Understanding Optimization in Deep Learning with Central Flows

Jeremy Cohen*
Carnegie Mellon and Flatiron Institute
jcohen@flatironinstitute.org

Alex Damian*
Princeton University
ad27@princeton.edu

Ameet Talwalkar
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University

Jason D. Lee
Princeton University

Abstract

Traditional theories of optimization cannot describe the dynamics of optimization in deep learning, even in the simple setting of deterministic training. The challenge is that optimizers typically operate in a complex, oscillatory regime called the *edge of stability*. In this paper, we develop theory that can describe the dynamics of optimization in this regime. Our key insight is that while the *exact* trajectory of an oscillatory optimizer may be challenging to analyze, the time-averaged (i.e. smoothed) trajectory is often much more tractable. To analyze an optimizer, we derive a differential equation called a *central flow* that characterizes this time-averaged trajectory. We empirically show that these central flows can predict long-term optimization trajectories for generic neural networks with a high degree of numerical accuracy. By interpreting these central flows, we are able to understand how gradient descent makes progress even as the loss sometimes goes up; how adaptive optimizers "adapt" to the local loss landscape; and how adaptive optimizers implicitly navigate towards regions where they can take larger steps. Our results suggest that central flows can be a valuable theoretical tool for reasoning about optimization in deep learning.

1 Introduction

While there is a rich body of work on the theory of optimization, few works attempt to analyze optimization in "real" deep learning settings. Instead, even works motivated by deep learning often rely on unrealistic assumptions such as convexity, or restrict their analyses to simplified models. Practitioners cannot use such theories to reason directly about their optimization problems. Our goal in this paper is to develop optimization theory that applies *directly* to deep learning problems. This is a difficult task: prior research has shown that, even in the seemingly simple setting of deterministic (i.e. full-batch) training, optimization typically operates in a complex, oscillatory regime called the *edge of stability* (EOS) (Xing et al., 2018; Wu et al., 2018; Jastrzębski et al., 2019, 2020; Cohen et al., 2021, 2022). The dynamics of optimization in this regime cannot be captured by traditional optimization theory.

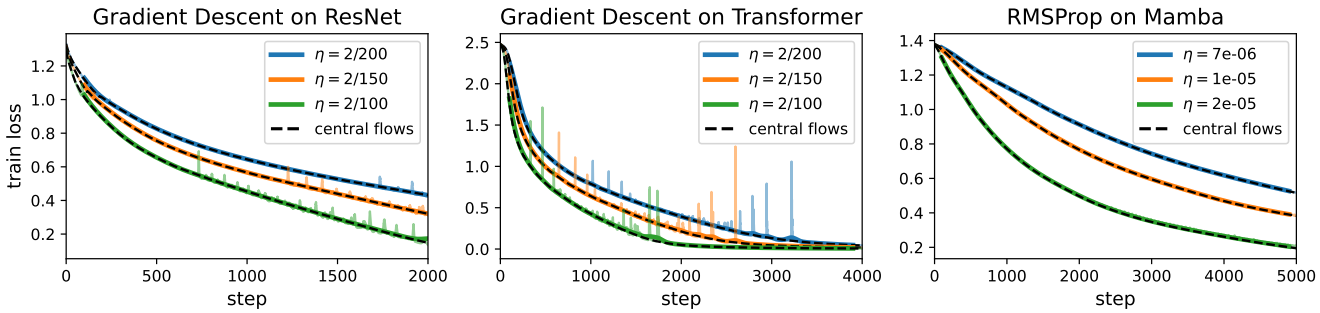


Figure 1: **Our theory accurately predicts long-term optimization trajectories of practical neural networks**. We hold our theory to the high standard of rendering accurate *numerical* predictions about the optimization of practical (i.e. non-toy) neural networks. For example, this figure shows that our central flows can accurately predict the time-averaged (smoothed) loss curves of gradient descent and RMSProp on various practical architectures.

*Equal contribution. Author ordering determined by coin flip over a Zoom call (Kingma and Ba, 2015).

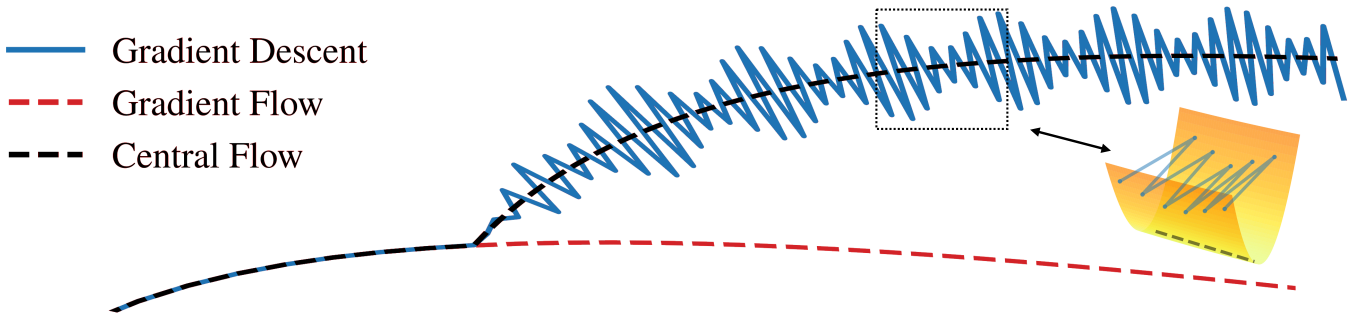


Figure 2: **The central flow models the time-averaged (smoothed) trajectory of the oscillatory optimizer**. In this representative cartoon, gradient descent (blue) takes an oscillatory path through weight space. The central flow (black) is a smooth curve that characterizes this trajectory, whereas gradient flow (red) takes a different path. As illustrated in the inset, an oscillatory optimizer can be visualized as moving through a "valley" while bouncing between the "valley walls" (Xing et al., 2018; Cohen et al., 2021; Wen et al., 2025).

In this paper, we devise a methodology for analyzing these oscillatory deep learning dynamics. Our key insight is that while the *fine-grained* trajectory of an oscillatory optimizer may be challenging to analyze, the *time-averaged* (i.e. locally smoothed) trajectory is often much more tractable. To analyze an optimization algorithm, we derive a differential equation called a *central flow* which explicitly captures this time-averaged trajectory (Figure 2). Being a smooth curve, the central flow is a simpler object than the original oscillatory trajectory. Hence, by interpreting the central flow, we can reason more easily about the original optimizer.

We start in Section 3 by analyzing gradient descent, the simplest optimizer. We first explain why traditional analyses cannot capture the typical dynamics of gradient descent in deep learning, and we then present a new analysis that does capture these dynamics. The product of this analysis is a central flow. We use this central flow to understand various aspects of gradient descent's behavior, such as how the train loss can behave non-monotonically over the short-term while nevertheless decreasing over the long term. We then examine a simple adaptive optimizer in Section 4, before turning to RMSProp (i.e. Adam without momentum) in Section 5. We show that much of the behavior of these optimizers is actually *implicit* in their oscillatory dynamics, and we render such behaviors *explicit* via our central flow analysis. In particular, our central flows reveal how these adaptive optimizers: (1) implicitly adapt their step size(s) to the local curvature, and (2) implicitly steer towards lower-curvature regions where they can take larger steps.

We focus in this paper on the simple, idealized setting of deterministic (i.e. full-batch) training. However, we emphasize that similar optimization dynamics have been observed in the practical stochastic setting (Jastrzębski et al., 2019, 2020; Andreyev and Beneventano, 2024). We view our analysis of deterministic optimization as a necessary stepping stone to a subsequent analysis of stochastic optimization.

While we derive each central flow using informal mathematical reasoning, we show that these flows can accurately predict long-term optimization trajectories in a variety of deep learning settings — a high standard of empirical proof. Thus, we believe that central flows hold promise as a framework for analyzing, reasoning about, and perhaps even inventing, deep learning optimizers.

Our code can be found at: `http://github.com/centralflows/centralflows`.

2 Related Work

Edge of stability The dynamics of optimization in deep learning remain poorly understood, even in the seemingly simple setting of deterministic (i.e. full-batch) training. Indeed, recent research showed that gradient descent on neural networks typically operates in a regime termed the "edge of stability" (EOS) in which (1) the largest Hessian eigenvalue equillibrates around the *critical threshold* $2/\eta$, and (2) the algorithm oscillates along high-curvature directions without diverging (Xing et al., 2018; Wu et al., 2018; Jastrzębski et al., 2019, 2020; Cohen et al., 2021). These dynamics could not be explained by existing optimization theory, which led Cohen et al. (2021) to observe that there was no explanation for how or why gradient descent can function properly in deep learning.

