# Backward Error Analysis of Adaptive Gradient Algorithms

**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 a different "norm" is 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 seem to influence generalization.

August 18, 2023

## 1 Introduction

Gradient descent can be seen as a numerical method solving the ordinary differential equation $\dot{\boldsymbol{\theta}} = -\nabla E(\boldsymbol{\theta})$, where $E(\cdot)$ is the loss function. Starting at $\boldsymbol{\theta}^{(0)}$, it creates a sequence of guesses $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$ that lie close to the solution trajectory $\boldsymbol{\theta}(t)$ of this equation. 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 $\boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}(nh)$, i. e. all the guesses of the descent lie exactly on the new solution curve or closer compared to the original curve. This is called backward error analysis in the numerical integration literature, see Chapter IX in (Ernst Hairer and Wanner 2006).

(D. Barrett and Dherin 2021) first used this idea for full-batch gradient descent and found that the modified loss function $\widetilde{E}(\tilde{\boldsymbol{\theta}}) = E(\tilde{\boldsymbol{\theta}}) + (h/4)\left\|\nabla E(\tilde{\boldsymbol{\theta}})\right\|^2$ makes the trajectory of the solution to $\dot{\tilde{\boldsymbol{\theta}}} = -\nabla \widetilde{E}(\tilde{\boldsymbol{\theta}})$ approximate the sequence $\left\{\boldsymbol{\theta}^{(n)}\right\}_{n=0}^{\infty}$ one order of $h$ better than the original differential equation.

(Miyagawa 2022) 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 extended to mini-batch gradient descent in (Smith, Dherin, D. Barrett, and De 2021). Assume that the training set is split in $B$ batches and there are $m$ batches per epoch (so the training

[*]Equal contribution

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 $\hat{E}_k(\boldsymbol{\theta}) = (1/B) \sum_{j=kB+1}^{kB+B} E_j(\boldsymbol{\theta})$. 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

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

(Ghosh, Lyu, X. Zhang, and R. Wang 2023) studied the gradient descent with heavy-ball momentum iteration $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - h\nabla E\left(\boldsymbol{\theta}^{(n)}\right) + \beta\left(\boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}^{(n-1)}\right)$, where $\beta$ is the momentum parameter. They proved that it is close to the continuous trajectory of the piecewise first-order ODE

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

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 (Zhao, H. Zhang, and Hu 2022), authors 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). Note that the only difference between this 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 (Tieleman, Hinton, et al. 2012) and Adam (Kingma and Ba 2015) is expected in the literature. (D. Barrett and Dherin 2021) 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”. (Smith, Dherin, D. Barrett, and De 2021) 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, (Ghosh, Lyu, X. Zhang, and R. Wang 2023) 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 (Miyagawa 2022) it is mentioned that “its [Adam’s] counter term and discretization error are open questions”.

We start filling this gap by conducting backward error analysis for (mini-batch, and full-batch as a special case) Adam and RMSProp. Our contributions:

- (Ma, L. Wu, and Weinan 2022) show that if the other hyperparameters are kept fixed, as the step-size goes to zero the continuous-time limit of both RMSProp and Adam are a (perturbed by $\varepsilon$) signGD flow
$$\dot{\boldsymbol{\theta}} = -\frac{\nabla E(\boldsymbol{\theta})}{|\nabla E(\boldsymbol{\theta})| + \varepsilon}$$
component-wise. We provide a higher order (in $h$) approximation in Theorem 3.3 and Theorem 3.6 in the general mini-batch setting.
- We discuss special cases in more detail. For the full-batch setting, we note that the correction term arising from backward error analysis 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). See Remark 3.4 and Remark 3.7 right after the corresponding theorems, and Section 4. We also obtain backward error analysis results for gradient descent and heavy-ball momentum gradient descent (which were derived before in (D. G. Barrett and Dherin 2020) and (Ghosh, Lyu, X. Zhang, and R. Wang 2023)) as special cases: see Example 3.5 and Example 3.8.

- 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.

**Notation**

We denote the loss of the $k$th 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 $E(\boldsymbol{\theta})$. As usual, $\nabla E$ means the gradient of $E$, and nabla 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 $\lVert \cdot \rVert$ is the two-norm of a vector, $\lVert \cdot \rVert_1$ is the one-norm and $\lVert \cdot \rVert_{1,\varepsilon}$ is the perturbed one-norm defined as $\lVert \mathbf{v} \rVert_{1,\varepsilon} = \sum_{i=1}^{p} \sqrt{v_i^2 + \varepsilon}$. (Of course, the perturbed one-norm is not a norm, but $\varepsilon = 0$ makes it the one-norm.)

## 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. (Kunin, Sagastuy-Brena, Ganguli, Yamins, and Tanaka 2020) show that the approximating continuous-time trajectories satisfy conservation laws that are broken in discrete time. (França, Jordan, and Vidal 2021) use backward error analysis while studying how to discretize continuous-time dynamical systems preserving stability and convergence rates. (Rosca, Y. Wu, Dherin, and D. Barrett 2021) find continuous-time approximations of discrete two-player differential games.

