A Continuous-Time Analysis of Adaptive Optimization and Normalization

Rhys Gould[1], **Hidenori Tanaka**[2,3]
[1] Department of Applied Mathematics & Theoretical Physics, University of Cambridge
[2] Center for Brain Science, Harvard University
[3] Physics & Informatics Laboratories, NTT Research, Inc.

Abstract

Adaptive optimization algorithms, particularly Adam and its variant AdamW, are fundamental to modern deep learning, however, their training dynamics lack comprehensive theoretical understanding, with limited insight into why common practices—such as specific hyperparameter choices and normalization layers—contribute to successful generalization. This work presents a continuous-time formulation of Adam and AdamW, facilitating a tractable analysis of training dynamics that can shed light on such practical questions. We theoretically derive a stable region for Adam's hyperparameters (β, γ) that ensures bounded updates, empirically verifying these predictions by observing unstable exponential growth of parameter updates outside this region. Furthermore, we theoretically justify the success of normalization layers by uncovering an implicit meta-adaptive effect of scale-invariant architectural components. This insight leads to an explicit optimizer, 2-Adam, which we generalize to k-Adam—an optimizer that applies an adaptive normalization procedure k times, encompassing Adam (corresponding to $k = 1$) and Adam with a normalization layer (corresponding to $k = 2$). Overall, our continuous-time formulation of Adam facilitates a principled analysis, offering deeper understanding of optimal hyperparameter choices and architectural decisions in modern deep learning.

1 Introduction

Adaptive optimization algorithms have become an essential component of modern deep learning, providing significant benefits to the training of neural networks compared to their non-adaptive counterparts. Among these algorithms, Adam [1] and its variant AdamW [2] have become widely used in practice, featuring both an adaptive learning rate (in the sense of RMSprop [3]) and an adaptive gradient direction (in the sense of momentum [4]). Despite their wide success, there is limited theoretical insight into why common practices – such as typical hyperparamter choice, and the use of normalization layers (e.g. layer-norm) – contribute towards successful generalization. In this work, we demonstrate the utility of continuous-time models of optimization in shedding light on such questions. A continuous-time approach allows for a mathematically tractable analysis that can leverage the tools of calculus (i.e. differential equations). Our contributions can be summarized as:

1. Continuous-time formulation (Section 2)

(a) We derive an explicit continuous-time expression for Adam's (& AdamW's) parameter update which we directly utilize in Section 3 (see contribution 2), and an associated second-order differential equation which we directly utilize in Section 4 (see contribution 3).

2. Theory of adaptive hyperparameters (Section 3)

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

(b) We determine a theoretical stability region for Adam's adaptive hyperparameters (β, γ) that ensures bounded parameter updates and stable training, which we verify empirically.
(c) This result implies that instability may occur when the hyperparameters move outside the stable region, a phenomenon we empirically verify, exhibiting *predictable* exponential growth.

3. Implicit effect of scale invariance (Section 4)

(d) We perform a theoretical analysis of the *implicit* beneficial effect of scale invariant architectural components (e.g. layer-norm), uncovering a *meta-adaptive* normalization effect.
(e) We convert this implicit effect to an *explicit* optimizer, which we name 2-Adam, and consider its extension k-Adam: a generalization of Adam/AdamW (which corresponds to $k = 1$) that performs a normalization procedure k times successively.

2 Continuous-time formulation of Adam

In this section we present a continuous-time formulation of the Adam (& AdamW) optimizer which forms the theoretical foundation for the rest of the paper. We provide brief descriptions of relevant optimizers (momentum, RMSprop, Adam, AdamW) in Appendix C. Experimental details are left to Appendix H.

Notation. We write $||x|| := \sqrt{\sum_i x_i^2}$ and $||x||_\infty := \max_i |x_i|$ for $x \in \mathbb{R}^p$, and denote the inner product $\langle x, y\rangle \equiv \sum_i x_i y_i$. We denote elementwise squaring by $x^{\odot 2}$. We consider a loss function $L : \Theta \to \mathbb{R}$ where $\Theta \subset \mathbb{R}^p$, and $\theta_n \in \Theta$ denotes a model's parameters after n discrete updates, with corresponding gradient $g_n := \nabla_\theta L(\theta_n)$, and initial parameter θ_0. We will write $g_{0:n} \equiv (g_0, \ldots, g_n)$.

2.1 Adam in continuous-time

Neglecting weight decay for the moment (i.e. $\lambda = 0$), Adam & AdamW possess the discrete-time update rule

$$\theta_{n+1} = \theta_n - \eta u_n, \quad \text{where} \quad u_n := \frac{\sqrt{1-\beta^{n+1}}}{1-\gamma^{n+1}} \frac{m_n}{\sqrt{v_n}} \tag{1}$$

with learning rate η, and *adaptive hyperparameters* $(\beta, \gamma) \in [0, 1]^2$ with moving-averages

$$m_n := \gamma m_{n-1} + (1-\gamma) g_n, \qquad v_n := \beta v_{n-1} + (1-\beta) g_n^{\odot 2}$$

where $(m_{-1}, v_{-1}) := (0, 0)$. We say that u_n is an *adaptive-normalization* of the gradients $g_{0:n}$ (see Appendix R for more details). We will work in continuous-time, at a timescale of η^p, such that e.g. $m_n = m(t_n)$, with $t_n := n\eta^p$ for some $p \in \mathbb{R}$ (typically $p = 1$). We can then derive continuous-time expressions for the moving-averages m_n and v_n (details left to Appendix E), which (to leading order in η^p) take the form

$$m(t) = \int_0^t d\tau \; K_\gamma(\tau, t) g(\tau), \quad v(t) = \int_0^t d\tau \; K_\beta(\tau, t) g(\tau)^{\odot 2} \tag{2}$$

with $g(\tau) := \nabla_\theta L(\theta(\tau))$ the gradient at continuous time τ and K_α defined below. This provides a continuous-time expression for Adam/AdamW's update $u_n = u(t_n)$,

$$u(t) = \frac{\sqrt{1-\beta^{1+t/\eta^p}}}{1-\gamma^{1+t/\eta^p}} \frac{\int_0^t d\tau \; K_\gamma(\tau, t) g(\tau)}{\sqrt{\int_0^t d\tau \; K_\beta(\tau, t) g(\tau)^{\odot 2}}}, \quad K_\alpha(\tau, t) := \frac{1-\alpha}{\eta^p \alpha} \exp\left(-\frac{1-\alpha}{\eta^p \alpha}(t-\tau)\right) \tag{3}$$

Including weight decay is simple and is described in Appendix E. As we will see, Equation (3) is a central result for Section 3, allowing us to derive theoretical guarantees regarding training stability with respect to Adam/AdamW's adaptive hyperparameters (β, γ). From Equation (1) we can also determine $\theta_n = \theta(t_n)$ in continuous-time via a governing differential equation (details left to Appendix E). To second-order, AdamW with weight decay λ is governed by the differential equation

$$\lambda \eta \theta(t) + \eta^p \dot{\theta}(t) + \frac{\eta^{2p}}{2} \ddot{\theta}(t) = -\eta u(t) \tag{4}$$

In combination with the continuous-time expression for $u(t)$ (Equation (3)), we can numerically solve Equation (4) (method described in Appendix I) to obtain a continuous-time parameter trajectory

$\theta(t)$. We demonstrate that the continuous-time trajectory (for timescale $p = 1$) associated with Equation (4) closely agrees with the true discrete-time trajectory in Appendix K. In Section 4 we will use Equation (4) to interpret the implicit effect of scale invariance.

3 Theory of adaptive hyperparameters

In this section we will present a theory-driven account of adaptive hyperparameter choice for Adam/AdamW, understanding the values of (β, γ) that result in stable training and effective generalization. First we will use the continuous-time expression for Adam's update (Equation (3)) to derive a theoretical region of hyperparameter space $\mathcal{B}_+$ for which updates are provably bounded (Section 3.1). We will then empirically verify that this region indeed exhibits stable training, with unstable training in the complementary region $\mathcal{B}_-$ (exhibiting a *predictable* exponential growth), and observe how generalization performance varies in $\mathcal{B}_+$ (Section 3.2).