Subsequently, several studies sought to theoretically explain EOS dynamics. Some works rigorously analyzed EOS dynamics on specific objective functions (Agarwala et al., 2023; Ahn et al., 2024; Chen and Bruna, 2023; Even et al., 2024; Kreisler et al., 2023; Song and Yun, 2023; Li et al., 2022b; Wu et al., 2024; Zhu et al., 2023), while other works (Arora et al., 2022; Lyu et al., 2022; Damian et al., 2023) gave generic analyses based on a local *third-order* Taylor expansion of the loss, which is one order higher than is normally used in the theoretical analysis of gradient descent. Similar arguments were first used by Blanc et al. (2019) to study implicit regularization in SGD. Our work is most directly inspired by Damian et al. (2023), as their analysis applies to generic objective functions, and holds throughout training, not just near convergence. However, whereas they analyze the *fine-grained* oscillatory dynamics, we argue that analyzing the *time-averaged* dynamics is simpler, and is sufficient for most purposes.

Continuous-time models for optimization The standard continuous-time model for gradient descent is the gradient flow. Barrett and Dherin (2021); Smith et al. (2021) argued that gradient descent is, instead, better approximated by a *modified* gradient flow that is augmented with a penalty on the squared gradient norm. We find that on deep learning objectives, this modified flow improves slightly over gradient flow in the stable regime, but fails in the edge of stability regime, where most of the discrepancy between gradient descent and gradient flow originates (see Appendix C.1). Rosca et al. (2023) proposed a flow that can model oscillations by using complex numbers. However, this flow still cannot track the long-term trajectory of gradient descent in EOS regime.[1]

Many works propose to model the dynamics of stochastic optimizers using stochastic differential equations (SDEs) (Li et al., 2017; Li et al., 2021, Malladi et al., 2022; Compagnoni et al., 2023, 2025). In the full batch limit, where SGD reduces to gradient descent, these SDEs reduce to gradient flow, which is a poor approximation to gradient descent at the edge of stability. Thus, these SDEs cannot be accurate in all hyperparameter regimes. Further, even when these SDEs do well-approximate the real optimizer trajectory, the SDE trajectories are themselves oscillatory, and accordingly can possess behaviors that are *implicit* in the oscillatory dynamics. Our central flows, by contrast, average out the oscillations and render all such behaviors *explicit*. Developing an analogue of the central flow for stochastic optimization is an interesting open question (see Section 7). While some works do aim to explicitly characterize the time-averaged trajectory of SGD (Blanc et al., 2019; Damian et al., 2021; Li et al., 2022a), existing analyses only apply in limiting regimes (e.g. $\eta \to 0$), and only when the loss is already near zero.

Understanding adaptive optimizers Ma et al. (2022) observed that RMSProp and Adam oscillate, and Cohen et al. (2022) showed that such dynamics can be viewed as an adaptive version of the edge of stability, a finding which we will leverage. Khaled et al. (2023) and Mishkin et al. (2024) observed that on quadratic functions, certain adaptive optimizers implicitly adapt their effective step size to the maximum stable step size; we show this holds more generally, beyond quadratics. Experiments in Roulet et al. (2024) and Wang et al. (2024d) are explained by the phenomenon we call "acceleration via regularization." Many works have also conducted rigorous convergence analyses of adaptive optimizers, generally focused on deriving rates of convergence to a global minimizer or stationary point (Duchi et al., 2011; Reddi et al., 2018; Chen et al., 2019a,b; Zaheer et al., 2018; Zou et al., 2019; Défossez et al., 2022; Li and Lin, 2024; Chen et al., 2022; Wang et al., 2024a; Yang et al., 2024; Guo et al., 2021; Shi et al., 2021; Zhang et al., 2022; Crawshaw et al., 2022; Li et al., 2024; Wang et al., 2024b; Hong and Lin, 2024; Zhang et al., 2025; Wang et al., 2024c; Hübler et al., 2024).

[1] Further, whereas their flow "runs through" each iterate of the oscillatory gradient descent trajectory, our central flow instead *averages out* the oscillations to yield a smooth trajectory, which we believe is a more useful mathematical object (see Section 3.3).

3 Gradient Descent

The simplest first-order optimizer is deterministic gradient descent with a fixed learning rate η:

$$w_{t+1} = w_t - \eta \nabla L(w_t). \tag{1}$$

Perhaps surprisingly, Cohen et al. (2021) showed that traditional optimization analyses cannot capture the typical dynamics of gradient descent in deep learning. We now present a new analysis that *does* capture these dynamics.

- In Section 3.1, we describe the typical dynamics of gradient descent in deep learning, and we explain why these oscillatory *edge of stability* dynamics cannot be captured by traditional optimization theory.
- In Section 3.2, we show that while the *exact* oscillatory trajectory may be hard to analyze, the *time-averaged* trajectory is more tractable. We derive a central flow that characterizes this time-averaged trajectory.
- In Section 3.3 we use this central flow to understand the behavior of gradient descent. For example, we show that while gradient descent's loss curve is non-monotonic, it can be viewed as the superposition of the loss along the central flow, plus a contribution from the oscillations. The central flow loss is a smoothly varying quantity that monotonically decreases, and therefore constitutes a hidden progress metric for gradient descent.

Our analysis of gradient descent will set the stage for subsequent analyses of more complex optimizers.

3.1 The Dynamics of Gradient Descent

To understand the oscillatory dynamics of gradient descent in deep learning, it is instructive to first consider the case of quadratic objective functions. On quadratic functions, gradient descent oscillates if the *curvature* (i.e. Hessian) is too large relative to the learning rate. For example, consider a one-dimensional quadratic objective $L(x) = \frac{1}{2}Sx^2$, which has global curvature S. Under gradient descent with learning rate η, the iterates $\{x_t\}$ evolve via $x_{t+1} = (1-\eta S)x_t$. If S exceeds the *critical threshold* $2/\eta$, then $(1-\eta S) < -1$, so the iterate x_t flips signs and grows in magnitude at each step, i.e. gradient descent oscillates with exponentially growing magnitude, as shown in Figure 3. More generally, on a quadratic objective in multiple dimensions, the curvature is quantified by the Hessian matrix, and gradient descent oscillates with exponentially growing magnitude along Hessian eigenvectors with eigenvalues exceeding $2/\eta$.[2]

Of course, deep learning objectives $L(w)$ are not globally quadratic. Still, at any point w in weight space, the objective can be locally approximated by a quadratic Taylor expansion around w. The dynamics of gradient descent on this quadratic are controlled by the largest eigenvalue of the Hessian $H(w)$, which we call the *sharpness* $S(w)$:

$$S(w) := \lambda_1(H(w)). \tag{2}$$

If the sharpness $S(w)$ exceeds $2/\eta$, then gradient descent on the quadratic Taylor approximation would oscillate with exponentially growing magnitude along the top Hessian eigenvector(s). This argument suggests that gradient descent cannot function properly in regions of weight space where the sharpness $S(w)$ exceeds $2/\eta$.

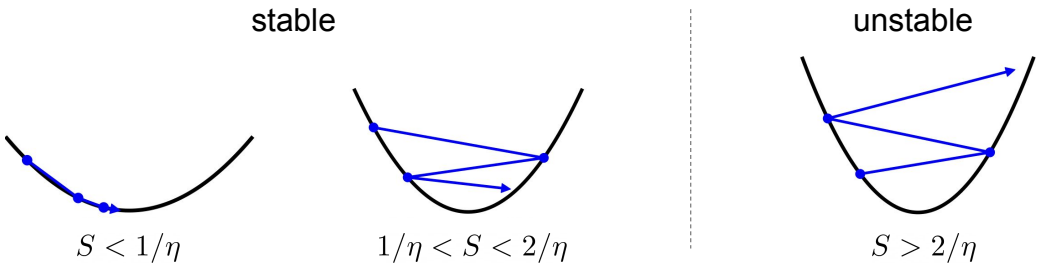


Figure 3: **Gradient descent on a quadratic function.** Consider gradient descent with learning rate η on a quadratic function $\frac{1}{2}Sx^2$, with sharpness S. If $S > 2/\eta$, gradient descent oscillates with exponentially growing magnitude.

[2] An exception is if the initial iterate has exactly zero alignment with these Hessian eigenvectors. However, this event has probability zero under any typical random initialization.

(a) **Expectation:** gradient descent stays throughout training inside the *stable region* (gray), the subset of weight space where the sharpness is bounded by $2/\eta$.

(b) **Reality:** gradient descent frequently exits the stable region, but dynamically steers itself back inside.

Figure 4: **Why does gradient descent converge in deep learning?** The reality (**right**) is dramatically different from the picture suggested by traditional theory (**left**).