**Approximating gradient methods by differential equation trajectories.** (Ma, L. Wu, and Weinan 2022) 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 (Q. Li, Tai, and E 2017).

**Generalization of adaptive methods.** (B. Wang, Meng, Chen, and Liu 2021) study implicit bias in homogeneous neural networks trained by adaptive methods. (J. M. Cohen, Ghorbani, Krishnan, Agarwal, Medapati, Badura, Suo, Cardoze, Nado, Dahl, et al. 2022) 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. (Jiang, Malik, and Y. Li 2022) introduce a statistic that measures the uniformity of the hessian diagonal and argue that adaptive gradient algorithms are biased towards making this statistic smaller. (Keskar and Socher 2017) propose to improve generalization of adaptive methods by switching to SGD in the middle of training.

**Convergence of Adam.** (S. J. Reddi, Kale, and Kumar 2019) 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 (Zaheer, S. Reddi, Sachan, Kale, and Kumar 2018) 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).

## 3 ODEs approximating Adam and RMSProp trajectories

Adaptive algorithms use component-wise scaling of gradient updates according to the whole history of previous gradients (though usually involving an exponential moving average, forcing the algorithm to forget past iterations exponentially fast). In this paper, we focus on RMSProp and Adam which are the two algorithms most widely used in practice. To avoid ambiguity, we define them below.

## Definitions

**Definition 3.1.** 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 $n \geq 0$, $j \in \{1, \dots, p\}$

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

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 $n \geq 0$, $j \in \{1, \dots, p\}$

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

**Remark 3.2** (The $\varepsilon$ 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 $x \mapsto \left( \sqrt{x + \varepsilon} \right)^{-1}$ is bounded for $x \geq 0$. This is useful for our analysis. In the appendix,where exactly 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 $\varepsilon$ inside vs. outside the square root: tutorials with both versions abound on machine learning related websites. Moreover, the popular Tensorflow variant of RMSProp has $\varepsilon$ inside the square root[2] even though in the documentation[3] (Kingma and Ba 2015) is cited, where $\varepsilon$ is outside. While conducting numerical experiments, we also noted that moving $\varepsilon$ inside or outside the square root does not change the behavior of the algorithms qualitatively.

## Backward error analysis of RMSProp (and GD as a special case)

Our setting is general: let $E_k$ be the loss of $k$th (stochastic) mini-batch (in particular, a mini-batch may be just one data-point, though in practice usually it is at least 32 points), where $k \in \{0, 1, \dots\}$.

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

$$\sup_k \sup_{\boldsymbol{\theta}} \left\{ \sup_i |\nabla_i E_k(\boldsymbol{\theta})| \vee \sup_{i,j} |\nabla_{ij} E_k(\boldsymbol{\theta})| \vee \sup_{i,j,s} |\nabla_{ijs} E_k(\boldsymbol{\theta})| \vee \sup_{i,j,s,r} |\nabla_{ijsr} E_k(\boldsymbol{\theta})| \right\} \leq M. \tag{3}$$

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

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

**Theorem 3.3.** *Assume* (3) *holds. Let* $\left\{\boldsymbol{\theta}^{(n)}\right\}$ *be iterations of the modified RMSProp as defined in Definition 3.1,* $\tilde{\boldsymbol{\theta}}(t)$ *be the continuous solution to the piecewise ODE*

$$\dot{\tilde{\theta}}_j(t) = -\frac{\nabla_j E_n\left(\tilde{\boldsymbol{\theta}}(t)\right)}{R_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)} + h\left(\frac{\nabla_j E_n\left(\tilde{\boldsymbol{\theta}}(t)\right)\left(2P_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right) + \bar{P}_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)\right)}{2R_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)^3} - \frac{\sum_{i=1}^p \nabla_{ij} E_n\left(\tilde{\boldsymbol{\theta}}(t)\right)\frac{\nabla_i E_n\left(\tilde{\boldsymbol{\theta}}(t)\right)}{R_i^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)}}{2R_j^{(n)}(\tilde{\boldsymbol{\theta}}(t))}\right). \tag{4}$$

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

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

*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*

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

**Remark 3.4** (Backward error analysis of modified RMSProp in the full-batch setting)**.** In the full-batch setting $E_k \equiv E$, the terms in Theorem 3.3 simplify to

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

If the iteration number $n$ is large, (4) rapidly becomes

$$\dot{\tilde{\theta}}_j(t) = \frac{1}{\sqrt{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 + \varepsilon}}\left[-\nabla_j E(\tilde{\boldsymbol{\theta}}(t)) + h\frac{\frac{\rho}{1-\rho}\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 - \frac{\varepsilon}{2}}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 + \varepsilon}\nabla_j \left\|\nabla E(\boldsymbol{\theta})\right\|_{1,\varepsilon}\Big|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}(t)}\right]$$

with $\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^p \sqrt{v_i^2 + \varepsilon}$ the perturbed 1-norm. The factor

$$h\frac{\frac{\rho}{1-\rho}\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 - \frac{\varepsilon}{2}}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 + \varepsilon} = \underbrace{h\frac{-\frac{\varepsilon}{2}}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 + \varepsilon}}_{\text{nonzero steps}} + \underbrace{h\frac{\frac{\rho}{1-\rho}\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|^2 + \varepsilon}}_{\text{memory}}$$

is a sum of two terms. The first one is explained by the fact that the steps are not infinitely small, leading the discrete iterations to be different from the continuous trajectory. This term is present not only in RMSProp but also in discrete signGD without memory. As in the classical gradient descent case, this term penalizes the norm of the gradient (though in this case it is the corrected