3.1 Deriving a condition for bounded updates

The continuous-time expression for Adam's update (Equation (3)) has an immediate consequence in regards to bounding the *max-update* $||u_n||_\infty \equiv ||\theta_{n+1} - \theta_n||_\infty/\eta$. We have the following bound,

Max-update bound. *The max-update can be bounded as*

$$||u_n||_\infty \leq \frac{\sqrt{1-\beta^{n+1}}}{1-\gamma^{n+1}} \frac{1-\gamma}{\gamma} \sqrt{\frac{\beta}{1-\beta}} B_n(\beta, \gamma) \tag{5}$$

where

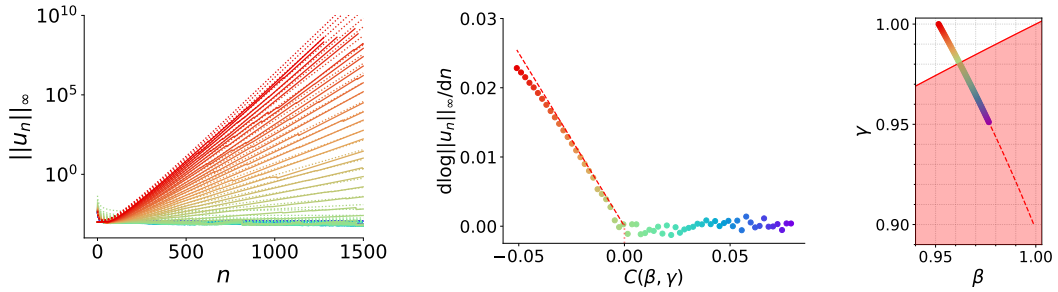
$$B_n(\beta, \gamma) := \begin{cases} 1/\sqrt{C(\beta,\gamma)}, & C(\beta,\gamma) > 0, \\ \sqrt{n}, & C(\beta,\gamma) = 0, \\ \exp(n|C(\beta,\gamma)|/2)/\sqrt{|C(\beta,\gamma)|}, & C(\beta,\gamma) < 0 \end{cases}$$

defining $C(\beta, \gamma) := (2\beta(1-\gamma) - \gamma(1-\beta))/\beta\gamma$. *We leave the derivation of this bound to Appendix E.*

We highlight that this bound is only possible because of the specific form of Adam's update $u(t)$: a moving-average of g divided by the square-root of a moving-average of $g^{\odot 2}$ (Equation (3)), i.e. an adaptive-normalization of g (see Appendix R), which allows us to apply the Cauchy-Schwarz inequality. We can therefore view this result as a theoretical justification for the form of Adam's update. We will define the bounded-update region $\mathcal{B}_+ := \{(\beta, \gamma) : C(\beta, \gamma) > 0\}$, and the complementary region $\mathcal{B}_- := \{(\beta, \gamma) : C(\beta, \gamma) < 0\}$. From Equation (5), we can bound the max-update by a constant independent of n when $(\beta, \gamma) \in \mathcal{B}_+$. Outside of $\mathcal{B}_+$, the bound depends on n and diverges over training as $n \to \infty$. This is suggestive that we may observe stable training when $(\beta, \gamma) \in \mathcal{B}_+$, and that the max-update may grow exponentially (at a rate/exponent proportional to $|C(\beta, \gamma)|$) when $(\beta, \gamma) \in \mathcal{B}_-$. Indeed, we will verify this phenomena empirically in Section 3.2. We comment on the case $(\beta, \gamma) \in \mathcal{B}_0$ in Appendix F. It is easy to show that $\beta > \gamma$ is a sufficient condition for $(\beta, \gamma) \in \mathcal{B}_+$, meaning that the choice of hyperparameters typically chosen in practice – e.g. the PyTorch default values $(\tilde{\beta}, \tilde{\gamma}) := (0.999, 0.9)$ – lie within $\mathcal{B}_+$. For the following, we define the *level curves* $\mathcal{B}_c := \{(\beta, \gamma) : C(\beta, \gamma) = c\}$, and consider *normal curves* perpendicular to these level curves.We will denote the (unique) normal curve passing through the point (β, γ) by $\mathcal{C}_{\beta,\gamma}$. Note that, by construction, $C(\beta, \gamma)$ varies monotonically along a normal curve.

3.2 Empirical analysis of max-update

We now consider the empirical implications of Equation (5), assessing whether these bounds indeed hold in practice and whether $C(\beta, \gamma)$ is predictive of training stability. We train a decoder-only transformer model on a shakespeare text dataset (details left to Appendix H) and observe how the max-update $||u_n||_\infty$ evolves over training iterations n. In Figure 1, we consider the normal curve $\mathcal{C}_{\tilde{\beta},\tilde{\gamma}}$ passing through the typical hyperparameter values $(\tilde{\beta}, \tilde{\gamma})$, taking 64 points (β, γ) uniformly along this curve (which we visualize in Figure 1c). For each point (β, γ), we plot the max-update trajectory (Figure 1a), finding that the theoretical bounds of Equation (5) are satisfied in both regions $\mathcal{B}_+$ and $\mathcal{B}_-$, observing well-behaved bounded growth in $\mathcal{B}_+$, whereas $\mathcal{B}_-$ exhibits exponential growth at a rate that appears correlated with $|C(\beta, \gamma)|$ as predicted by Equation (5). We more closely verify this



(a) (b) (c)

Figure 1: **Max-update bound accurately predicts stable region and unstable exponent of divergence.** We consider 64 hyperparameter points (β, γ) taken uniformly along a section of the normal curve $\mathcal{C}_{\tilde{\beta},\tilde{\gamma}}$ passing through the point $(\tilde{\beta}, \tilde{\gamma}) = (0.999, 0.9)$, which we visualize in (c). For each of these points, we plot (a) the max-update $||u_n||_\infty \equiv ||\theta_n - \theta_{n-1}||_\infty/\eta$ over training iterations n, and (b) the slope $d \log ||u_n||_\infty/dn$ of the log-max-update at iteration $n = 500$ (in order to interpret exponential growth). In (a) we visualize the bounds of Equation (5) as dotted lines, and in (b) we denote the predicted slope/exponent $|C(\beta, \gamma)|/2$ (when $C(\beta, \gamma) < 0$) as a dashed line.

phenomena in Figure 1b, finding that in $\mathcal{B}_-$, the slope $d \log ||u_n||_\infty/dn$ has a near-perfect agreement with the theoretically predicted growth rate of $|C(\beta, \gamma)|/2$. It is surprising that not only are the theoretical bounds (Equation (5)) satisfied in practice, but the bounds are very accurate models of the true empirical dynamics, with exponential growth occurring almost immediately after entering $\mathcal{B}_-$. We look more closely at the results of Figure 1a in the case of $(\beta, \gamma) \in \mathcal{B}_+$ in Appendix M. We comment on correcting the exponential growth in $\mathcal{B}_-$ via learning rate annealing in Appendix P.

These results partly justify the success of typical values $(\tilde{\beta}, \tilde{\gamma}) = (0.999, 0.9)$ in practice: these values lie in $\mathcal{B}_+$ and hence benefit from theoretical guarantees regarding training stability (which, as we have seen, are faithful to practice). Can we further obtain a more fine-grained picture as to what choices of (β, γ) within the region $\mathcal{B}_+$ will result in successful generalization? We assess how the generalization performance varies in $\mathcal{B}_+$ in Appendix N, the results of which suggest that for a given normal curve, the point (β, γ) with the largest value of $C(\beta, \gamma)$ generalizes the fastest along the curve. This would further justify why the values $(\tilde{\beta}, \tilde{\gamma})$ succeed in practice (since $\tilde{\beta} \approx 1$, hence $C(\tilde{\beta}, \tilde{\gamma})$ is large relative to other points along its normal curve), however unlike the results of Section 3.2, this claim lacks an explicit supporting theoretical result; it is only suggestive from Equation (5) that a larger value of $C(\beta, \gamma)$ may correlate with better training properties. We leave direct theoretical guarantees regarding the rate of generalization to future work.

4 Implicit effect of scale invariance

The presence of scale-invariant architectural components has become ubiquitous in modern machine learning. One particular instance of such components are normalization layers, such as layer-norm [5], batch-norm [6], and qk-norm [7, 8]. It has been observed in practice that such normalization provides an implicit beneficial effect to training, compared to e.g. explicitly fixing/constraining the weight norm after each update step [9, 10, 11]. In this section we will use the second-order differential equation for Adam/AdamW (Equation (4)) to solve for the norm dynamics and uncover an implicit *meta-adaptive* effect of scale invariance (Section 4.1). Furthermore, we will introduce the k-Adam optimizer, a generalization of Adam that explicitly models this implicit effect (Section 4.2).

4.1 Uncovering an implicit meta-adaptive effect

A function $f : \Theta \to \mathbb{R}$ is *scale-invariant* with respect to a weight $W \in \mathcal{W} \subset \Theta$ if and only if $f(\theta)$ remains unchanged under the transformation $W \mapsto \alpha W$ for all $\alpha \in \mathbb{R}$. One can show (see Appendix L) that if L is scale-invariant with respect to a weight W, then

$$\langle W(t), g_W(t) \rangle = 0 \tag{6}$$

at all times t, where $g_W(t) := \nabla_W L(\theta(t))$ is the gradient associated with W. To theoretically analyse the implicit effect of scale invariance, we will use the second-order differential equation for AdamW described by Equation (4) and choose a timescale of $p = 1$ (we verified the accuracy of this setup in Appendix K). For an arbitrary scale invariant weight W (e.g. a key/query matrix under qk-norm), the continuous-time dynamics of W are governed by

$$\frac{1}{2}\eta\ddot{W}(t) + \dot{W}(t) + \lambda W(t) = -u_W(t) \tag{7}$$

to second-order, with u_W defined analogously to u (replacing g with g_W in Equation (3)). From here, using Equation (6), we can derive (see Appendix J) the following approximation for the weight norm,

$$||W(t)||^2 \approx ||W(0)||^2 e^{-2\lambda t} + \eta \int_0^t d\tau\, e^{-2\lambda(t-\tau)} ||u_W(\tau)||^2 \tag{8}$$

Interestingly, Equation (8) has an identical form to an exponential moving-average of $||u_W||^2$ in continuous-time (see Appendix J). Further (also see Appendix J), the evolution of the unit direction $\hat{W}$ – which is most relevant to consider, given that W is scale invariant – is governed by

$$\frac{1}{2}\eta\ddot{\hat{W}}(t) + \dot{\hat{W}}(t) + \Lambda\hat{W}(t) = -\frac{1}{||W(t)||}u_W(t) \tag{9}$$

where we define $\Lambda := \frac{1}{2}\eta r ||\dot{\hat{W}}||^2$. Equation (9) has the same form as Equation (7), with weight decay $\lambda = \Lambda$ and updating via $u_W/||W||$. With the aforementioned interpretation of $||W||$ as the square-root of a moving-average of $||u_W||^2$, we can undo the coarse-graining assumption (Appendix J) and view $\hat{W}$ as updating by an adaptive-normalization of u_W without additional momentum. This concept of *meta-adaptive normalization*, i.e. adaptive-normalization of an already adaptively-normalized gradient u_W, can be converted into an explicit optimizer 2-Adam which we now describe.