In light of this discussion, why does gradient descent converge in deep learning? The natural explanation is that the sharpness remains below $2/\eta$ throughout training. In other words, if we define the "stable region" $\{w : S(w) \leq 2/\eta\}$ as the subset of weight space where the sharpness is bounded by $2/\eta$, then one might suppose that gradient descent remains inside the stable region throughout training, as depicted in the cartoon Figure 4(a). This is the picture suggested by traditional analyses of gradient descent.[3]

Yet, Cohen et al. (2021) observed a very different reality when training neural networks using gradient descent. Figure 5 depicts a typical gradient descent trajectory, with important events annotated (a) - (g). Initially, the sharpness rises (a).[4] Indeed, it is a robust empirical phenomenon, dubbed *progressive sharpening*, that the sharpness tends to rise when training neural networks.[5][6] Soon enough, the sharpness rises past the critical threshold $2/\eta$. Once this happens, gradient descent begins to oscillate with growing magnitude along the top Hessian eigenvector,[7] just as one would predict from a quadratic Taylor approximation (b). These oscillations grow large enough that the train loss starts to go up rather than down (c). Yet, gradient descent does not diverge. Instead, something odd happens: as if "by magic," the sharpness rapidly *drops* (d). Indeed, it drops all the way below the critical threshold $2/\eta$, after which point the oscillations start to shrink in magnitude (e), as one would expect from a new quadratic Taylor approximation. The unexplained rapid drop in the sharpness has conveniently prevented gradient descent from diverging.[8] Similar dynamics recur throughout the rest of training: gradient descent oscillates without diverging along the highest-curvature direction(s),[9] as the sharpness stays dynamically regulated around the critical threshold $2/\eta$ (f). Meanwhile, the train loss decreases over the long run, but behaves non-monotonically over the short run (g).

Intuitively, whereas the traditional theory implies that gradient descent remains inside the stable region throughout training, as in Figure 4(a), in reality gradient descent is frequently exiting the stable region, but is somehow steering itself back inside, as in Figure 4(b). Cohen et al. (2021) dubbed these dynamics *edge of stability* (EOS), and noted that they could not be explained by traditional optimization theory.

[3] We are referring to analyses which assume L-smoothness, i.e. Lipschitzness of the gradient / boundedness of the Hessian spectral norm. This assumption is usually stated as a global condition, but analyses generally only require it to hold locally, in the vicinity of the trajectory.

[4] Throughout this paper, we report the Hessian eigenvalues measured not at the iterates themselves, but rather at the second-order midpoints between the iterates. This results in plots that are slightly cleaner, while retaining all essential features (see Appendix B.1).

[5] Progressive sharpening remains theoretically unexplained. Our goal in this paper is not to understand the origin of progressive sharpening in deep learning, but rather to understand the dynamics of gradient descent on objective functions which may or may not possess this property.

[6] In the literature, progressive sharpening has also been called a "narrowing valley" (Liu et al., 2025a,b) and "lower loss as sharper" loss landscape structure (Li et al., 2023; Bai et al., 2025).

[7] To compute "displacement along top Hessian eigenvector" for Figure 5, we let t_0 be the first step of the figure (i.e. step 2990), we let u be the top Hessian eigenvector computed at step t_0, and we report $u^T(w_t - w_{t_0})$.

[8] This process is similar to the "catapult" phenomenon observed in Lewkowycz et al. (2020) at initialization. Indeed, the EOS dynamics with one unstable eigenvalue resemble a sequential series of catapults. However, the dynamics with >1 unstable eigenvalues are more complex.

[9] As gradient descent oscillates along the high-curvature directions, it can be visualized as moving through a "valley" while bouncing between the "walls" of the valley (Xing et al., 2018; Cohen et al., 2021; Wen et al., 2025).

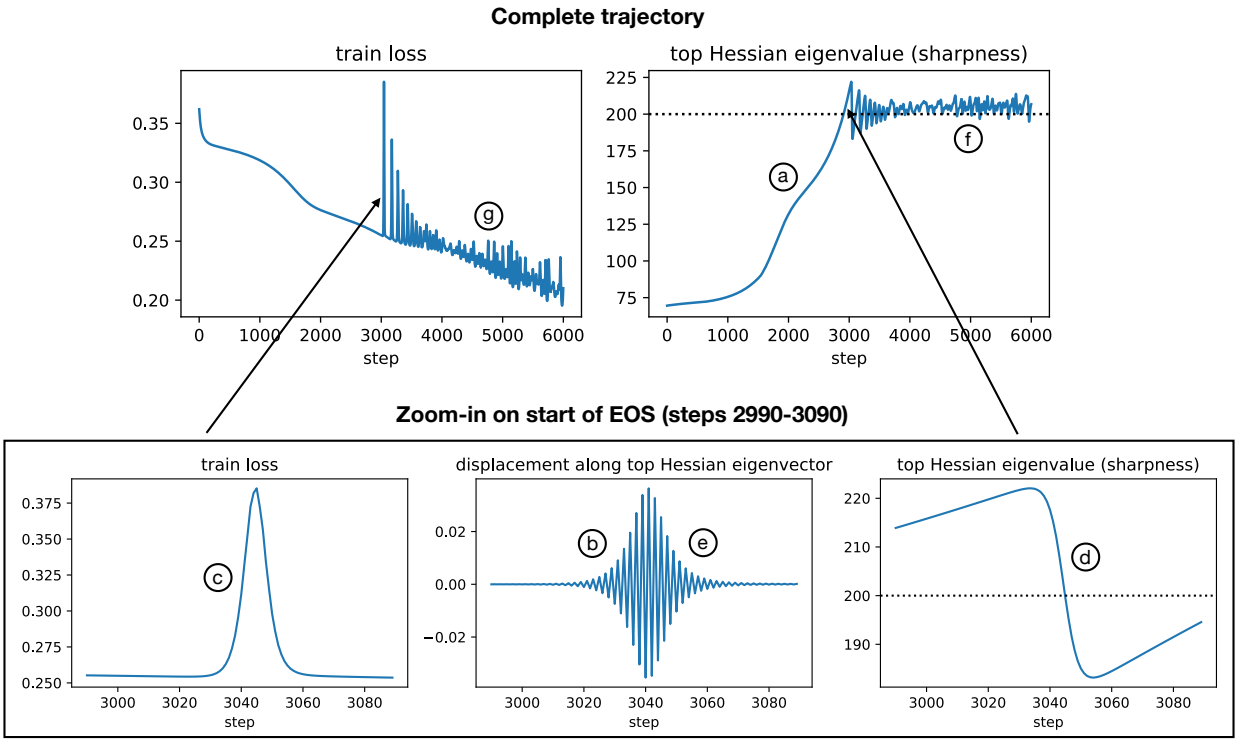
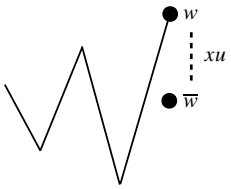


Figure 5: **A typical gradient descent trajectory in deep learning**. We train a neural network using gradient descent with step size $\eta = 0.01$. The top row shows the long-term trajectory, while the bottom row zooms in on a particular time segment. **(a)** The sharpness rises, reaching the critical threshold $2/\eta$ around step 2900. **(b)** Once the sharpness crosses the critical threshold $2/\eta$, gradient descent oscillates with growing magnitude along the top Hessian eigenvector. **(c)** These oscillations cause the train loss to go up rather than down. **(d)** However, gradient descent does not diverge; instead, as if "by magic", the sharpness decreases until falling below $2/\eta$. **(e)** Once the sharpness is below $2/\eta$, the oscillations shrink. Throughout the rest of training: **(f)** the sharpness stays regulated around the critical threshold $2/\eta$ and **(g)** the train loss behaves non-monotonically over short timescales, while decreasing over long timescales. *Details*: the network is a Vision Transformer trained on a subset of CIFAR-10 using MSE loss.

Damian et al. (2023) showed that the key for understanding these surprising dynamics is to Taylor-expand the objective to *third* order, which is one order higher than traditionally used in analyses of gradient descent. A third-order Taylor expansion reveals the crucial ingredient missing from traditional optimization theory:

Oscillations along the top Hessian eigenvector automatically trigger reduction of the top Hessian eigenvalue.

Let us informally sketch this argument. Suppose that gradient descent is oscillating around a reference point $\overline{w}$, along the top Hessian eigenvector u, with current magnitude x, so that (illustration on right):



$$w = \overline{w} + xu. \tag{3}$$

Due to the oscillation, the optimizer follows the gradient at w rather than the gradient at $\overline{w}$. How do the two relate? A Taylor expansion of ∇L around $\overline{w}$ yields:

$$\nabla L(\overline{w} + xu) = \underbrace{\nabla L(\overline{w})}_{\text{first term}} + \underbrace{xH(\overline{w})u}_{\text{second term}} + \underbrace{}_{=xS(\overline{w})u} + \mathcal{O}(x^2). \tag{4}$$

Since u is an eigenvector of $H(\overline{w})$ with eigenvalue $S(\overline{w})$, we recognize the second term as $xS(\overline{w})u$. This term causes a negative gradient step computed at $\overline{w}+xu$ to move in the $-u$ direction. In other words, this term is causing gradient descent to oscillate back and forth along the top Hessian eigenvector u, as predicted by the traditional theory. The "magic" comes from the *next* term, which arises from third-order terms in the Taylor expansion of the loss:

$$\nabla L(\overline{w}+xu) = \overbrace{\nabla L(\overline{w})}^{\text{first term}} + \overbrace{xS(\overline{w})u}^{\text{second term}} + \overbrace{\underbrace{\tfrac{1}{2}x^2\nabla_{\overline{w}}[u^T H(\overline{w})u]}_{=\frac{1}{2}x^2\nabla S(\overline{w})}}^{\text{third term}} + \mathcal{O}(x^3). \tag{5}$$

Since $u^T H(\overline{w})u = S(\overline{w})$, we recognize this term as $\frac{1}{2}x^2\nabla S(\overline{w})$, where ∇S is none other than the *gradient of the sharpness*.[10] Thus, a negative gradient step computed at $\overline{w}+xu$ implicitly takes a negative gradient step *on the sharpness* with step size $\frac{1}{2}\eta x^2$. This is the key ingredient missing from the traditional theory. When gradient descent exits the stable region, it oscillates along the top Hessian eigenvector, just as the traditional theory predicts; but what the traditional theory fails to anticipate is that these oscillations in turn perform gradient descent *on the sharpness*, thereby steering the trajectory back into the stable region automatically.