1-norm instead of the squared two-norm). The second term is explained by memory and is specific to RMSProp. It seems to **reverse** the effect of penalization the first term provides.

In particular,

$$\dot{\tilde{\theta}}_j(t) \approx -\frac{1}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|}\nabla_j\left\{E(\tilde{\boldsymbol{\theta}}(t)) - h\frac{\rho}{1-\rho}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon}\right\} \quad \text{if } \left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \gg \sqrt{\varepsilon},$$

$$\dot{\tilde{\theta}}_j(t) \approx -\frac{1}{\sqrt{\varepsilon}}\nabla_j\left\{E(\tilde{\boldsymbol{\theta}}(t)) + \frac{h}{2}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon}\right\} \quad \text{if } \left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \ll \sqrt{\varepsilon}.$$

**Example 3.5** (Backward Error Analysis for GD)**.** If $\varepsilon$ is large compared to all gradient components, one has

$$\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon} \approx \sum_{i=1}^{p}\sqrt{\varepsilon}\left(1+\frac{\left|\nabla_i E(\tilde{\boldsymbol{\theta}}(t))\right|^2}{2\varepsilon}\right) = p\sqrt{\varepsilon} + \frac{1}{2\sqrt{\varepsilon}}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|^2, \tag{5}$$

and the ODE becomes

$$\dot{\tilde{\theta}}_j(t) \approx -\frac{1}{\sqrt{\varepsilon}}\nabla_j\left\{E(\tilde{\boldsymbol{\theta}}(t)) + \frac{h}{4\sqrt{\varepsilon}}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|^2\right\}.$$

This is exactly the same form as in (D. Barrett and Dherin 2021) up to two $1/\sqrt{\varepsilon}$ factors coming from a change in learning rate (and therefore time scale) since RMSProp can be seen as gradient descent with learning rate $h/\sqrt{\varepsilon}$ in this case.

## Backward Error Analysis of Adam (and GD with momentum as a special case)

**Theorem 3.6.** *Assume* (3) *holds. Let* $\left\{\boldsymbol{\theta}^{(n)}\right\}$ *be iterations of the modified Adam as defined in Definition 3.1,* $\tilde{\boldsymbol{\theta}}(t)$ *be the continuous solution to the piecewise ODE*

$$\dot{\tilde{\theta}}_j(t) = -\frac{M_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)}{R_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)} + h\left(\frac{M_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)\left(2P_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right) + \bar{P}_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)\right)}{2R_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)^3} - \frac{2L_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right) + \bar{L}_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)}{2R_j^{(n)}\left(\tilde{\boldsymbol{\theta}}(t)\right)}\right). \tag{6}$$

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

$$R_j^{(n)}(\boldsymbol{\theta}) := \sqrt{\sum_{k=0}^{n}\rho^{n-k}(1-\rho)\left(\nabla_j E_k(\boldsymbol{\theta})\right)^2/\left(1-\rho^{n+1}\right)+\varepsilon},$$

$$M_j^{(n)}(\boldsymbol{\theta}) := \frac{1}{1-\beta^{n+1}}\sum_{k=0}^{n}\beta^{n-k}(1-\beta)\nabla_j E_k(\boldsymbol{\theta}),$$

$$L_j^{(n)}(\boldsymbol{\theta}) := \frac{1}{1-\beta^{n+1}}\sum_{k=0}^{n}\beta^{n-k}(1-\beta)\sum_{i=1}^{p}\nabla_{ij}E_k(\boldsymbol{\theta})\sum_{l=k}^{n-1}\frac{M_i^{(l)}(\boldsymbol{\theta})}{R_i^{(l)}(\boldsymbol{\theta})},$$

$$\bar{L}_j^{(n)}(\boldsymbol{\theta}) := \frac{1}{1-\beta^{n+1}}\sum_{k=0}^{n}\beta^{n-k}(1-\beta)\sum_{i=1}^{p}\nabla_{ij}E_k(\boldsymbol{\theta})\frac{M_i^{(n)}(\boldsymbol{\theta})}{R_i^{(n)}(\boldsymbol{\theta})},$$

$$P_j^{(n)}(\boldsymbol{\theta}) := \frac{1}{1-\rho^{n+1}}\sum_{k=0}^{n}\rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta})\sum_{i=1}^{p}\nabla_{ij}E_k(\boldsymbol{\theta})\sum_{l=k}^{n-1}\frac{M_i^{(l)}(\boldsymbol{\theta})}{R_i^{(l)}(\boldsymbol{\theta})},$$

$$\bar{P}_j^{(n)}(\boldsymbol{\theta}) := \frac{1}{1-\rho^{n+1}}\sum_{k=0}^{n}\rho^{n-k}(1-\rho)\nabla_j E_k(\boldsymbol{\theta})\sum_{i=1}^{p}\nabla_{ij}E_k(\boldsymbol{\theta})\frac{M_i^{(n)}(\boldsymbol{\theta})}{R_i^{(n)}(\boldsymbol{\theta})}.$$