4.2 The k-Adam optimizer

The k-Adam optimizer captures the concept of meta-adaptivity, applying an adaptive-normalization procedure k times with respect to hyperparameters $(\beta_{1:k}, \gamma_{1:k})$, with update rule

$$k\text{-Adam:} \quad \theta_{n+1} := \theta_n - \eta u_n^{(k)} \quad \text{where } u_n^{(i)} := \frac{\sqrt{1-\beta_i^{n+1}}}{1-\gamma_i^{n+1}} \frac{m_n^{(i)}}{\sqrt{v_n^{(i)}}} \tag{10}$$

$$m_n^{(i)} := \gamma_i m_{n-1}^{(i)} + (1-\gamma_i) u_n^{(i-1)}, \quad v_n^{(i)} := \beta_i v_{n-1}^{(i)} + (1-\beta_i)(u_n^{(i-1)})^{\odot 2}$$

with $u_n^{(0)} := g_n \equiv \nabla_\theta L(\theta_n)$ and $(m_{-1}^{(i)}, v_{-1}^{(i)}) = (0, 0)$ for all $i = 1, \ldots, k$.. We present an explicit algorithm for implementing k-Adam in Appendix S. A more succinct (but equivalent) description of k-Adam is presented in Appendix R. Note that 1-Adam is equivalent to Adam. A scale-invariant weight trained under Adam is analogous to 2-Adam with $\gamma_2 = 0$.

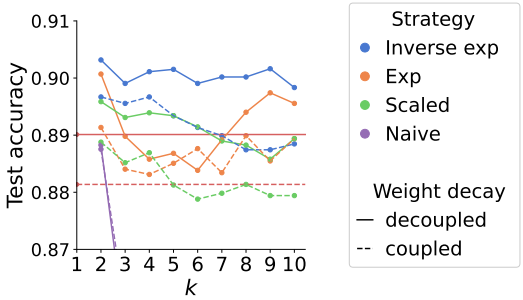


Figure 2: Plot of the best test accuracy against k after training a CNN for 100 epochs on CIFAR10 using the k-Adam optimizer. We highlight the $k = 1$ case in red, corresponding to Adam/AdamW.

Hyperparameter choice. We will consider 4 strategies for choosing hyperparameters $(\beta_{1:k}, \gamma_{1:k})$: **(1)** Inverse exp: $\beta_i := 1 - (1-\tilde{\beta})^{1/k}$; **(2)** Exp: $\beta_i := \tilde{\beta}^k$; **(3)** Scaled: $\beta_i := \tilde{\beta}/k$; **(4)** Naive: $\beta_i := \tilde{\beta}$. In Appendix Q we discuss hyper-parameter choice and motivate the inverse exp strategy; the other strategies have been chosen heuristically.

Evaluating k-Adam. We motivated interpreting the implicit effect of scale invariance based on its beneficial influence on training [9, 10, 11]. We would therefore expect 2-Adam, and perhaps k-Adam for $k > 2$, to outperform Adam; is this indeed the case? We train a CNN (architecture details in Appendix H) on the CIFAR10 dataset for 100 epochs using k-Adam, for values $k = 1, \ldots, 10$ and using each hyperparameter strategy defined above. We display the results in Figure 2, finding that k-Adam outperforms Adam/AdamW at various $k > 1$ under the first three strategies, however the naive strategy performs particularly badly for $k > 2$; we show this in more detail in Appendix Q. We find that 2-Adam with the inverse exp strategy and decoupled weight decay performs best. We discuss related future directions in Appendix B.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 1, 9

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 1, 9

[3] T. Tieleman and G. Hinton. Lecture 6.5—rmsprop, coursera: Neural networks for machine learning. Coursera Lecture Slides, 2012. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. 1, 9

[4] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. 1, 8

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 4

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 4

[7] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters, 2023. 4, 12

[8] Andrea Schioppa Justin Gilmer and Jeremy Cohen. Intriguing properties of transformer training instabilities, 2023. 4, 12

[9] Hidenori Tanaka and Daniel Kunin. Noether's learning dynamics: Role of symmetry breaking in neural networks, 2021. 4, 5, 8

[10] Ekdeep Singh Lubana, Robert P. Dick, and Hidenori Tanaka. Beyond batchnorm: Towards a unified understanding of normalization in deep learning, 2021. 4, 5, 8

[11] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?, 2019. 4, 5, 8

[12] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization, 2019. 8

[13] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes, 2019. 8

[14] Dongruo Zhou, Jinghui Chen, Yuan Cao, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization, 2024. 8

[15] Anas Barakat and Pascal Bianchi. Convergence and dynamical behavior of the adam algorithm for non-convex stochastic optimization, 2020. 8

[16] André Belotto da Silva and Maxime Gazeau. A general system of differential equations to model first order adaptive algorithms, 2019. 8

[17] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel L. K. Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics, 2021. 8

[18] Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow, 2023. 8

[19] Kaifeng Lyu, Zhiyuan Li, and Sanjeev Arora. Understanding the generalization benefit of normalization layers: Sharpness reduction, 2023. 8

[20] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019. 8

[21] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020. 8

[22] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023. 8

[23] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids rank collapse for randomly initialised deep networks, 2020. 8

[24] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth, 2023. 8

[25] Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse, 2022. 8

[26] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr, and Thomas Hofmann. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization, 2018. 8

[27] Ryo Karakida, Shotaro Akaho, and Shun ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks, 2019. 8

[28] Andrej Karpathy. nanogpt. `https://github.com/karpathy/nanoGPT`, 2022. 13

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 13

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 13

A Related work

Previous work has studied adaptive optimization theoretically [12, 13, 14, 15, 16], but this work focuses on asymptotic convergence rates and hence does not cover the results presented here. These works perform a discrete-time analysis, except for [16] which uses the same differential equation representation of Adam to the one we consider in Section 2, and [15] which considers a different differential equation for Adam, yet these works do not explicitly solve for Adam's moving-averages $m(t)$, $v(t)$ or update $u(t)$ in continuous-time as we do in this paper.

There has been previous work on interpreting the effect of scale invariance on training dynamics from a theoretical perspective [9, 17, 18, 19, 10]. Most relevant to our work is [9], which – using a Lagrangian-based approach – shows that scale-invariant weights trained under gradient descent with momentum exhibits an adaptive effect analogous to RMSprop. Since this work does not consider an adaptive learning rate, the theoretical analysis is greatly simplified, however the results are less faithful to practical machine learning where optimizers with adaptive learning rates (i.e. Adam/AdamW) are widely used; our analysis in Section 4 extends this argument to the case of Adam/AdamW, where we observe a meta-adaptive effect. We further use our theoretical insights to motivate a novel optimizer, k-Adam, which captures the implicit role of scale invariance.

Various extensions to Adam have been proposed – such as the AMSGrad [20], LAMB [21], and Lion [22] optimizers – however they differ from our proposed optimizer k-Adam: AMSGrad extends Adam by scaling the learning rate by an adaptive factor $1/\sqrt{\hat{v}_n}$ where $\hat{v}_n$ is a maximum of all past moving-averages v_n; LAMB uses layer-wise adaptive learning rates and a trust ratio mechanism; Lion [22] uses the *sign* of a moving-average of the gradient in order to update the parameters. k-Adam extends Adam uniquely, instead applying an adaptive normalization procedure k times in succession, and is motivated by theoretical insight.

B Discussion

In this work we have presented a continuous-time framework for Adam, facilitating a mathematically tractable analysis of training dynamics. We have demonstrated the utilty of our framework for answering questions of practical interest regarding hyperparameter choice and the role of normalization.

We have seen that the derived bound on the max-update (Equation (5)) is surprisingly accurate to empirical training dynamics, however our bound does not rigorously justify (only suggests) why we observe a faster rate of generalization for larger $C(\beta, \gamma)$ along a given normal curve, hence there is still progress to be made for a complete understanding of adaptive hyperparameter choice.

In this paper we do not perform a rigorous analysis of k-Adam's performance – i.e. evaluating its performance across various datasets and architectures – as our focus in this paper is a theory-driven analysis of training dynamics. Though we find $k = 2$ to perform best in this paper, it is possible that a larger k may be beneficial in certain contexts, e.g. at small batch sizes (to possibly alleviate noisy gradients). There may also be a more natural, theoretically-motivated hyperparameter strategy than the strategies we consider here. We leave a study of these aspects to future work.

We note that the meta-adaptive effect we describe in this paper is not a complete account of the benefits of normalization layers. Normalization layers possess additional benefits, such as avoiding rank collapse [23, 24, 25] and contributing towards the smoothness of the loss landscape [11, 26, 27, 19]; it is unclear whether such phenomena are related to the meta-adaptive effect we describe in this paper.

C Descriptions of optimizers

Momentum [4] features an adaptive gradient direction – updating by a moving-average of the gradient – as described by the update rule,

$$\text{Momentum:} \quad \theta_{n+1} := \theta_n - \eta m_n,$$

$$m_n := \gamma m_{n-1} + (1 - \gamma) g_n$$

for $n = 0, 1, 2, \ldots$, with moving-average hyperparameter γ and $m_{-1} := 0$.