Note that traditional optimization theory fails to capture the basic *causal structure* of the optimization process: gradient descent converges not because the sharpness is "already" small, but rather due to an automatic negative feedback mechanism that *keeps* the sharpness small.

Damian et al. (2023) analyzed the EOS dynamics in the special case where only one Hessian eigenvalue has crossed the critical threshold $2/\eta$, as in steps 2901-3607 in Figure 6. In this setting, the dynamics consist of consecutive cycles in which: (1) the sharpness rises above $2/\eta$; (2) this triggers growing oscillations along the top Hessian eigenvector; (3) such oscillations reduce sharpness via eq. (5), pushing it below $2/\eta$; (4) the oscillations consequently shrink in magnitude.[11] However, a more common situation is when *multiple* Hessian eigenvalues have reached $2/\eta$, as in steps 3607-4086 in Figure 6. In this situation, gradient descent oscillates simultaneously along all the corresponding eigenvectors,[12] and these oscillations cause all such eigenvalues to remain dynamically regulated around $2/\eta$.

Unfortunately, analyzing EOS dynamics in fine-grained detail is challenging (Damian et al., 2023). The difficulty arises from the need to account for the mutual interactions between the oscillations and the curvature. Even in the

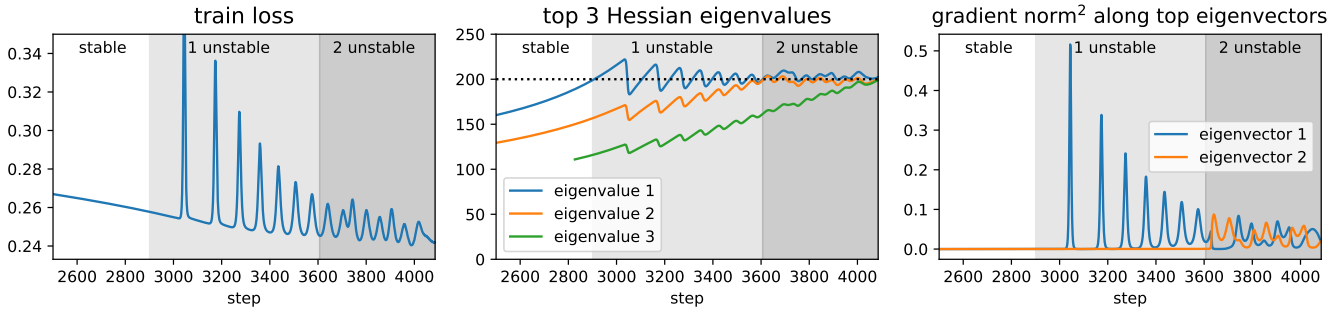


Figure 6: **Multiple Hessian eigenvalues can be at the edge of stability.** From steps 2901-3607, one Hessian eigenvalue is at the edge of stability, and gradient descent oscillates along the top Hessian eigenvector. From steps 3607-4086, two Hessian eigenvalues are at the edge of stability, and gradient descent oscillates simultaneously along both the corresponding eigenvectors. The number of oscillating directions can be easily read off from the right plot, which shows the squared norm of the gradient when projected onto each of the top 3 Hessian eigenvectors.

[10] Technically, equating $\nabla_{\overline{w}}[u^T H(\overline{w})u] = \nabla_{\overline{w}} S(\overline{w})$ requires invoking Danskin's theorem. This is made precise in Appendix A.8, Fact 1.

[11] Notice that the drop in the sharpness is rapid, yielding a sawtooth-like plot for the evolution of the sharpness. This is because the size of the sharpness reduction effect is proportional to x^2. When x is small, the sharpness-reduction effect is negligible, but when x grows larger, the effect quickly becomes strong. See Damian et al. (2023) for a simplified ODE model of the joint dynamics between x and sharpness.

[12] With $k > 1$ unstable eigenvalues, the corresponding eigenvectors are not individually identifiable; instead, one should think of gradient descent as oscillating within the k-dimensional *eigenspace* spanned by the k eigenvectors at the edge of stability.

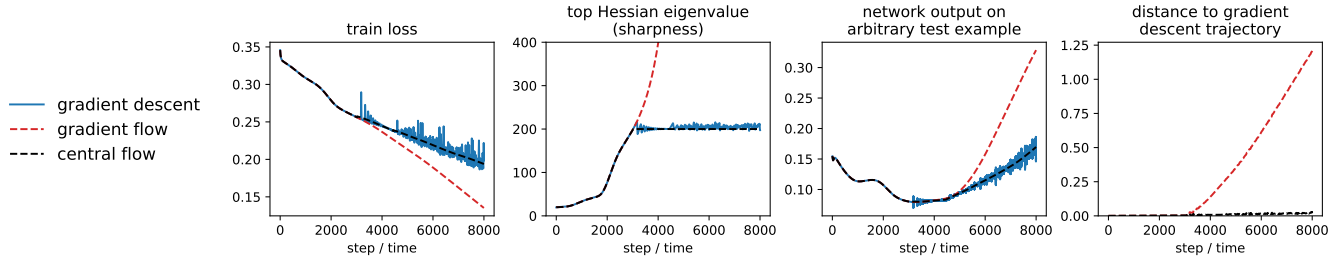
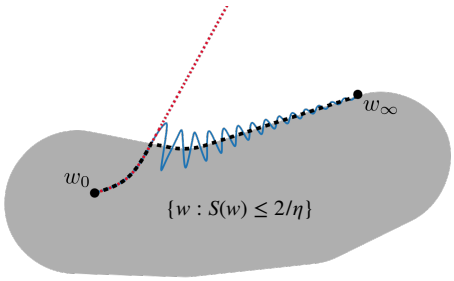


Figure 7: **What macroscopic path does gradient descent take?** Gradient descent (blue) is well-approximated by gradient flow (red) so long as the sharpness is below $2/\eta$. However, once gradient descent reaches the edge of stability, it takes a different path. Our *central flow* (black) approximates gradient descent even at the edge of stability. The plots on top present data from an experiment (ViT / CIFAR-10); the drawing on the right is a cartoon of the underlying weight-space dynamics.



special case of one unstable eigenvalue, these dynamics are nonlinear and highly sensitive to initial conditions. The more typical case of multiple unstable eigenvalues is even harder to analyze: the dynamics with k unstable eigenvalues do not decouple into k independent systems, and instead involve $O(k^2)$ mutually interacting quantities, yielding complex and often chaotic behavior.

Our key insight in this paper is that a fine-grained analysis of the EOS dynamics may not be necessary. Rather, we argue that the more important question is: what *macroscopic* (i.e. long-term) trajectory does gradient descent take through weight space? In the next section, we will use a heuristic time-averaging argument to characterize this macroscopic trajectory. Our analysis will not only recover the main finding of Damian et al. (2023) for a single unstable eigenvalue, but will also readily generalize to the challenging setting of multiple unstable eigenvalues.

3.2 Deriving the Gradient Descent Central Flow

The standard continuous-time approximation to gradient descent is the gradient flow:[13]

$$\frac{dw}{dt} = -\eta \nabla L(w). \tag{6}$$

Cohen et al. (2021) observed that trajectory of gradient descent is well-approximated[14][15] by that of gradient flow so long as training is *stable*, i.e. so long as the sharpness $S(w)$ remains below $2/\eta$. However, once the sharpness reaches $2/\eta$ and the dynamics enter the EOS regime, gradient descent departs from the gradient flow trajectory and takes a different path, as illustrated in Figure 7.[16]

[13] We fold η into the definition of gradient flow so that there is a correspondence between step t of gradient descent and time t of gradient flow. This will especially be useful when analyzing adaptive optimizers where the effective step size is a dynamic quantity.

[14] Barrett and Dherin (2021) argued that the accuracy of the gradient flow approximation can be improved by adding a penalty on the squared gradient norm. However, their modified flow does not hold in the EOS regime, and in the stable regime, we found that the accuracy improvement it brings is relatively small (Appendix C.1). Therefore, for simplicity, we leave out any such term from our flows.