*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*

$$\left\| \tilde{\boldsymbol{\theta}}(nh) - \boldsymbol{\theta}^{(n)} \right\| \leq Ch^2, \quad n = 0, 1, \dots, \lfloor T/h \rfloor. \tag{7}$$

**Remark 3.7** (Backward error analysis of modified Adam in the full-batch setting)**.** In the full-batch setting $E_k \equiv E$, the terms in Theorem 3.6 simplify to

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

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

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

If the iteration number $n$ is large, (6) rapidly becomes

$$\begin{aligned}\dot{\tilde{\theta}}_j(t) = &- \frac{\nabla_j E\left(\tilde{\boldsymbol{\theta}}(t)\right)}{\sqrt{\left|\nabla_j E\left(\tilde{\boldsymbol{\theta}}(t)\right)\right|^2 + \varepsilon}} \\ &+ h \frac{\frac{1+\rho}{1-\rho}\left|\nabla_j E\left(\tilde{\boldsymbol{\theta}}(t)\right)\right|^2 - \frac{1+\beta}{1-\beta}\left(\left|\nabla_j E\left(\tilde{\boldsymbol{\theta}}(t)\right)\right|^2 + \varepsilon\right)}{2\left(\left|\nabla_j E\left(\tilde{\boldsymbol{\theta}}(t)\right)\right|^2 + \varepsilon\right)^{3/2}} \nabla_j \left\|\nabla E(\boldsymbol{\theta})\right\|_{1,\varepsilon}\Big|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}(t)}.\end{aligned}$$

(Putting $\beta = 0$ leads to the same expression as the one in Remark 3.4.)

In particular, if $\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \gg \sqrt{\varepsilon}$, we have

$$\dot{\tilde{\theta}}_j(t) \approx -\frac{1}{\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right|} \nabla_j \left\{E(\tilde{\boldsymbol{\theta}}(t)) + \frac{h}{2}\left(\frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho}\right)\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon}\right\},$$

and if $\left|\nabla_j E(\tilde{\boldsymbol{\theta}}(t))\right| \ll \sqrt{\varepsilon}$, we have

$$\dot{\tilde{\theta}}_j(t) \approx -\frac{1}{\sqrt{\varepsilon}} \nabla_j \left\{E(\tilde{\boldsymbol{\theta}}(t)) + \frac{h}{2} \cdot \frac{1+\beta}{1-\beta}\left\|\nabla E(\tilde{\boldsymbol{\theta}}(t))\right\|_{1,\varepsilon}\right\}.$$

**Example 3.8** (Backward Error Analysis for SGD with Heavy-ball Momentum)**.** Using (5) again, we see that if $\varepsilon$ is large compared to all gradient components, the ODE becomes

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

Since Adam with a large $\varepsilon$ 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 (Ghosh, Lyu, X. Zhang, and R. Wang 2023).

## 4 Discussion

**First conclusion.** Recall that from (Ghosh, Lyu, X. Zhang, and R. Wang 2023) 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 $\varepsilon$ is small) and Adam with $\rho > \beta$ (if $\varepsilon$ is small). This may partially explain why these algorithms generalize worse than their non-adaptive counterparts.

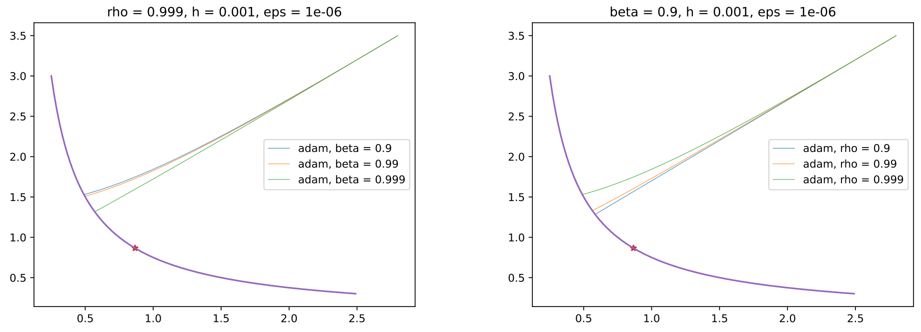

Figure 1: Increasing $\beta$ moves the trajectory of Adam towards the regions with smaller one-norm of the gradient (if $\varepsilon$ is sufficiently small); increasing $\rho$ does the opposite. The violet line is the line of global minima, and the cross denotes the limiting point of minimal one-norm of the gradient. All Adam trajectories start at $(2.8, 3.5)$.