RMSprop [3] features an adaptive learning rate – normalizing the update gradient g_n by a moving-average of the squared gradient $g_n^{\odot 2}$ – as described by the update rule,

$$\text{RMSprop:} \quad \theta_{n+1} := \theta_n - \eta \frac{g_n}{\sqrt{v_n}},$$

$$v_n := \beta v_{n-1} + (1-\beta) g_n^{\odot 2}$$

with $v_{-1} := 0$. Note that $g_n^{\odot 2}$ describes an element-wise squaring, and the division $g_n/\sqrt{v_n}$ is also element-wise. Here we have neglected weight decay; we will discuss weight decay strategies (coupled vs decoupled) below.

Adam [1] and its variant AdamW [2] are a combination of momentum and RMSprop, and also use a bias correction factor, which we justify in Appendix D. Neglecting weight decay, Adam & AdamW are equivalent, with update rule

$$\text{Adam/AdamW:} \quad \theta_{n+1} := \theta_n - \eta \frac{\tilde{m}_n}{\sqrt{\tilde{v}_n}},$$

$$\tilde{m}_n := \frac{1}{1-\gamma^{n+1}} m_n, \quad \tilde{v}_n := \frac{1}{1-\beta^{n+1}} v_n,$$

$$m_n := \gamma m_{n-1} + (1-\gamma) g_n, \quad v_n := \beta v_{n-1} + (1-\beta) g_n^{\odot 2}$$

where a tilde denotes bias correction. The difference between Adam and AdamW comes from how they apply weight decay: Adam uses *coupled* weight decay, equivalent to transforming the loss $L(\theta) \mapsto L(\theta) + \frac{\lambda}{2}||\theta||^2$ such that $g_n \mapsto g_n + \lambda\theta_n$, and AdamW uses *decoupled* weight decay, which involves subtracting $\lambda\eta\theta_n$ from the RHS of the update rule, i.e.

$$\text{AdamW:} \quad \theta_{n+1} := \theta_n - \lambda\eta\theta_n - \eta \frac{\tilde{m}_n}{\sqrt{\tilde{v}_n}}$$

We note that the PyTorch implementation of RMSprop uses coupled weight decay, i.e. RMSprop is Adam with $\gamma = 0$.

D Motivating bias correction in Adam

Consider an exponential moving average of a sequence $\{x_0, x_1, x_2, \ldots\}$ of tensors,

$$\begin{aligned} y_n &:= \beta y_{n-1} + (1-\beta) x_n \\ &= \cdots \\ &= (1-\beta)(x_n + \beta x_{n-1} + \cdots + \beta^n x_0) \end{aligned}$$

In the case that $\mathbb{E}[x_n]$ is independent of n,

$$\mathbb{E}[y_n] = \mathbb{E}[x_n](1-\beta)(1+\beta+\cdots+\beta^n) = \mathbb{E}[x_n](1-\beta^{n+1})$$

hence for an unbiased exponential moving average, we should instead consider a *bias-corrected* form of y_n:

$$\tilde{y}_n := \frac{1}{1-\beta^{n+1}} y_n$$

such that

$$\mathbb{E}[\tilde{y}_n] = \mathbb{E}[x_n]$$

E Deriving the parameter update bound

First we will derive a continuous-time expression for m_n by Taylor expanding its definition,

$$\begin{aligned} m(t_n) &= \gamma m(t_n - \eta^p) + (1-\gamma) g(t_n) \\ &= \gamma m(t_n) - \eta^p \gamma \dot{m}(t_n) + (1-\gamma) g(t_n) + O(\eta^{2p}) \end{aligned}$$

and so to leading order,

$$\dot{m}(t) + \frac{1-\gamma}{\eta^p \gamma} m(t) = \frac{1-\gamma}{\eta^p \gamma} g(t)$$

$$\frac{d}{dt}\left[\exp\left(\frac{1-\gamma}{\eta^p\gamma}t\right) m(t)\right] = \frac{1-\gamma}{\eta^p\gamma}\exp\left(\frac{1-\gamma}{\eta^p\gamma}t\right) g(t)$$

$$\begin{aligned}\implies m(t) &= \frac{1-\gamma}{\eta^p\gamma}\int_0^t d\tau\ \exp\left(-\frac{1-\gamma}{\eta^p\gamma}(t-\tau)\right) g(\tau) \\ &\equiv \int_0^t d\tau\ K_\gamma(\tau,t) g(\tau)\end{aligned}$$

Similarly for v_n,

$$\dot{v}(t) + \frac{1-\beta}{\eta^p \beta} v(t) = \frac{1-\beta}{\eta^p \beta} g(t)^{\odot 2}$$

$$\begin{aligned}\implies v(t) &= \frac{1-\beta}{\eta^p\beta}\int_0^t d\tau\ \exp\left(-\frac{1-\beta}{\eta^p\beta}(t-\tau)\right) g(\tau)^{\odot 2} \\ &\equiv \int_0^t d\tau\ K_\beta(\tau,t) g(\tau)^{\odot 2}\end{aligned}$$

As a result, the ratio of moving-averages present in Adam's update $u(t)$ takes the form,

$$\frac{m(t)}{\sqrt{v(t)}} = \eta^{-p/2}\frac{1-\gamma}{\gamma}\sqrt{\frac{\beta}{1-\beta}}\frac{\int_0^t d\tau\ \exp\left(-\frac{1-\gamma}{\eta^p\gamma}(t-\tau)\right) g(\tau)}{\sqrt{\int_0^t d\tau\ \exp\left(-\frac{1-\beta}{\eta^p\beta}(t-\tau)\right) g(\tau)^{\odot 2}}} \tag{11}$$

Note that this describes an element-wise division of tensors. This expression is particularly amenable to the Cauchy-Schwarz inequality, which says that for (square-integrable) functions $f, g : \mathbb{R} \to \mathbb{R}$,

$$\left|\int dx\ f(x)g(x)\right| \leq \sqrt{\int dx\ f(x)^2 \int dx\ g(x)^2}$$

We can apply the Cauchy-Schwarz inequality (element-wise) to the integral in the numerator of Equation (11) to provide an upper bound,

$$\begin{aligned}&\left|\int_0^t d\tau\ \exp\left(-\frac{1-\gamma}{\eta^p\gamma}(t-\tau)\right) g(\tau)\right| \\ &\qquad\equiv \left|\int_0^t d\tau\ \left[\exp\left(-\frac{C(t-\tau)}{2\eta^p}\right)\right]\left[\exp\left(-\frac{1-\beta}{2\eta^p\beta}(t-\tau)\right) g(\tau)\right]\right| \\ &\qquad\leq \sqrt{\int_0^t d\tau\ \exp\left(-\frac{C(t-\tau)}{\eta^p}\right)}\sqrt{\int_0^t d\tau\ \exp\left(-\frac{1-\beta}{\eta^p\beta}(t-\tau)\right) g(\tau)^{\odot 2}} \\ &\qquad= \sqrt{\frac{\eta^p}{C}(1-\exp(-Ct/\eta^p))}\sqrt{\int_0^t d\tau\ \exp\left(-\frac{1-\beta}{\eta^p\beta}(t-\tau)\right) g(\tau)^{\odot 2}}\end{aligned}$$

where we have defined

$$C := \frac{2\beta(1-\gamma) - \gamma(1-\beta)}{\beta\gamma}$$

Note that the second factor is exactly equal to the denominator in the expression for $m(t)/\sqrt{v(t)}$, and so they cancel to give the following bound:

$$\begin{aligned}\left|\frac{m(t)}{\sqrt{v(t)}}\right| &\leq \frac{1-\gamma}{\gamma}\sqrt{\frac{\beta}{1-\beta}}\sqrt{\frac{1-\exp(-Ct/\eta^p)}{C}} \\ &\leq \frac{1-\gamma}{\gamma}\sqrt{\frac{\beta}{1-\beta}}\begin{cases}1/\sqrt{C}, & C>0, \\ \sqrt{t/\eta^p}, & C=0, \\ \sqrt{\exp(|C|t/\eta^p)/|C|}, & C<0\end{cases}\end{aligned}$$

We can then obtain an expression for $u_n = u(t_n)$ using $n = t_n/\eta^p$.

Weight decay. Note that extending Equation (3) to include weight decay is simple: if we want coupled weight decay (as in Adam), we can transform $g(\tau) \mapsto g(\tau) + k\theta(\tau)$; if we want decoupled weight decay (as in AdamW), we can subtract $k\eta\theta_n$ from the RHS of Equation (1) and proceed identically.

Differential equation form. We can obtain a differential equation for Adam/AdamW by noting that,

$$\theta_{n+1} - \theta_n \equiv \theta(t_n + \eta^p) - \theta(t_n) = \eta^p\dot{\theta}(t_n) + \frac{\eta^{2p}}{2}\ddot{\theta}(t_n) + O(\eta^{3p})$$

by Taylor expansion, which we can then substitute into the update rule $\theta_{n+1} = \theta_n - \eta u_n$.