[15] It remains theoretically unexplained why gradient flow is such a good fit to gradient descent. Existing bounds for the distance between gradient descent and gradient flow increase exponentially with time, with an exponent determined by the most negative Hessian eigenvalue (Elkabetz and Cohen, 2021). Empirically, such bounds are overly conservative.

[16] Even in the simplest setting of one unstable eigenvalue, capturing the EOS dynamics necessarily requires three variables: one for the oscillations along the top Hessian eigenvector, one for the top Hessian eigenvalue (sharpness), and one for the remaining directions. Since visualizing three-dimensional dynamics is difficult, we will frequently resort to two-dimensional cartoons (e.g. Figure 7). Such a "projection" will necessarily drop information. Accordingly, Figure 7 captures sharpness and remaining directions, but leaves out the back-and-forth oscillations along the top Hessian eigenvector. Figure 2, by contrast, captures these back-and-forth oscillations but leaves out the sharpness.

We now derive a more general differential equation, which we call a central flow, that approximates the trajectory of gradient descent even at the edge of stability. The central flow directly models the *time-averaged* (i.e. smoothed) trajectory of the oscillatory optimizer. In other words, the central flow averages out the oscillations while retaining their lasting effect on the macroscopic trajectory. We will derive the central flow using a heuristic time-averaging argument, and we will empirically demonstrate that it can accurately predict long-term optimization trajectories on a variety of neural networks with a high degree of numerical accuracy, as illustrated in Figure 7.

We will abuse notation and use $\mathbb{E}$ to denote "local time-averages" of deterministic quantities — see Appendix A.1.5 for additional discussion. The gradient descent central flow is intended to model the time-averaged trajectory $\mathbb{E}[w_t]$. To simplify notation, we will also use $\overline{w}_t := \mathbb{E}[w_t]$ to denote the time-averaged trajectory.

3.2.1 Warm-up: the Special Case of One Unstable Eigenvalue

We will introduce our time-averaging methodology by analyzing the special case when only the largest Hessian eigenvalue has crossed the critical threshold $2/\eta$, and gradient descent oscillates along a single direction — the corresponding eigenvector. Our fully general analysis, given later in Section 3.2.2, will allow for an arbitrary number of eigenvalues to be at the edge of stability, and for eigenvalues to enter and leave the edge of stability.

Thus, in this section, we start our analysis at the instant when the sharpness $S(w)$ first reaches $2/\eta$. From this point onward, we will model the gradient descent trajectory by:

$$w_t = \overline{w}_t + x_t u_t, \tag{7}$$

where w_t is the gradient descent iterate, $\overline{w}_t$ is the time-averaged iterate, u_t is the top Hessian eigenvector at $\overline{w}_t$, and x_t denotes the displacement between w_t and $\overline{w}_t$ along the u_t direction.[17] Note that by definition, $\mathbb{E}[x_t] = 0$, i.e. the time-averaged displacement is zero. To track the evolution of the time-averaged iterate $\overline{w}_t$, we time-average both sides of the gradient descent update eq. (1):

$$\overline{w}_{t+1} = \overline{w}_t - \eta \underbrace{\mathbb{E}[\nabla L(w_t)]}_{\text{time-averaged gradient}}. \tag{8}$$

That is, the time-averaged iterates follow the (negative) time-averaged gradient. To approximate the time-averaged gradient, we first Taylor-expand the gradient ∇L around the time-averaged iterate $\overline{w}_t$:

$$\nabla L(w_t) = \underbrace{\nabla L(\overline{w}_t)}_{\text{gradient at } \overline{w}_t} + \underbrace{x_t\, S(\overline{w}_t) u_t}_{\text{oscillation}} + \underbrace{\tfrac{1}{2}\, x_t^2\, \nabla S(\overline{w}_t)}_{\text{sharpness reduction}} + \mathcal{O}(x^3). \tag{9}$$

We then take the time average of both sides, averaging over the x oscillations. This reflects an implicit assumption that the x oscillations are happening fast relative to the remaining training dynamics:[18]

$$\mathbb{E}[\nabla L(w_t)] \;\approx\; \nabla L(\overline{w}_t) + \underbrace{\cancel{\mathbb{E}[x_t] S(\overline{w}_t) u_t}}_{\text{0 because } \mathbb{E}[x_t]=0} + \underbrace{\tfrac{1}{2}\, \mathbb{E}[x_t^2] \nabla S(\overline{w}_t)}_{\text{implicit sharpness penalty}}. \tag{10}$$

This calculation shows that the time-averaged gradient $\mathbb{E}[\nabla L(w_t)]$ is equal to the gradient at the time-averaged iterate $\nabla L(\overline{w}_t)$, plus an implicit sharpness penalty whose strength is proportional to $\mathbb{E}[x_t^2]$, the variance of the oscillations at step t. Substituting eq. (10) into eq. (8) and switching to continuous time, we therefore model the time-averaged iterates $\overline{w}_t$ by the sharpness-penalized gradient flow $w(t)$ defined by:

$$\frac{dw}{dt} = -\eta \Big[\nabla L(w) + \underbrace{\tfrac{1}{2}\sigma^2(t) \nabla S(w)}_{\text{implicit sharpness penalty}} \Big]. \tag{11}$$

[17] This is a simplification. In reality, we know that gradient descent is displaced from $\overline{w}$ in at least *two* directions: the u direction and the $\nabla S(\overline{w})$ direction, with the latter responsible for the fluctuations in the sharpness. However, when modeling the time-averaged gradient, we will only account for the displacement in the u direction. This is analogous to Damian et al. (2023, Assumption 5). The success of our experiments validates this simplification.

[18] When time-averaging eq. (9), we assume that the eigenvector u changes slowly relative to the displacement x so that $\mathbb{E}[x_t u_t] \approx \mathbb{E}[x_t] u_t$.

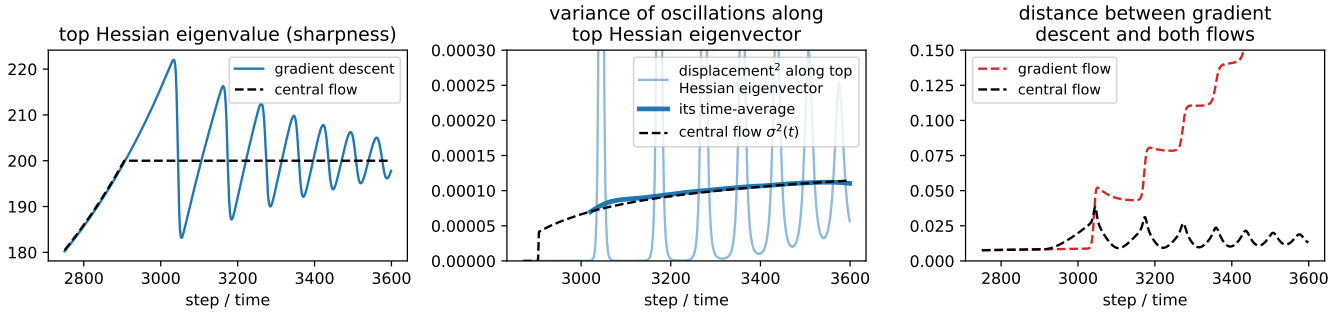


Figure 8: **Illustrating the central flow with one unstable eigenvalue**. **Left**: the sharpness (top Hessian eigenvalue) cycles around $2/\eta$ under gradient descent, and stays locked exactly at $2/\eta$ under the central flow. **Center**: we plot the squared magnitude of the displacement between gradient descent w_t and the central flow $w(t)$ along the top Hessian eigenvector (light blue). Observe that the time average of this quantity (dark blue) is well-predicted by the central flow's $\sigma^2(t)$ (black). **Right**: the Euclidean distance (black) between gradient descent w_t and the central flow $w(t)$ stays small over time, indicating that the central flow accurately predicts the long-term trajectory of gradient descent. In contrast, the distance (red) between gradient descent and the gradient flow eq. (6) grows large over time.