**Second conclusion.** However, the first-order term in adaptive gradient algorithms does contain a kind of "norm" which is the perturbed one-norm $\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^{p} \sqrt{v_i^2 + \varepsilon}$. For Adam, the piecewise ODE is close to

$$\dot{\boldsymbol{\theta}} = -\frac{\nabla E(\boldsymbol{\theta})}{\sqrt{|\nabla E(\boldsymbol{\theta})|^2 + \varepsilon}} + \underbrace{h\frac{\frac{1+\rho}{1-\rho}|\nabla E(\boldsymbol{\theta})|^2 - \frac{1+\beta}{1-\beta}\left(|\nabla E(\boldsymbol{\theta})|^2 + \varepsilon\right)}{2\left(|\nabla E(\boldsymbol{\theta})|^2 + \varepsilon\right)^{3/2}}\nabla\|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}}_{\text{correction}}. \tag{8}$$

(For RMSProp, put $\beta = 0$.) In particular, if all components of $\nabla E(\boldsymbol{\theta})$ are large compared to $\sqrt{\varepsilon}$, which is the case during the initial learning stage, this is just

$$\dot{\boldsymbol{\theta}} = -\frac{1}{|\nabla E(\boldsymbol{\theta}|}\nabla\left\{E(\boldsymbol{\theta}) + \frac{h}{2}\left(\frac{1+\beta}{1-\beta} - \frac{1+\rho}{1-\rho}\right)\|\nabla E(\boldsymbol{\theta})\|_{1,\varepsilon}\right\}.$$

We can conclude with moderate confidence that it is only in the case $\beta > \rho$ that the perturbed norm *is* penalized, and decreasing $\rho$ or increasing $\beta$ 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 $\beta$ and decreasing $\rho$ should improve the test error.

## 5 Illustration: simple bilinear model

We now analyze the effect of the first-order term for Adam in the same model as (D. Barrett and Dherin 2021) and (Ghosh, Lyu, X. Zhang, and R. Wang 2023) 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 1 shows that increasing $\beta$ forces the trajectory to the region with smaller 1-norm of the gradient of the loss $\|\nabla E(\boldsymbol{\theta})\|_1$, and increasing $\rho$ does the opposite. Figure 2 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 $\varepsilon$ is small compared to gradient components), and does the opposite if $\rho > \beta$. All these observations are exactly what Theorem 3.6 predicts.

## 6 Numerical experiments

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

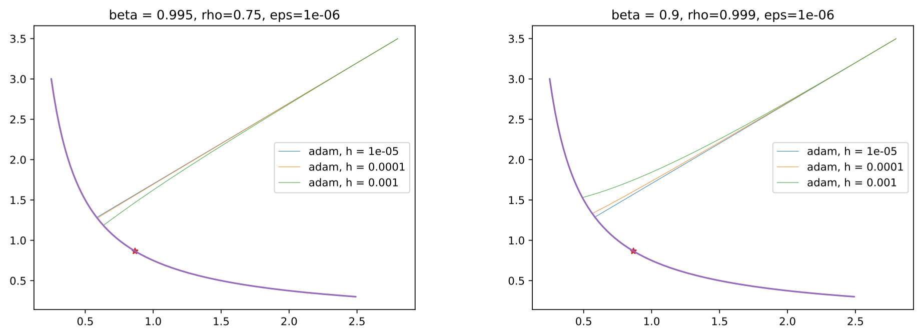

Figure 2: The setting is the same as in Figure 1. Increasing the learning rate moves the Adam trajectory towards the regions with smaller one-norm of the gradient if $\beta$ is significantly larger than $\rho$ and does the opposite if $\rho$ is larger than $\beta$.

(Ma, L. Wu, and Weinan 2022) divides training regimes of Adam into three categories: the spike regime when $\rho$ is sufficiently larger than $\beta$, in which the training loss curve contains very large spikes and the training process is obviously unstable; the (stable) oscillation regime when $\rho$ is sufficiently close to $\beta$, in which the loss curve contains fast and small oscillations; the divergence regime when $\beta$ is sufficiently larger than $\rho$, 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 $\rho$ and $\beta$ do not have to be equal, but should not be too far apart). This is the regime (Ma, L. Wu, and Weinan 2022) 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 3 shows that in the stable oscillations regime increasing $\rho$ seems to increase the perturbed one-norm. This is consistent with backward error analysis (the smaller $\rho$, the more this "norm" is penalized). We also observe that increasing $\rho$ seems to decrease the test accuracy: see Figure 4. The opposite effect was noticed in (J. M. Cohen, Ghorbani, Krishnan, Agarwal, Medapati, Badura, Suo, Cardoze, Nado, Dahl, et al. 2022), 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).