Normal curves. The level curve $\mathcal{B}_c$ is determined by

$$f(\beta,\gamma) = c, \qquad f(\beta,\gamma) := \frac{2\beta(1-\gamma) - \gamma(1-\beta)}{\beta\gamma} \equiv C(\beta,\gamma)$$

and normal curves $\gamma = \gamma(\beta)$ satisfy

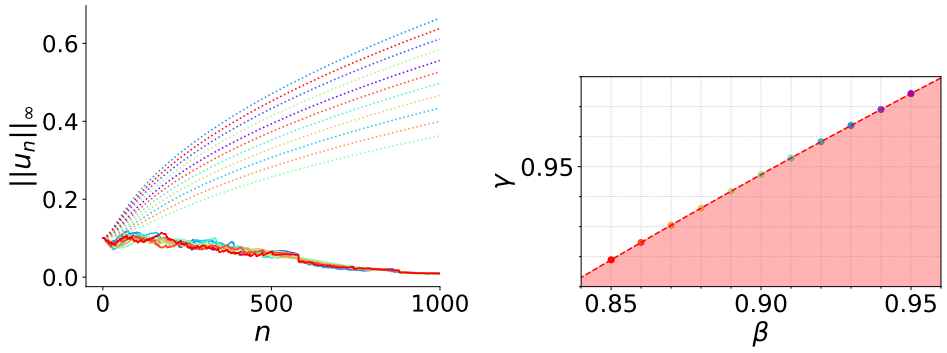
$$\frac{d\gamma(\beta)}{d\beta} = \frac{\partial f/\partial\gamma}{\partial f/\partial\beta}$$

$$\implies \gamma(\beta) = (k - 2\beta^3)^{1/3}$$

for an integration constant k. The normal curve passing through (β_0, γ_0) has integration constant $k = 2\beta_0^3 + \gamma_0^3$.

F Empirical analysis of $\mathcal{B}_0$

Equation (5) suggests that $||u_n||_\infty$ may scale like $\sqrt{n}$ when $(\beta,\gamma) \in \mathcal{B}_0$. Do we observe this in practice? In Figure 3 we instead see bounded behaviour analogous to that of the region $\mathcal{B}_+$ (as seen in Figure 1), i.e. there is bounded growth rather than a $\sqrt{n}$ scaling in the max-update. We note that the bound Equation (5) is not violated; the predicted upper bound is indeed met, though the prediction is just not *tight* relative to the true dynamics empirically. We suspect that the condition $C(\beta,\gamma) = 0$ may be sensitive to numerical precision issues in practice, which is perhaps the reason why we observe behaviour analogous to $C(\beta,\gamma) > 0$ instead.

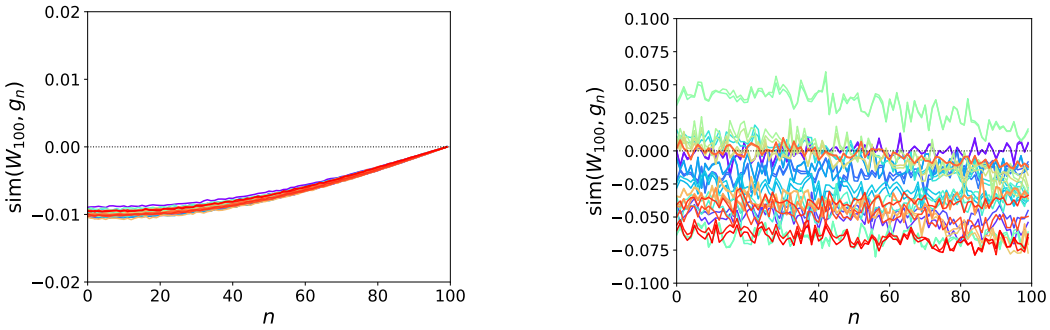


(a) (b)

Figure 3: We plot the max-update over training for 11 points along the boundary $\mathcal{B}_0$. The theoretically predicted $\sqrt{n}$-like scaling is denoted by dotted lines. We use the same experimental setup as used to produce Figure 1, which we describe in Appendix H.

G Assumptions of Section 4.1

Assumption 1. To verify Assumption 1 we consider the transformer model setup described in Appendix H with qk-norm [7, 8] enabled, such that the loss is scale invariant under each query and key matrix. In Figure 4 we plot the cosine similarity $\text{sim}(W_{100}, g_n) := \frac{\langle W_{100}, g_n \rangle}{||W_{100}||||g_n||} \in [-1, 1]$ for iterations $n = 1, \ldots, 100$, for each query and key matrix W in the model (a total of 32 matrices). We observe a cosine similarity of ≈ 0 at $n = 100$ (supporting Equation (6)) and an increase in cosine similarity as n decreases, however the cosine similarity still remains negligible. We also demonstrate that without qk-norm, the cosine similarity is no longer well behaved.



(a) With qk-norm (b) Without qk-norm

Figure 4: Plot of the cosine similarity between each query/key matrix at iteration $m = 100$, with the gradient at all previous iterations n (x-axis). In (a) we enable qk-norm, whereas in (b) the model does not include qk-norm.

Assumption 2. To verify Assumption 2 we choose 16 random parameter values from random query/key matrices W, and plot their trajectory in regular training vs. their trajectory when using coarse-graining on W. We display the results in Figure 5, finding a strong agreement.

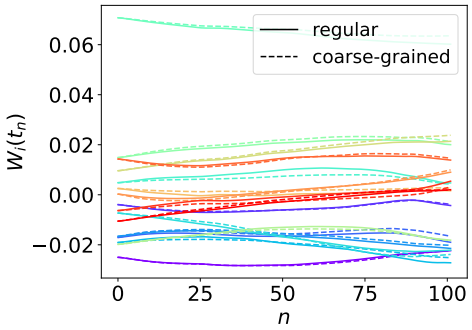


Figure 5: Plot of the trajectories, with and without coarse-graining, for 16 randomly selected parameters from query/key matrices.

A consequence of these assumptions is that $\langle W(t), m_W(t)\rangle \approx 0$ (from Assumption 1 and Equation (2)), and hence (in combination with Assumption 2) we have $\langle W(t), u_W(t)\rangle \approx 0$. We use this in Section 4.1 in order to obtain an adaptive-normalization interpretation of training dynamics. Note that though the cosine similarity is larger at earlier iterations in Figure 4, the kernel K_γ present in m_W places a lower weight on earlier iterations such that a larger cosine similarity maintains a negligible effect.

H Experimental setups

Architecture details. Throughout the paper we consider a nanoGPT model [28] – which is a decoder-only transformer model [29] – and a CNN model [30]. The nanoGPT model architecture we use has 6 layers, with 6 attention heads per layer. The embedding dimension is 384. We use a dropout of 0.2. When training we use the shakespeare dataset associated with the nanoGPT repository [28], using character-level tokenization, for a max input length of 256 tokens. The CNN model architecture we use is shown in Figure 6 which we use for Figure 2.

Continuous-time trajectory. For experiments that require continuous-time trajectories, we use the method described in Appendix I.

??. We describe the method for plotting normal curves in Appendix E.

Figure 1. We train the nanoGPT model using Adam at a learning rate of 10^{-3} (and batch size 64) for a range of adaptive hyperparameters (β, γ) for 1500 iterations, saving the max-update $||u_n||_\infty$ at every iteration n. We compute the slope of $\log ||u_n||_\infty$ at $n = 500$ using the values at $n = 495$ and $n = 505$.

Figure 9. We use the same setup as Figure 1, and additionally use a learning rate warmup (linear, for the first 100 iterations) and after warmup, a cosine decay down to 10^{-4}. We train for 3000 iterations in order for training to converge such that we can assess generalization performance. We evaluate on a test split of the shakespeare dataset (on 200 iterations worth of test data) every 100 iterations to obtain the test losses.

Figure 2. We train a CNN (using architecture in Figure 6) on CIFAR10 for 100 epochs and for each $k \in \{1, \ldots, 10\}$, we sweep over learning rates $\eta \in \{$6e-5, 1e-4, 3e-4, 6e-4, 1e-3, 3e-3$\}$ and weight decays $\lambda \in \{$1e-6, 1e-5, 1e-4, 1e-3, 1e-2$\}$ and plot the best test accuracy across this sweep. We use a linear warmup for the first 2 epochs, and a cosine decay to $\eta/10$ for the remaining epochs. We evaluate on the test split every 5 epochs. We use $(\tilde{\beta}, \tilde{\gamma}) = (0.999, 0.9)$ and a batch size of 512.

