).

[17] Diederick P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*. 2015.

[18] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel LK Yamins, and Hidenori Tanaka. "Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics". In: *arXiv preprint arXiv:2012.04728* (2020).

[19] Qianxiao Li, Cheng Tai, and Weinan E. "Stochastic Modified Equations and Adaptive Stochastic Gradient Algorithms". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2101–2110. URL: https://proceedings.mlr.press/v70/li17f.html.

[20] Kaifeng Lyu and Jian Li. "Gradient descent maximizes the margin of homogeneous neural networks". In: *arXiv preprint arXiv:1906.05890* (2019).

[21] Chao Ma, Lei Wu, and E Weinan. "A qualitative study of the dynamic behavior for adaptive gradient algorithms". In: *Mathematical and Scientific Machine Learning*. PMLR. 2022, pp. 671–692.

[22] Taiki Miyagawa. "Toward Equation of Motion for Deep Neural Networks: Continuous-time Gradient Descent and Discretization Error Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=qq84D17BPu.

[23] Mor Shpigel Nacson, Suriya Gunasekar, Jason Lee, Nathan Srebro, and Daniel Soudry. "Lexicographic and depth-sensitive margins in homogeneous and non-homogeneous deep models". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4683–4692.

[24] Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. "Convergence of gradient descent on separable data". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3420–3428.

[25] Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. "Stochastic gradient descent on separable data: Exact convergence with a fixed learning rate". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3051–3059.

[26] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond". In: *arXiv preprint arXiv:1904.09237* (2019).

[27] Mihaela C Rosca, Yan Wu, Benoit Dherin, and David Barrett. "Discretization drift in two-player games". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9064–9074.

[28] Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. "On the Origin of Implicit Regularization in Stochastic Gradient Descent". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=rq_Qr0c1Hyo.

[29] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. "The implicit bias of gradient descent on separable data". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 2822–2878.

[30] Tijmen Tieleman, Geoffrey Hinton, et al. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[31] Bohan Wang, Qi Meng, Wei Chen, and Tie-Yan Liu. "The Implicit Bias for Adaptive Optimization Algorithms on Homogeneous Neural Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 10849–10858. URL: https://proceedings.mlr.press/v139/wang21q.html.

[32] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. "Kernel and rich regimes in overparametrized models". In: *Conference on Learning Theory*. PMLR. 2020, pp. 3635–3673.

[33] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. "Adaptive Methods for Nonconvex Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/90365351ccc7437a1309dc64e4db32a3-Paper.pdf.

[34] Yang Zhao, Hao Zhang, and Xiuyuan Hu. "Penalizing gradient norm for efficiently improving generalization in deep learning". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 26982–26992.