Figure 5 shows that increasing $\beta$ seems to decrease the perturbed one-norm. This is consistent with backward error analysis (the larger $\beta$, the more this norm is penalized). Similarly, Figure 6 shows that increasing $\beta$ increases the test accuracy. Note that the effective learning rate does not depend on $\beta$ (as is the case for gradient descent with heavy-ball momentum), so we compare training with the same effective learning rate.

We also train Resnet-101 on CIFAR-10 with full-batch Adam, investigating how increasing $\beta$ influences the training process. Figure 7 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 $\beta$ the faster. The "norm" decreases, then rises again, and then decreases further until convergence. Throughout most of the training, the larger $\beta$ the smaller the "norm" (so the norm behaves with respect to $\beta$ in the opposite way the training loss does). The "hills" of the "norm" curves are higher with smaller $\beta$ 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 $\rho$ with respect to $\beta$, the more $\|\nabla E\|_{1,\varepsilon}$ is prevented from falling by the correction term in (8). The last picture in Figure 7 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 8, where we train Resnet-101 on CIFAR-100 with increasing $\rho$. We see that the "hills" are there even if $\beta > \rho$, but their height is larger for larger $\rho$.

We do not claim that the correlations observed are always true: that would mean that some hyperparameter values are universally optimal, which, as the industry knows very well, is not the case. We

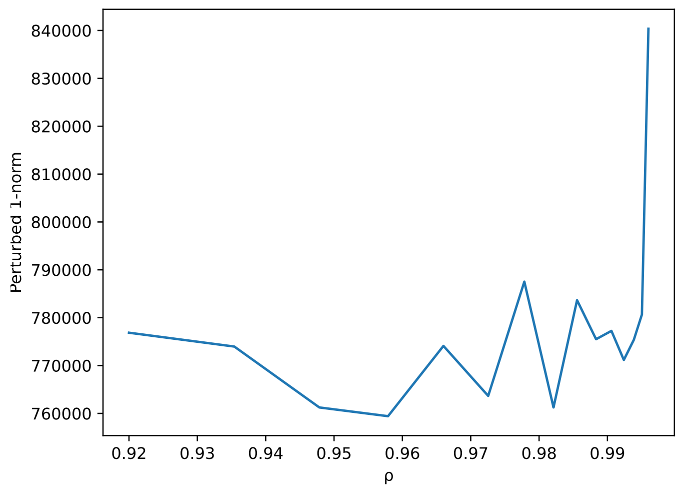

Figure 3: Resnet-50 on CIFAR-10 trained with full-batch Adam. The perturbed one-norm seems to rise as $\rho$ increases (in the stable “small oscillations” regime of training). The hyperparameters are as follows: $\beta = 0.97$, $h = 7.5 \cdot 10^{-5}$, $\varepsilon = 10^{-3}$. First half of Exp6

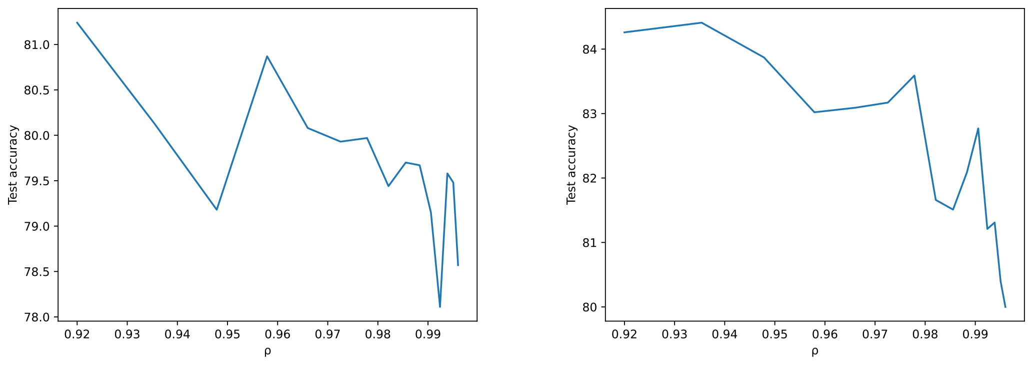

Figure 4: Resnet-50 on CIFAR-10 trained with full-batch Adam. The test accuracy seems to fall as $\rho$ increases (in the stable “small oscillations” regime of training). The hyperparameters are as follows: $h = 7.5 \cdot 10^{-5}$, $\varepsilon = 10^{-3}$; on the left $\beta = 0.97$, on the right $\beta = 0.99$. The test accuracies are calculated at the first point the training loss falls below 0.01. Exp6