I Numerically solving Equation (4)

Numerically solving for continuous dynamics. Defining $\psi(t) := \dot{\theta}(t)$, we can write Equation (4) as two coupled first-order DEs,

$$\dot{\theta}(t) = \psi(t)$$

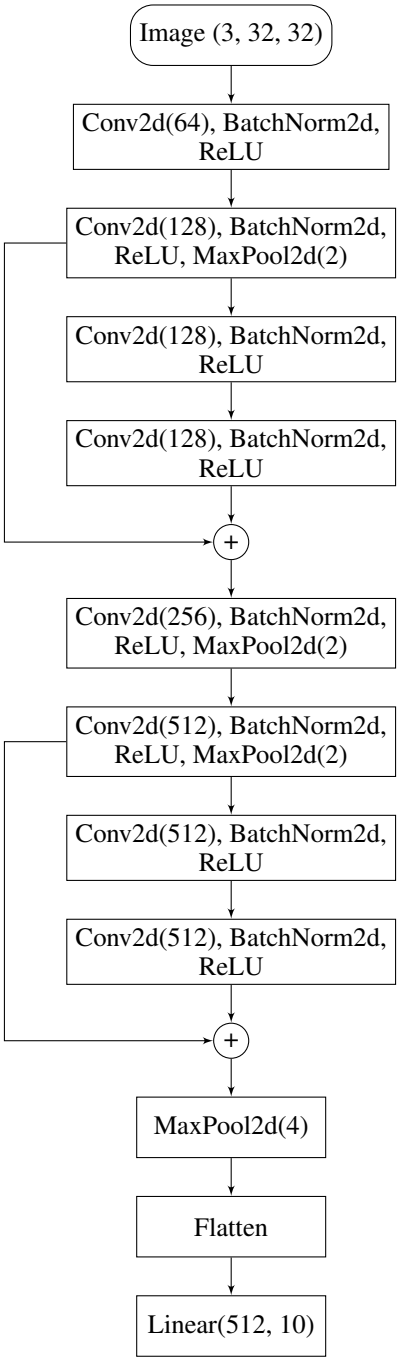


Figure 6: CNN model architecture used in Figure 2. All convolutional layers use a kernel size of 3×3 and a padding of 1.

$$\dot{\psi}(t) = -\frac{2}{\eta^{2p}}\left(\lambda\eta\theta(t) + \eta^p\psi(t) + \eta\frac{\sqrt{1-\beta^{1+t/\eta^p}}}{1-\gamma^{1+t/\eta^p}}\frac{m(t)}{\sqrt{v(t)}}\right)$$

and numerically solve via Euler's method,

$$\theta((n+1)\Delta t) \approx \theta(n\Delta t) + \Delta t\psi(n\Delta t)$$
$$\text{with} \quad \psi(n\Delta t) \approx \psi((n-1)\Delta t) + \Delta t\dot{\psi}((n-1)\Delta t)$$

defining $\Delta t := \eta^p/K$. In our numerical validation experiments, we use the timescale $p = 1$ and $K = 100$.

To compute $\dot{\psi}(n\Delta t)$ via the above DE we must compute

$$m(t) = \int_0^t d\tau\ M(\tau,t)g(\tau) = \int_0^{t-\Delta t} d\tau\ M(\tau,t)g(\tau) + \int_{t-\Delta t}^t d\tau\ M(\tau,t)g(\tau)$$
$$= \exp\left(-\frac{1-\gamma}{\eta^p\gamma}\Delta t\right)\left[\underbrace{\int_0^{t-\Delta t} d\tau\ M(\tau,t-\Delta t)g(\tau)}_{\text{from } n-1\text{th update step}} + \underbrace{\int_{t-\Delta t}^t d\tau\ M(\tau,t-\Delta t)g(\tau)}_{\approx\Delta t\frac{1-\gamma}{\eta^p\gamma}g(t-\Delta t)}\right]$$

Empirically validating $\varphi(t)$. We note that we can write Equation (12) approximately as

$$\varphi(n\Delta t) \approx \varphi_0 e^{-2\lambda n\Delta t} + \Delta t\sum_{m=0}^{n-1} e^{-2\lambda(n-m)\Delta t} f(m\Delta t)$$

where

$$f(t) := \eta||u_W(t)||^2 - 2\langle W(t), u_W(t)\rangle$$

We then compare this expression to the true value of $||W(n\Delta t)||^2$ to validate Equation (12) in Appendix K.

J Implicit meta-adaptive effect

Deriving an expression for $||W(t)||^2$. Applying the inner product $\langle W(t), \cdot\rangle$ to both sides of Equation (7), we arrive at the differential equation

$$\frac{1}{2}\eta\ddot{\varphi} + \dot{\varphi} + 2\lambda\varphi \approx \eta||\dot{W}||^2 - 2\langle W, u_W\rangle$$

where we have defined the squared norm $\varphi(t) := ||W(t)||^2$. One can show (see further below) that to leading order, $||\dot{W}||^2 \approx \|u_W\|^2$ and $\eta\ddot{\varphi} = O(\eta^2) + O(\lambda\eta)$, hence we neglect the $\ddot{\varphi}$ term, with the resulting first-order differential equation,

$$\dot{\varphi} + 2\lambda\varphi \approx \eta||u_W||^2 - 2\langle W, u_W\rangle$$

which when integrated, gives

$$||W(t)||^2 \approx ||W(0)||^2 e^{-2\lambda t} + \int_0^t d\tau\ e^{-2\lambda(t-\tau)}(\eta||u_W(\tau)||^2 - 2\langle W(\tau), u_W(\tau)\rangle) \tag{12}$$

We empirically verify this expression in Appendix K.

A further approximation. In order to derive Equation (12) we did not need to use the property of scale invariance. We will now make use of this property, in combination with the following two simplifying assumptions:

Assumption 1. *We will assume that*

$$\langle W(t), g_W(\tau)\rangle \approx 0 \;\; \forall\, \tau \leq t$$

Assumption 2. *We will use a coarse-graining approximation for $v_W(t)$ (using Equation (2)):*

$$v_W(t) \equiv \int_0^t d\tau\; K_\beta(\tau,t) g_W(\tau)^{\odot 2} \approx \int_0^t d\tau\; \tilde{K}_\beta(\tau,t)||g_W(\tau)||^2 =: \tilde{v}_W(t)$$

with $\tilde{K}_\beta(\tau,t) := K_\beta(\tau,t)/N$ where N is the total number of entries of W, i.e. we replace the entries of $g_W(\tau)^{\odot 2}$ with the average entry across the tensor.

Assumption 1 is intuitive given the properties of high-dimensional spaces[1], but also Equation (6) is suggestive of such a result. We provide empirical support for this assumption in Appendix G. We also show (in Appendix G) that replacing v_W with the coarse-grained version $\tilde{v}_W$ has little effect on training dynamics. A consequence of these assumptions is that $\langle W(t), u_W(t)\rangle \approx 0$, allowing us to approximate,

$$||W(t)||^2 \approx ||W(0)||^2 e^{-2\lambda t} + \eta \int_0^t d\tau\; e^{-2\lambda(t-\tau)}||u_W(\tau)||^2$$

$||W(t)||^2$ as a moving average. Restating Equation (8),

$$||W(t)||^2 \approx ||W(0)||^2 e^{-2\lambda t} + \eta \int_0^t d\tau\; e^{-2\lambda(t-\tau)}||u_W(\tau)||^2$$

which takes an analogous form to a moving-average of $||u_W||^2$ in continuous-time. Specifically, consider a moving-average of $(x_0, x_1, x_2, \ldots)$,

$$w_n = (1-\alpha)w_{n-1} + \alpha x_n$$

Then in continuous time, with $w_n = w(t_n)$ and $x_n = x(t_n)$, we have shown in Appendix E that

$$w(t) = w(0)\exp\left(-\frac{1-\alpha}{\eta^p\alpha}t\right) + \frac{1-\alpha}{\eta^p\alpha}\int_0^t d\tau\; \exp\left(-\frac{1-\alpha}{\eta^p\alpha}(t-\tau)\right)x(\tau)$$

Comparing this expression to Equation (8) allows us to view $||W||^2$ as a moving-average of $||u_W||^2$.

Weight direction dynamics. To obtain Equation (9) from Equation (7), we write $W = r\hat{W}$ with $r := ||W||$ (and $||\hat{W}|| = 1$), which after substituting into Equation (7) becomes

$$(\frac{1}{2}\eta\ddot{r} + \dot{r} + \lambda r)\hat{W} + (\eta\dot{r} + r)\dot{\hat{W}} + \frac{1}{2}\eta r\ddot{\hat{W}} = -u_W \tag{13}$$

Note that taking the derivative of $\langle \hat{W}, \hat{W}\rangle = 1$ implies $\langle \dot{\hat{W}}, \hat{W}\rangle = 0$, and further $\langle \ddot{\hat{W}}, \hat{W}\rangle = -||\dot{\hat{W}}||^2$. As a result, applying $\langle \cdot, \hat{W}\rangle$ to Equation (13) gives

$$\frac{1}{2}\eta\ddot{r} + \dot{r} + \lambda r = \frac{1}{2}\eta r||\dot{\hat{W}}||^2$$

Substituting this expression back into Equation (13) and neglecting higher-order terms (noting that from above, $\eta\dot{r} = O(\eta^2)$), we arrive at

$$\frac{1}{2}\eta\ddot{\hat{W}} + \dot{\hat{W}} + \Lambda\hat{W} = -\frac{1}{r}u_W$$

where we define $\Lambda := \frac{1}{2}\eta r||\dot{\hat{W}}||^2$.

[1]For example the Johnson–Lindenstrauss lemma implies that in high-dimensional spaces, randomly chosen vectors are likely to be almost orthogonal.

Leading order approximations. We can write

$$\dot{W} = -\lambda W - \frac{1}{2}\eta \ddot{W} - u_W$$

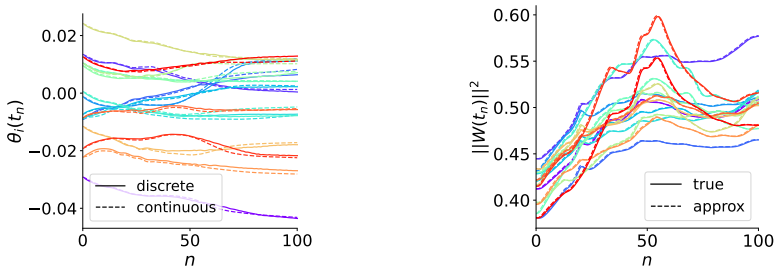
Squaring both sides and using $\langle W, u_W \rangle \approx 0$ (a consequence of Assumption 1 and Assumption 2) results in

$$||\dot{W}||^2 = ||u_W||^2 + \lambda\eta \langle W, \ddot{W} \rangle + \eta \langle \ddot{W}, u_W \rangle + \frac{\eta^2}{4}||\ddot{W}||^2 + \lambda^2 ||W||^2$$

hence to leading order in (η, λ), $||\dot{W}||^2 \approx ||u_W||^2$.