Here, $\sigma^2(t)$ is a still-unknown quantity intended to model $\mathbb{E}[x_t^2]$, the instantaneous variance of the oscillations at time t. This quantity also controls the strength of the implicit sharpness penalty. To determine $\sigma^2(t)$, we argue that only one value is consistent with the observed behavior of gradient descent. Empirically, once the sharpness reaches the critical threshold $2/\eta$, it does not continue to rise indefinitely; rather, it remains dynamically regulated around $2/\eta$. Thus, we will enforce that the central flow never increases the sharpness $S(w(t))$ past $2/\eta$. The time derivative of the sharpness under a flow of the form eq. (11) can be easily computed using the chain rule:

$$\frac{dS(w)}{dt} = \left\langle \nabla S(w), \frac{dw}{dt} \right\rangle = \underbrace{\eta \left\langle \nabla S(w), -\nabla L(w) \right\rangle}_{\substack{\text{change in sharpness} \\ \text{under gradient flow}}} - \underbrace{\tfrac{1}{2}\eta\sigma^2(t)\|\nabla S(w)\|^2}_{\substack{\text{sharpness reduction} \\ \text{from oscillations}}}. \tag{12}$$

When the first term, the change in sharpness under the gradient flow, is *negative*, gradient descent will leave the edge of stability and will once again follow gradient flow — this is made precise in Section 3.2.2. Therefore, we focus on the case where this first term is positive, i.e. where progressive sharpening holds. As the sharpness is currently at $2/\eta$ and must remain at $2/\eta$, we must have that $\frac{dS(w)}{dt} = 0$. Since $\frac{dS(w)}{dt}$ is affine in $\sigma^2(t)$, we can easily solve for the unique value of $\sigma^2(t)$ that ensures $\frac{dS(w)}{dt} = 0$:

$$\sigma^2(t) = \frac{2\left\langle \nabla S(w), -\nabla L(w) \right\rangle}{\|\nabla S(w)\|^2}. \tag{13}$$

Intuitively, this is the unique $\sigma^2(t)$ for which the downward force of oscillation-induced sharpness reduction "cancels out" the upwards force of progressive sharpening so the sharpness remains locked at $2/\eta$. The central flow for a single unstable eigenvalue is given by substituting this $\sigma^2(t)$ into eq. (11):

$$\frac{dw}{dt} = -\eta\Big[\nabla L(w) + \tfrac{1}{2}\sigma^2(t)\nabla S(w)\Big] \quad \text{where} \quad \sigma^2(t) = \frac{2\left\langle \nabla S(w), -\nabla L(w) \right\rangle}{\|\nabla S(w)\|^2}. \tag{14}$$

Figure 8 demonstrates this flow in action. We run gradient flow until the sharpness hits $2/\eta$, and then switch to eq. (14) at that time. (The complete central flow, defined in the next section, will handle such switches automatically.) Observe that the distance in weight space between gradient descent and the central flow remains small over time,[19]

[19]Observant readers might notice that the distance between gradient descent and the central flow starts to grow once the sharpness hits $2/\eta$ and is actually initially larger than the distance between gradient descent and the gradient flow. This is because the central flow has already started to apply sharpness regularization, but the discrete gradient descent trajectory has not yet done so.

verifying that the central flow accurately predicts the long-term trajectory of gradient descent. Moreover, observe that $\sigma^2(t)$ from eq. (13) accurately predicts the empirical variance of the oscillations along the top Hessian eigenvector, further demonstrating that our time-averaging argument is accurately capturing gradient descent's behavior.

Intuitively, whereas gradient descent reduces sharpness in impulse-like spurts which are triggered whenever the oscillations grow large, the central flow applies a sharpness-reduction force continuously, with the same average strength over time. That these two processes stay close over long timescales implies that the oscillations are only affecting the long-term gradient descent trajectory via their *variance* rather than via their fine-grained details (e.g the precise shape of the light blue line in Figure 8, center). This is good news: while the fine-grained oscillations may be challenging to analyze, we have shown that their variance is easy to analyze, as there is only one possible value that is consistent with the observed edge of stability equilibrium. In this way, we have successfully used a heuristic argument to solve for the time-averaged trajectory of gradient descent.

Interpretation as Projection While we have derived the central flow as a sharpness-penalized gradient flow, it can be equivalently interpreted as a *projected* gradient flow. In particular, simplifying eq. (14) gives:

$$\frac{dw}{dt} = -\eta \left[I - \frac{\nabla S(w) \nabla S(w)^\top}{\|\nabla S(w)\|^2} \right] \nabla L(w) = -\eta \Pi^\perp_{\nabla S(w)} \nabla L(w), \quad (15)$$

where $\Pi^\perp_v := I - \frac{vv^T}{\|v\|^2}$ denotes the projection matrix onto the orthogonal complement of v. This flow projects out the ∇S direction from the gradient ∇L to keep the sharpness fixed at $2/\eta$, as illustrated on the right.

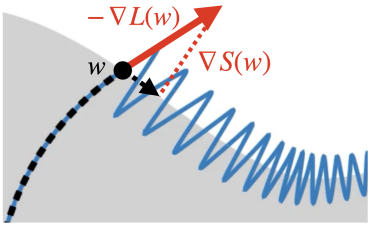


Figure 9: The central flow projects out the $\nabla S(w)$ direction from the loss gradient $\nabla L(w)$.

Previously, Damian et al. (2023) proved that under certain conditions, gradient descent at the edge of stability implicitly follows the trajectory of projected gradient descent constrained to the stable region. We have thus nearly[20] rederived their result in a simpler, albeit non-rigorous, manner.

The projection interpretation will be useful below for reasoning about gradient descent's behavior.

3.2.2 The Fully General Case

We will now derive the complete central flow, which applies in the fully general setting where any number of eigenvalues can be at the edge of stability, including zero. When no eigenvalues are at the edge of stability, the central flow will automatically reduce to the gradient flow. As above, we decompose the gradient descent trajectory as:

$$w_t = \overline{w}_t + \delta_t, \quad (16)$$

where w_t is the gradient descent iterate, $\overline{w}_t := \mathbb{E}[w_t]$ is the time-averaged iterate, and δ_t denotes the displacement between w_t and $\overline{w}_t$, i.e. the oscillations. Because gradient descent oscillates along the Hessian eigenvectors that are at the edge of stability, we model δ_t as lying within the span of these eigenvectors.[21] For example, in the case where only one direction is at the edge of stability, taking $\delta_t = x_t u_t$ recovers the analysis in Section 3.2.1. Note that by definition of $\overline{w}_t$, we have that $\mathbb{E}[\delta_t] = 0$. As before, the time-averaged iterates follow the time-averaged gradient $\mathbb{E}[\nabla L(w_t)]$. To compute the time-averaged gradient, we first Taylor-expand the gradient around $\overline{w}_t$:

$$\nabla L(w_t) = \underbrace{\nabla L(\overline{w}_t)}_{\text{gradient at } \overline{w}} + \underbrace{H(\overline{w}_t)\delta_t}_{\text{oscillation}} + \frac{1}{2} \underbrace{\nabla_{\overline{w}_t} \, \delta_t^T H(\overline{w}_t) \delta_t}_{\text{implicit curvature penalty}} + \mathcal{O}(\|\delta_t\|^3). \quad (17)$$

[20] The flow eq. (15) actually differs slightly from the constrained trajectory in Damian et al. (2023), beyond being continuous rather than discrete. In Damian et al. (2023), the stable region was defined as the set where $S(w) \leq 2/\eta$ *and* the the loss is directionally minimized along the top Hessian eigenvector. The latter condition prevents their theory from applying to certain models (e.g. Kreisler et al., 2023, Appendix A). Our central flow does not use the latter condition and hence does not suffer from such restrictions.

[21] Similar to footnote 17, this neglects the motion in the top Hessian eigenvalues. The success of our experiments justifies this simplification.

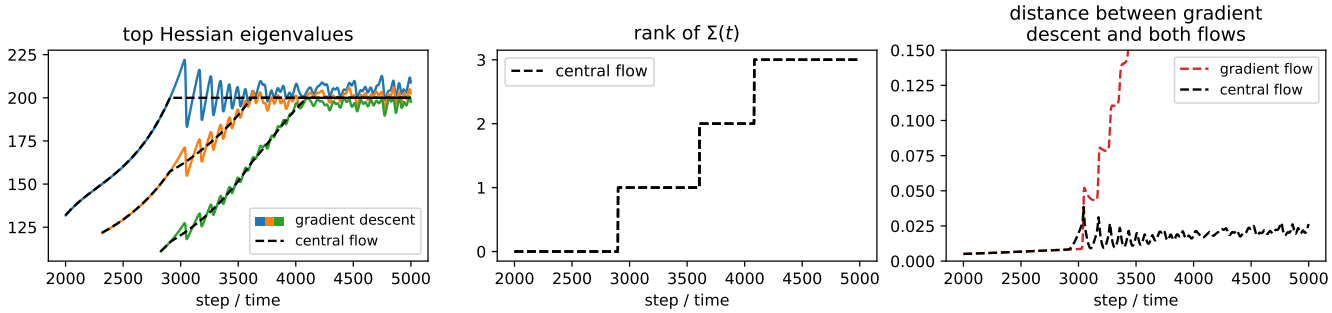


Figure 10: **Illustrating the central flow (general case)**. **Left**: whenever a Hessian eigenvalue rises to $2/\eta$, the central flow prevents it from increasing further. **Center**: since $\Sigma(t)$ models the covariance of the oscillations, its rank is always equal to the number of Hessian eigenvalues at the edge of stability. We show in Figure 11 that $\Sigma(t)$ accurately predicts the covariance of oscillations. **Right**: the Euclidean distance between gradient descent w_t and the central flow $w(t)$ stays small over time, indicating that the central flow accurately predicts the long-term trajectory of gradient descent. In contrast, the distance between gradient descent and the gradient flow eq. (6) grows large over time.