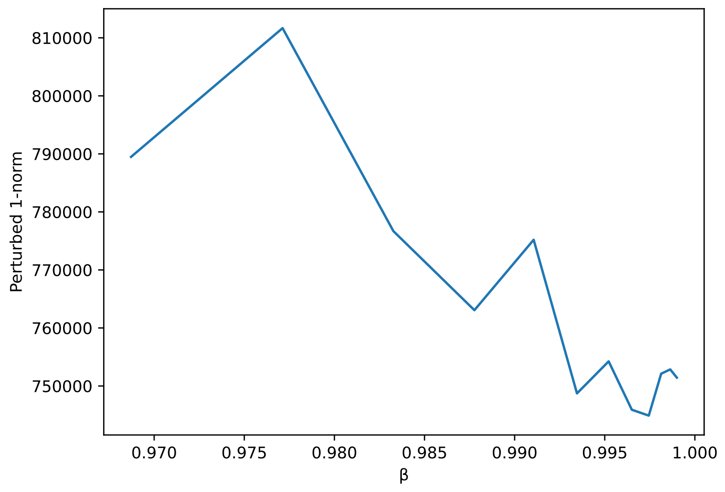

Figure 5: Resnet-50 on CIFAR-10 trained with full-batch Adam. The perturbed one-norm seems to fall as $\beta$ increases (in the stable oscillation regime of training). The hyperparameters are as follows: $h = 10^{-4}$, $\rho = 0.999$, $\varepsilon = 10^{-3}$. First half of Exp5

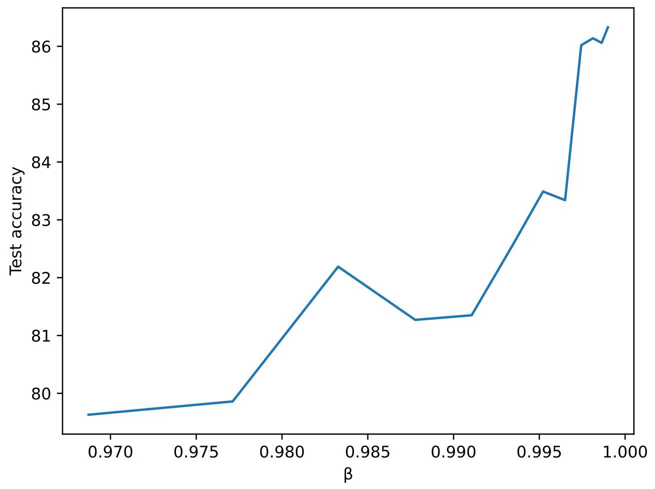

Figure 6: Resnet-50 on CIFAR-10 trained with full-batch Adam. The test accuracy seems to rise as $\beta$ increases (in the stable "small oscillations" regime of training). The hyperparameters are as follows: $h = 10^{-4}$, $\rho = 0.999$, $\varepsilon = 10^{-3}$. First half of Exp5