K Empirical validation of continuous dynamics

In Figure 7 we empirically verify the differential equation form for AdamW (Equation (4)) and the prediction for $||W(t)||^2$ (Equation (12)). We use the method described in Appendix I – with $K = 100$, $p = 1$ and $\eta = 10^{-3}$ – to numerically solve Equation (4), obtaining a continuous-time trajectory in parameter space. We train a nanoGPT model for 100 iterations using setup in Appendix H, with $(\beta, \gamma) = (0.999, 0.9)$. To produce Figure 7a we randomly select 16 parameters from the model and plot their trajectories. For Figure 7b we randomly select 16 query/key matrices W and plot the trajectory of $||W||^2$ for each.



(a) (b)

Figure 7: (a) Parameter trajectories and (b) norm trajectories over 100 iterations.

L Symmetries of the transformer architecture

Say $\Phi_s : \Theta \to \Theta$ describes some continuous invariance parameterised by some continuous parameter $s \in \mathbb{R}^S$, i.e.

$$L(\Phi_s(\theta)) = L(\theta) \quad \forall\, s \in \mathbb{R}^S$$

with $\Phi_0(\theta) = \theta \;\forall\, \theta \in \Theta$. Differentiating the above with respect to s_i and evaluating at $s = 0$, we obtain

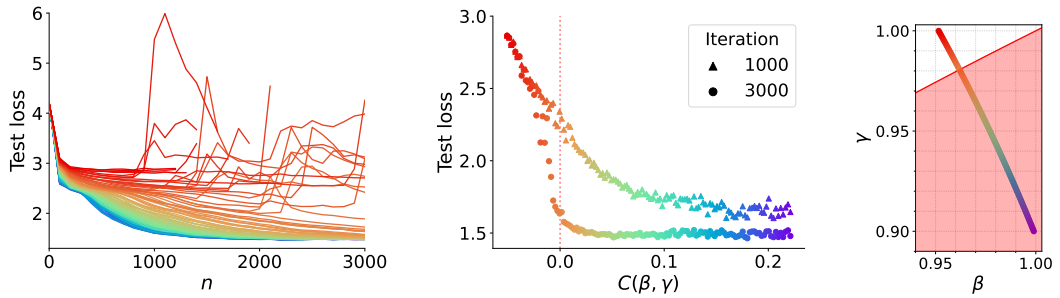
$$\langle \frac{\partial \Phi_s}{\partial s_i}\bigg|_{s=0}, \nabla_\theta L(\theta) \rangle = 0 \qquad \forall\, i = 1, \ldots, S \tag{14}$$

Scale invariance. In the special case of scale invariance with respect to a weight W, if we write $\theta = (W, \phi)$ then the relevant transformation is $\Phi_s(\theta) = (sW, \phi)$, and Equation (14) becomes,

$$\langle W, \nabla_W L(\theta) \rangle = 0$$

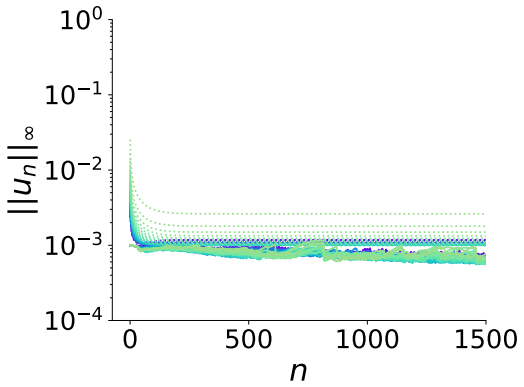
M Max-update trajectory in $\mathcal{B}_+$

To visualize the theoretical bounds in the case of $(\beta, \gamma) \in \mathcal{B}_+$ in more detail, we plot the results of Figure 1a but restricted to $(\beta, \gamma) \in \mathcal{B}_+$ in Figure 8. We indeed see that the theoretically predicted bounds (dotted lines) are approximately satisfied.



(a) (b) (c)

Figure 9: **Our theory accurately predicts the divergence of test loss across Adam's hyperparameter space.** We consider 128 hyperparameter points (β, γ) taken uniformly along the entire normal curve $\mathcal{C}_{\tilde{\beta},\tilde{\gamma}}$, visualized in (c). For each point, we plot (a) the test loss over training iterations n, and (b) the best test loss achieved over the first 1000 and 3000 iterations. The rightmost point in (c) corresponds to $(\tilde{\beta}, \tilde{\gamma}) = (0.999, 0.9)$.



(a)

Figure 8: We plot the results of Figure 1a but restrict to (β, γ) that lie in $\mathcal{B}_+$.

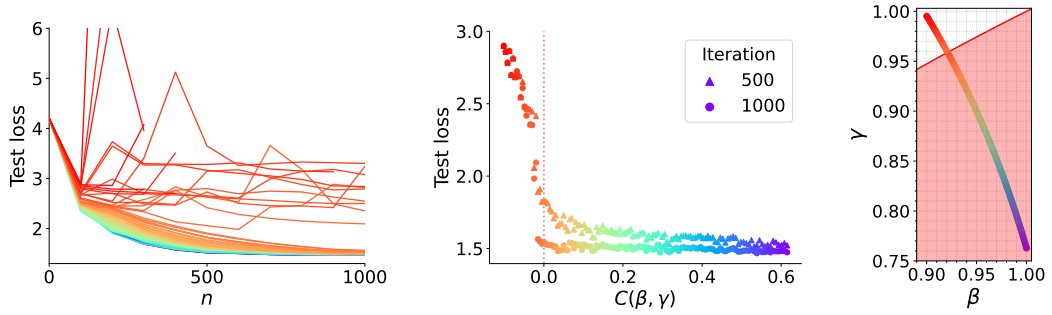
N Analysis of generalization perforamnce

We will now assess how generalization performance varies in $\mathcal{B}_+$, and particularly, how the value of $C(\beta, \gamma)$ correlates with generalization performance. In Figure 9 we observe how test loss varies along the normal curve $\mathcal{C}_{\tilde{\beta},\tilde{\gamma}}$. taking 128 points uniformly along the entire curve (visualized in Figure 9c). We see that larger values of $C(\beta, \gamma)$ display faster generalization, as seen in Figure 9a and highlighted at iteration 1000 in Figure 9b. After sufficient training, i.e. at iteration 3000 in Figure 9b, all points in $\mathcal{B}_+$ achieve roughly the same test loss. The test loss exhibits a rapid increase after entering the region $\mathcal{B}_-$, which is to be expected given the unstable exponential growth in the max-update shown in Section 3.2.

We provide supporting evidence for this claim, finding similar results for the normal curve that instead passes through the hyperparameter values $(0.95, 0.9)$, in Appendix O.

O Supporting data for Section 3.2

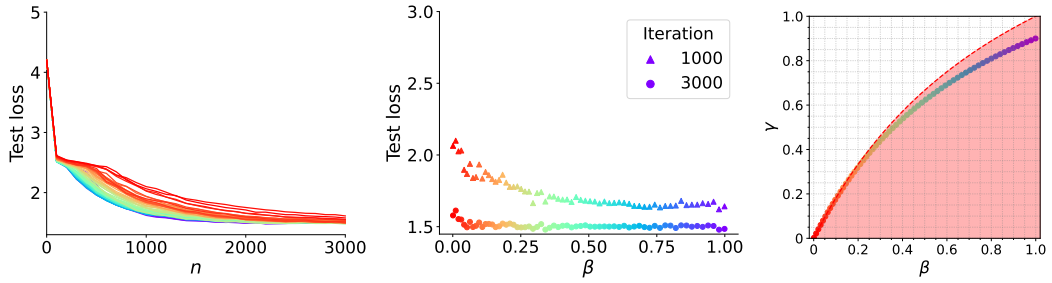
In Section 3.2 we found that a larger value of $C(\beta, \gamma)$ correlated with a faster generalization. We further verify this in Figure 10 by instead taking points along the normal curve through $(0.95, 0.9)$, but otherwise an identical experimental setup to the one used to produce Figure 9.



(a) (b) (c)

Figure 10: Reproduced Figure 9 but instead for the normal curve through the point $(0.95, 0.9)$.

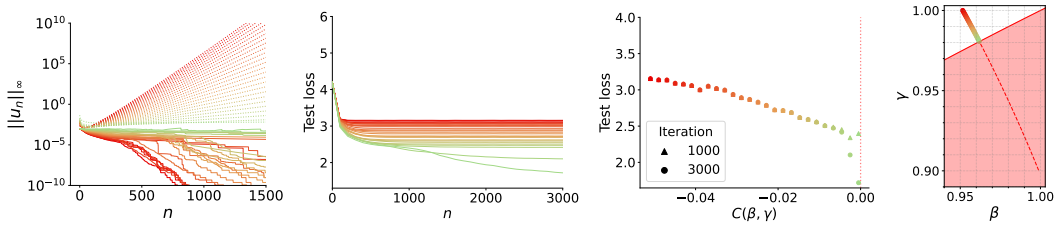
We also consider the level curve $\mathcal{B}_{C(\tilde{\beta},\tilde{\gamma})}$, the results of which we plot in Figure 11. We see a slower rate of generalization for points closer to the origin, as expected, given that such points will have a smaller value of $C(\beta, \gamma)$ relative to other points along their normal curve.