The third term in this Taylor expansion reveals that the negative gradient at the iterate w_t implicitly acts to decrease the directional curvature in the direction δ_t. Time-averaging both sides and rearranging the third term yields:

$$\mathbb{E}[\nabla L(w_t)] \approx \nabla L(\overline{w}_t) + \underbrace{\cancel{H(\overline{w}_t)\mathbb{E}[\delta_t]}}_{0 \text{ because } \mathbb{E}[\delta_t]=0} + \frac{1}{2}\underbrace{\nabla_{\overline{w}_t}\left\langle H(\overline{w}_t), \mathbb{E}[\delta_t\delta_t^T]\right\rangle}_{\text{implicit curvature penalty}}, \tag{18}$$

where we use $\langle \cdot, \cdot \rangle$ to denote the Frobenius inner product between two matrices, equivalent to flattening the matrices into vectors and taking the dot product. Thus, we see that the time-averaged gradient is the gradient at the time-averaged iterate, plus an implicit curvature penalty whose strength and direction are determined by the *covariance* of the oscillations $\mathbb{E}[\delta_t\delta_t^T]$. Substituting eq. (18) into the time-averaged gradient descent update (eq. 8) and switching to continuous time, we model the time-averaged iterates $\overline{w}_t$ by an ODE of the form:

$$\frac{dw}{dt} = -\eta\Big[\nabla L(w) + \frac{1}{2}\underbrace{\nabla_w \left\langle H(w), \Sigma(t)\right\rangle}_{\text{implicit curvature penalty}}\Big]. \tag{19}$$

Here, $\Sigma(t)$ is a still-unknown quantity intended to model $\mathbb{E}[\delta_t\delta_t^T]$, the instantaneous covariance of the oscillations at time t. This matrix also controls an implicit curvature penalty which penalizes the Σ-weighted Hessian $\langle \Sigma(t), H(w)\rangle$.[22] Similar as before, to determine $\Sigma(t)$, we impose three conditions:

1. Since Hessian eigenvalues which reach the critical threshold $2/\eta$ do not continue to rise further, we impose the condition that $\Sigma(t)$ should not allow any Hessian eigenvalues to rise beyond $2/\eta$.
2. Since gradient descent oscillates within the span of the unstable eigenvectors, we impose the condition that $\Sigma(t)$, which models the covariance of these oscillations, should be supported[23] within the span of the Hessian eigenvectors whose eigenvalue is equal to $2/\eta$.[24]
3. Since $\Sigma(t)$ models a covariance matrix, we impose the condition that $\Sigma(t)$ should be positive semidefinite.

These three conditions turn out to imply a unique value of $\Sigma(t)$. In particular, we detail in Appendix A.2 that $\Sigma(t)$ must be the unique solution to a type of convex program known as a *semidefinite complementarity problem* (SDCP),

[22] This is a weighted sum of all entries in the Hessian matrix, where each entry is weighted by the corresponding entry of $\Sigma(t)$.

[23] We mean that $\text{span}[\Sigma] \subseteq \mathcal{U}$, where $\mathcal{U}$ is the span of the Hessian eigenvectors with eigenvalue $2/\eta$. Equivalently, we mean that Σ can be written as $\Sigma = UXU^T$ where the k columns of U form a basis for the k-dimensional subspace $\mathcal{U}$, and X is a $k \times k$ symmetric matrix.

[24] For gradient descent, the unstable eigenvectors have eigenvalues which fluctuate around $2/\eta$. However, for the central flow, the unstable eigenvectors will have eigenvalues which are exactly equal to $2/\eta$.

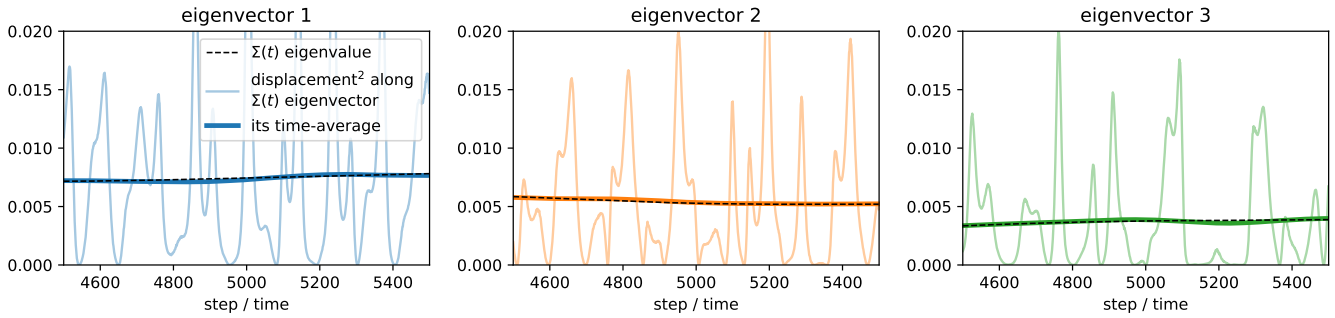


Figure 11: **Central flow can accurately predict covariance of oscillations.** We show that the central flow's $\Sigma(t)$ accurately predicts the instantaneous covariance with which gradient descent is oscillating around the central flow. For this stretch of training, there are 3 Hessian eigenvalues at EOS, so $\Sigma(t)$ has 3 nonzero eigenvalues (subpanels). In black, we plot each eigenvalue of $\Sigma(t)$; in colors, we plot the squared magnitude of gradient descent's displacement from the central flow along the corresponding eigenvector (light = raw values, dark = time average). Observe that each eigenvalue of $\Sigma(t)$ accurately predicts the instantaneous variance of oscillations along the corresponding eigenvector.

which are described in Appendix A.1.4.[25] The central flow is defined as eq. (19) with this $\Sigma(t)$:

$$\frac{dw}{dt} = -\eta\Big[\nabla L(w) + \tfrac{1}{2}\nabla_w \langle H(w), \Sigma(t)\rangle\Big] \quad \text{where} \quad \Sigma(t) \text{ solves the SDCP in eq. (70).} \tag{20}$$

A formal definition for the central flow is given in Appendix A.2, Definition 4. We note that $\Sigma(t)$ can be efficiently represented numerically as it is a low rank matrix, with rank at most the number of unstable eigenvalues.

We now elaborate on the behavior of the central flow:

1. **Stable regime:** If all Hessian eigenvalues are below $2/\eta$, then the SDCP returns $\Sigma(t) = 0$, and the central flow reduces to the gradient flow.
2. **One unstable eigenvalue**: If one Hessian eigenvalue is at $2/\eta$, and if the gradient flow would *increase* this eigenvalue above $2/\eta$, then our analysis reduces to that of Section 3.2.1. In particular, the SDCP returns a rank-one matrix of the form $\Sigma(t) = \sigma^2\, u\, u^\top$ where u is the top Hessian eigenvector at w, and σ^2 is defined in eq. (14). On the other hand, if the gradient flow would *decrease* this eigenvalue below $2/\eta$, then $\Sigma(t) = 0$, and the central flow will follow the gradient flow out of the edge of stability.
3. **Multiple unstable eigenvalues:** In general, the SDCP returns a $\Sigma(t)$ which constrains all Hessian eigenvalues currently at $2/\eta$ from rising above that value. Often, this $\Sigma(t)$ causes all Hessian eigenvalues currently at $2/\eta$ to remain fixed at $2/\eta$.[26]. However, it also allows for eigenvalues to leave EOS when appropriate.[27]

Figure 10 demonstrates the central flow in action. Initially, all Hessian eigenvalues are below $2/\eta$, so $\Sigma(t) = 0$ and the central flow reduces to the gradient flow. Once the top Hessian eigenvalue reaches $2/\eta$ around step 2900, $\Sigma(t)$ becomes a rank-one matrix, and the central flow keeps the top Hessian eigenvalue locked at $2/\eta$, as it mimics the effects of oscillating along the the top eigenvector direction. Once the second Hessian eigenvalue also reaches $2/\eta$ around step 3600, $\Sigma(t)$ becomes a rank-two matrix, and the central flow keeps the top two eigenvalues both locked at $2/\eta$, as it mimics the effects of oscillating simultaneously along the top two eigenvector directions. Throughout, the Euclidean distance between gradient descent's w_t and the central flow's $w(t)$ stays small over time (right plot),

[25] Interestingly, complementarity problems arise frequently in the the study of contact mechanics. EOS can be interpreted as the gradient descent trajectory making "contact" with the boundary of the stable region, and then sliding along the boundary.

[26] There is a unique Σ that causes all Hessian eigenvalues currently at $2/\eta$ to remain fixed at $2/\eta$, and it can be found by solving a linear inverse, generalizing eq. (13). The solution to the SDCP coincides with this Σ at almost all times. However, this Σ cannot be used to define the central flow, as it would never allow an eigenvalue to leave the edge of stability, and it is not necessarily PSD.

[27] Figure depicts an instance where an eigenvalue leaves the edge of stability.

indicating that the central flow accurately tracks the long-term trajectory of gradient descent. In contrast, the distance between gradient descent and the *gradient flow* eq. (6) grows large over time.