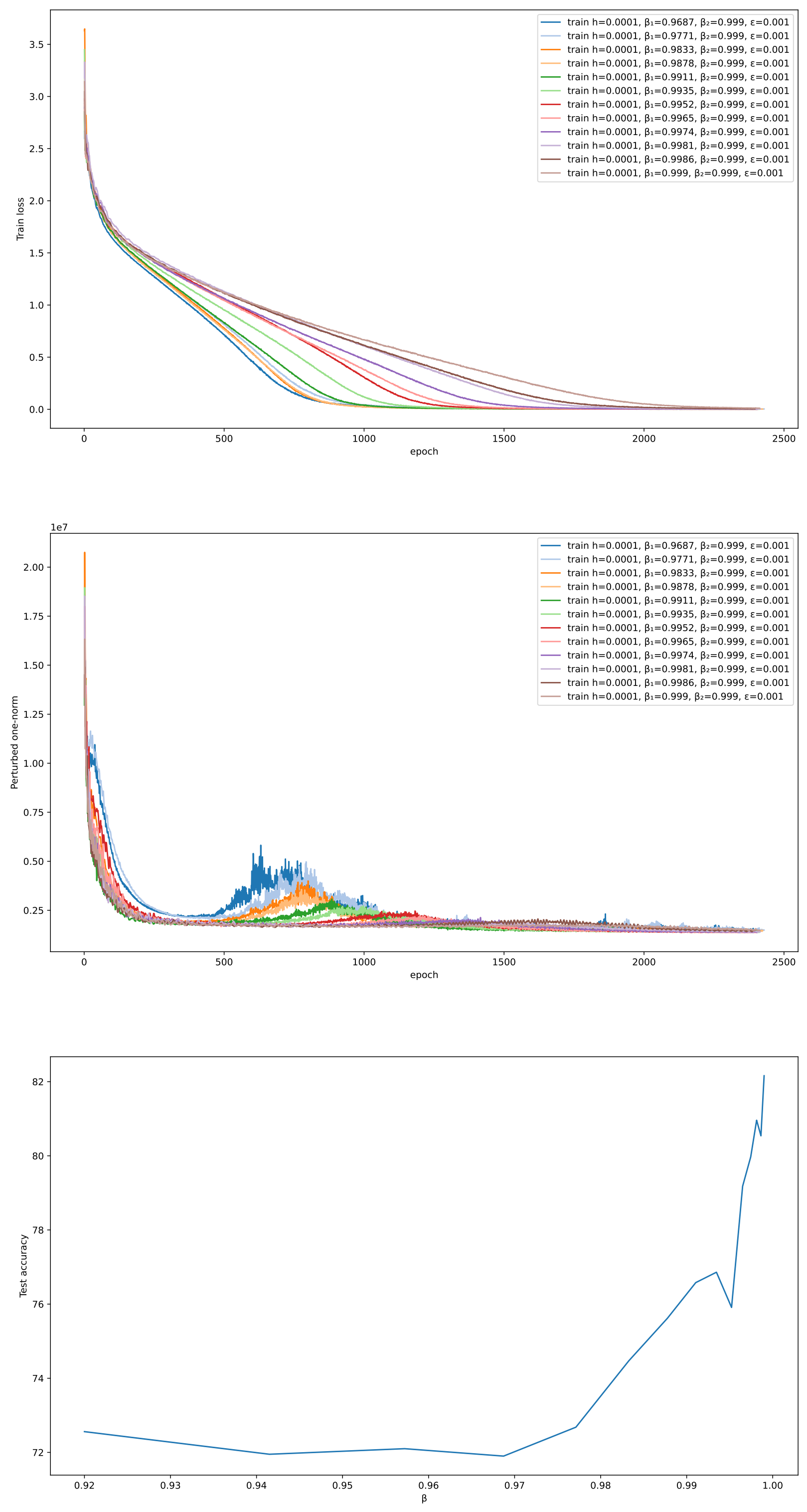

Figure 7: Resnet-101 on CIFAR-10 trained with full-batch Adam. First picture from the top: training loss curves. Second picture: curves plotting $\|\nabla E\|_{1,\varepsilon}$ after each epoch. Third picture: test accuracy the moment loss drops below 0.1 as a function of $\beta$. Hyperparameters: $h = 10^{-4}$, $\rho = 0.999$, $\varepsilon = 10^{-3}$. First half of Exp12

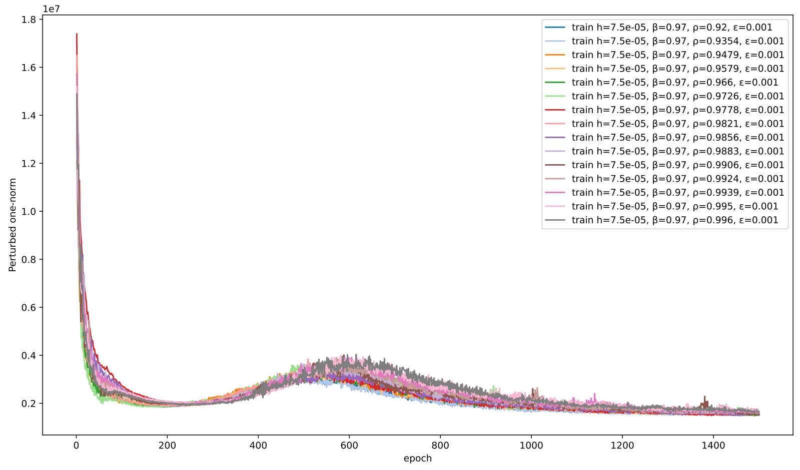

Figure 8: Resnet-101 trained on CIFAR-100 with full-batch Adam. We plot $\|\nabla E\|_{1,\varepsilon}$ after each epoch. Hyperparameters: $h = 7.5 \cdot 10^{-5}$, $\beta = 0.97$, $\varepsilon = 10^{-3}$. Second half of Exp13

only claim that they are true sometimes, and it can be explained at least partially by backward error analysis. In other cases different effects can conflict, such as the components of the hessian growing and the instability (or near-instability) of the training regime.

## 7 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 (3) is either explicitly or implicitly present in all previous work on backward error analysis in machine learning, 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 is evidence that large-batch algorithms often operate at the edge of stability (J. Cohen, Kaur, Y. Li, Kolter, and Talwalkar 2021, J. M. Cohen, Ghorbani, Krishnan, Agarwal, Medapati, Badura, Suo, Cardoze, Nado, Dahl, et al. 2022), 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.cite Pier?

Second, note that the constant in (7) depends on $\varepsilon$ and goes to infinity as $\varepsilon$ goes to zero. Theoretically, very small $\varepsilon$ 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 (Ma, L. Wu, and Weinan 2022): 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 6 and Ma, L. Wu, and Weinan 2022) 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.

## References

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

Barrett, David GT and Benoit Dherin (2020). "Implicit gradient regularization". In: *arXiv preprint arXiv:2009.11162*.

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

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

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

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

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

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

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

Kingma, Diederick P and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*.

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

Li, Qianxiao, Cheng Tai, and Weinan E (June 2017). "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, pp. 2101–2110. URL: https://proceedings.mlr.press/v70/li17f.html.

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

Miyagawa, Taiki (2022). "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. URL: https://openreview.net/forum?id=qq84D17BPu.

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

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

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

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

Wang, Bohan, Qi Meng, Wei Chen, and Tie-Yan Liu (18–24 Jul 2021). "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, pp. 10849–10858. URL: https://proceedings.mlr.press/v139/wang21q.html.

Zaheer, Manzil, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar (2018). "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. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/90365351ccc7437a1309dc64e4db32a3-Paper.pdf.

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