(a) (b) (c)

Figure 11: Reproduced Figure 9 but instead for the level curve through the point $(0.999, 0.9)$.

P Attempting to correct exponential growth with learning rate annealing



(a) (b) (c) (d)

Figure 12: Reproduced Figure 1 and Figure 9 restricted to the region $\mathcal{B}_-$ and using learning rate annealing described below.

Since the parameter update is $\theta_{n+1} - \theta_n \equiv -\eta u_n$, it appears that a well chosen step-dependent learning rate η_n may correct the exponential growth in the case of $C(\beta, \gamma) < 0$ described by Equation (5), which occurs via the factor $\exp(n|C(\beta, \gamma)|/2)$. Particularly, given the tightness of the bound seen in Figure 1b, we could consider the learning rate

$$\eta_n = \eta_0 \exp(-n|C(\beta, \gamma)|/2) \tag{15}$$

at step n, which results in a bound on $||\theta_{n+1} - \theta_n||_\infty \equiv \eta_n ||u_n||_\infty$ that is equivalent to the case of $C(\beta, \gamma) > 0$ with learning rate η_0. However when we run this learning rate annealing strategy (Equation (15)) in practice, we find that it is too strong, as shown in Figure 12; the max-update becomes very small, and we do not observe good generalization performance. Hence, even though this annealment strategy for $\mathcal{B}_-$ results in the regions $\mathcal{B}_+$ and $\mathcal{B}_-$ possessing the same bounds on $||\theta_{n+1} - \theta_n||_\infty$, their training properties are still quite different. Analysis of weaker annealing methods are out of scope for this work. It is unclear whether correcting such exponential growth in $\mathcal{B}_-$ by choosing an appropriate annealing method would provide any benefits to simply choosing $(\beta, \gamma) \in \mathcal{B}_+$.

Q Choice of k-Adam hyperparameters

We note that max-update bounds as shown for Adam (Equation (5)) hold analogously for each intermediate update $u_n^{(i)}$, which we describe in Appendix T. For the strategies defined above, $(\beta_i, \gamma_i) \in \mathcal{B}_+$ for all $i = 1, \ldots, k$ (since $\beta_i > \gamma_i$) hence we expect stable training as a result of this bound.

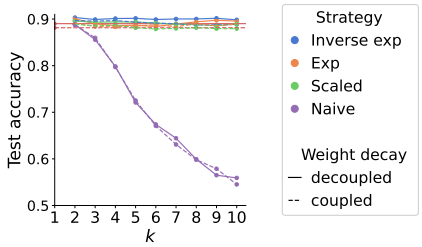


Figure 13: Zoomed out version of Figure 2 to show performance of the naive strategy.

Inverse-exp strategy. The moving-average $m_n^{(k)}$ will have a prefactor of $(1-\beta_1)\cdots(1-\beta_k)$, and we can consider a uniform strategy $\beta_1 = \cdots = \beta_k$ and enforce that this prefactor is equal to the typical prefactor for Adam/AdamW, i.e.

$$(1-\beta_i)^k = 1 - \tilde{\beta}$$
$$\implies \beta_i = 1 - (1-\tilde{\beta})^{1/k}$$

which we can do similarly for γ_i.

Naive strategy. In Figure 13 we show a zoomed-out version of Figure 2 to show the performance of the naive strategy across all k, emphasizing the importance of well-selected k-Adam hyperparameters.

We also found the strategy $\beta_i = \tilde{\beta}^{1/k}$, $\gamma_i = \tilde{\gamma}^{1/k}$ to perform badly.

R Adaptive normalization and k-Adam

For a sequence of tensors $x_{0:n} \equiv (x_0, \ldots, x_n)$, we define the bias-corrected moving-average operator M_γ,

$$M_\gamma(x_{0:n}) := \frac{1-\gamma}{1-\gamma^{n+1}}(x_n + \gamma x_{n-1} + \cdots + \gamma^n x_0)$$

We justify bias correction in Appendix D. We then define the *adaptive-normalization* operator $A_{\gamma,\beta}$,

$$A_{\gamma,\beta}(x_{0:n}) := \frac{M_\gamma(x_{0:n})}{\sqrt{M_\beta(x_{0:n}^{\odot 2})}}$$

Adaptive optimization. We can describe Adam/AdamW with hyperparameters (β, γ) simply as

$$\text{Adam/AdamW:} \quad \theta_{n+1} = \theta_n - \eta A_{\gamma,\beta}(g_{0:n})$$

i.e. updating by the adaptive-normalization of gradients $g_{0:n}$, where $g_i \equiv \nabla_\theta L(\theta_i)$.

k-Adam extends Adam/AdamW by applying the adaptive normalization operator k-times with hyperparameters $(\beta_{1:k}, \gamma_{1:k})$,

$$k\text{-Adam:} \quad \theta_{n+1} = \theta_n - \eta(A_{\gamma_k,\beta_k} \circ \cdots \circ A_{\gamma_1,\beta_1})(g_{0:n})$$

which is equivalent to Equation (10). Specifically, the intermediate updates take the form $u_n^{(i)} \equiv (A_{\gamma_i,\beta_i} \circ \cdots \circ A_{\gamma_1,\beta_1})(g_{0:n})$ (meaning that $u_n^{(i+1)}$ is an adaptive-normalization of $u_n^{(i)}$).

Continuous-time properties. We move into continuous-time, with $x_n \equiv x(t_n)$ where $t_n = n\eta^p$. Appendix E has shown that

$$M_\gamma(x_{0:n}) = \int_0^{t_n} d\tau \; K_\gamma(\tau, t) x(\tau), \qquad K_\gamma(\tau, t) := \frac{1-\gamma}{\eta^p \gamma} \exp\left(-\frac{1-\gamma}{\eta^p \gamma}(t-\tau)\right)$$

to first-order. Then using Cauchy-Schwarz on $A_{\gamma,\beta}$, Appendix E has also shown that

$$||A_{\gamma,\beta}(x_{0:n})||_\infty \leq \frac{\sqrt{1-\beta^{n+1}}}{1-\gamma^{n+1}} \frac{1-\gamma}{\gamma} \sqrt{\frac{\beta}{1-\beta}} B_n(\beta, \gamma) \tag{16}$$

where B_n is defined as in Section 3.1. Note that this bound is in general independent of the elements $x_{0:n}$. Since each k-Adam sub-update $u_n^{(i)} = A_{\gamma_i,\beta_i}((A_{\gamma_{i-1},\beta_{i-1}} \circ \cdots \circ A_{\gamma_1,\beta_1})(g_{0:n}))$, this bound holds analogously for $u_n^{(i)}$.

S Algorithm for k-Adam

We provide an explicit algorithm for the k-Adam optimizer below.

Algorithm 1 k-Adam update rule

given learning rate $\eta \in \mathbb{R}$, weight decay $\lambda \in \mathbb{R}$, *coupled* $\in \{\text{False}, \text{True}\}$, hyperparameters $(\beta_{1:k}, \gamma_{1:k})$, epsilon ϵ (10^{-30} by default)
initialize step count $n \leftarrow 0$, initial parameter $\theta_0 \in \mathbb{R}^p$, $(m_i, v_i) \leftarrow (0, 0)$ for $i = 1, \ldots, k$
$g \leftarrow \nabla_\theta L(\theta_n)$
if *coupled* **then**
  $g \leftarrow g + \lambda\theta_n$
end if
$\hat{g} \leftarrow g$
for $i = 1, \ldots, k$ **do**
  $m_i \leftarrow \gamma_i m_i + (1-\gamma_i)\hat{g}$
  $v_i \leftarrow \beta_i v_i + (1-\beta_i)\hat{g}^{\odot 2}$
  $C \leftarrow \sqrt{1-\beta_i^{n+1}}/(1-\gamma_i^{n+1})$
  $\hat{g} \leftarrow C m_i/(\sqrt{v_i} + \epsilon)$
end for
$u \leftarrow \hat{g}$
if not *coupled* **then**
  $u \leftarrow u + \lambda\theta_n$
end if
$\theta_{n+1} \leftarrow \theta_n - \eta u$
$n \leftarrow n + 1$
return parameter trajectory $(\theta_0, \theta_1, \ldots)$

T k-Adam update bounds

Theoretical guarantees in the form of bounds – as shown for Adam/AdamW in Section 3 – hold analogously for k-Adam. Specifically, since $u_n^{(i)}$ is an adaptive-normalization of $u_{0:n}^{(i-1)}$ for each

$i = 1, \ldots, k$, we have the bound

$$||u_n^{(i)}||_\infty \leq \frac{\sqrt{1-\beta_i^{n+1}}}{1-\gamma_i^{n+1}} \frac{1-\gamma_i}{\gamma_i} \sqrt{\frac{\beta_i}{1-\beta_i}} B_n(\beta_i, \gamma_i)$$

with B_n defined as in Equation (5). As a result, we can view k-Adam (for $k > 1$) as possessing stronger stability guarantees than Adam – e.g. k-Adam updates by $u_n^{(k)}$, an adaptive-normalization of a *bounded* quantity $u_{0:n}^{(k-1)}$ due to the above, whereas Adam updates by an adaptive-normalization of $g_{0:n}$, however the gradients $g_{0:n}$ have no such bound guarantees.