In Figure 11, we show that the central flow's $\Sigma(t)$ accurately predicts the covariance with which gradient descent is oscillating around the central flow. In particular, we show that each eigenvalue of $\Sigma(t)$ accurately predicts the instantaneous variance of oscillations along the corresponding eigenvector of $\Sigma(t)$. We find it striking that our theory is able to accurately predict the covariance of these oscillations. While the oscillations are erratic and might appear unpredictable, our findings reveal that a certain statistic — their covariance — is predictable after all. Moreover, predicting this covariance seems to be sufficient to predict the long-term trajectory of gradient descent.

Interpretation as projection The projection interpretation in Section 3.2.1 generalizes to the case of an arbitrary number of unstable eigenvalues. In particular, the central flow eq. (20) can be written as a flow which orthogonally projects the negative gradient onto the so-called *tangent cone* of the stable region $\mathbb{S} = \{w : S(w) \leq 2/\eta\}$, which is the set of directions in which one can move while still staying, to first order, within the stable region:[28]

$$\frac{dw}{dt} = \eta \underbrace{\text{proj}_{T_w\mathbb{S}}[-\nabla L(w)]}_{\substack{\text{project negative gradient onto} \\ \text{tangent cone } T_w\mathbb{S} \text{ of set } \mathbb{S}}} \quad \text{where} \quad \mathbb{S} = \underbrace{\{w : S(w) \leq 2/\eta\}}_{\text{stable region } \mathbb{S}}. \tag{21}$$

A formal definition is given in Definition 5. This projection interpretation will be used in Section 3.3 to show that the loss along the central flow decreases monotonically.

Where does deep learning come in? Our principal claim is that, if initialized stably, gradient descent will approximately follow the central flow over the long term. In the case where the sharpness does not rise to $2/\eta$ (e.g. on a quadratic objective, where the sharpness is constant), then the central flow reduces to the gradient flow, and so our claim reduces to the somewhat "uninteresting" claim that gradient descent will approximately follow the gradient flow. The central flow only becomes nontrivial, and our claim only becomes "interesting," in the event that the sharpness rises to $2/\eta$. This empirically tends to happen on deep learning objectives. However, we suspect that the central flow might also hold on other kinds of objectives where the sharpness rises to $2/\eta$ during gradient descent.

3.2.3 Understanding the train loss curve and more

Figure 12 shows that the loss along the actual gradient descent trajectory is consistently *higher* than the loss along the central flow. The intuitive explanation is that when gradient descent oscillates along the top Hessian eigenvector(s), it can be visualized as "bouncing between valley walls" (Xing et al., 2018; Cohen et al., 2021; Wen et al., 2025),

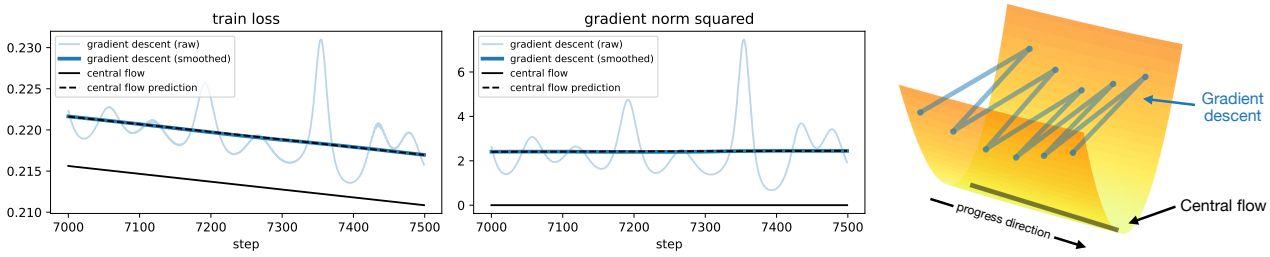


Figure 12: The train loss and gradient norm^2 are larger along the raw gradient descent trajectory (light blue) than along the central flow (solid black). The intuitive reason is that an oscillatory optimizer can be visualized as oscillating between the walls of a "valley" (see cartoon on right). Because the central flow models the covariance of the oscillations, it can render predictions for the *time-averaged* values of the loss and gradient norm^2 along the gradient descent trajectory (dashed black). These predictions are close to the empirical time-averaged values (dark blue).

[28] In the interior of the stable region (i.e. $S(w) < 2/\eta$), the tangent cone is the entire space, so this projection is a no-op and the central flow reduces to the gradient flow. When exactly one eigenvalue is at the edge of stability, the tangent cone is the half-space $\{v : \nabla S(w)^T v \leq 0\}$.

whereas the time-averaged iterates run nearly along the "valley floor" (called a "river" in Wen et al. (2025)), as illustrated on the right of Figure 12. The loss is higher on the valley walls, where the actual iterates are located, than on the valley floor, where the time-averaged iterates are located.[29]

Fortunately, the central flow framework will still let us reason about the loss along the actual gradient descent trajectory. Recall that the central flow predicts not just the time-averaged iterates $w(t)$, but also the covariance of the oscillations $\Sigma(t)$. In particular, the central flow models the gradient descent trajectory $\{w_t\}$ as:[30]

$$w_t = w(t) + \delta_t, \quad \text{where} \quad \mathbb{E}[\delta_t] = 0 \quad \text{and} \quad \mathbb{E}[\delta_t \delta_t^T] = \Sigma(t).$$

Thus, for any quantity $f(w)$ derived from the weights (e.g. loss or gradient norm), we can predict its time-averaged value $\mathbb{E}\, f(w_t)$ along the gradient descent trajectory w_t by taking a quadratic Taylor expansion along the central flow $w(t)$, and time-averaging over δ_t:

$$\underbrace{\mathbb{E}[f(w_t)]}_{\substack{\text{time-averaged} \\ \text{value along trajectory}}} \approx \underbrace{f(w(t))}_{\text{value along central flow}} + \underbrace{\tfrac{1}{2}\left\langle \nabla^2 f(w(t)), \Sigma(t) \right\rangle}_{\text{contribution from oscillations}}. \tag{22}$$

For example, if f is the loss L, then because $\Sigma(t)$ is supported on the Hessian eigenvectors with eigenvalue $2/\eta$:

$$\underbrace{\mathbb{E}[L(w_t)]}_{\substack{\text{time-averaged} \\ \text{loss along trajectory}}} \approx \underbrace{L(w(t))}_{\text{loss along central flow}} + \underbrace{\tfrac{1}{\eta}\operatorname{tr}(\Sigma(t))}_{\text{contribution from oscillations}} := \bar{L}(t). \tag{23}$$

See Appendix A.2.2 for an explicit derivation. Figure 12 shows that this prediction $\bar{L}(t)$ for the time-averaged loss closely matches the actual time-averaged loss (computed with Gaussian smoothing). Both the central flow loss $L(w(t))$ and the predicted time-averaged loss $\bar{L}(t)$ model important quantities that are meaningful to DL practice:

- The central flow's prediction for the time-averaged loss $\bar{L}(t)$ models the smoothed training loss curve, often monitored in practice by Tensorboard (Abadi et al., 2015) or Weights and Biases (Biewald, 2020).
- The central flow loss $L(w(t))$ models the loss at the time-averaged iterate. This is similar to the loss at an exponential moving average of the weights, or the loss after annealing the learning rate (Sandler et al., 2023).

The central flow perspective allows us to quantify both of these loss values and reason about them separately.

A similar point holds for other quantities,[31] such as the gradient norm. Figure 12 shows that the squared gradient norm along the central flow, $\|\nabla L(w(t))\|^2$ is much smaller than at the actual iterates, $\|\nabla L(w_t)\|^2$. Intuitively, most of the gradient at the iterates is spent "oscillating across the valley" and cancels out over the long run. The central flow's gradient is smaller because it leaves out these oscillations. Yet, because the central flow models the covariance $\Sigma(t)$ of the oscillations, it can still predict the *time-average* of the squared gradient norm at the iterates, using eq. (74) in Appendix A.2.2. Figure 12 demonstrates the accuracy of this prediction.

3.2.4 Empirical verification

We empirically find that the central flow can accurately predict the long-term trajectory of gradient descent in a variety of deep learning settings. For example, Figure 13 shows the central flow on several different deep learning settings (details in Section 6). Observe that the central flow accurately predicts the weight-space trajectory, the covariance of the oscillations, and the time-averaged training loss curve. Figures 32(a) and 32(b) show that the central flow can accurately predict the time-averaged training loss curve at different learning rates across a variety of deep learning settings. Our full set of gradient descent experiments can be found in Appendix E.1.

[29] This does not mean that the loss landscape is "locally convex" in any deep sense (indeed, the Hessian generally has negative eigenvalues). It merely reflects that the optimizer is oscillating along directions of positive curvature.

[30] We emphasize that the central flow does not model the "distribution" (so to speak) of the oscillations δ_t. Rather, it only models the second moment $\mathbb{E}[\delta_t \delta_t^T]$, under the theory that the macroscopic trajectory of gradient descent is completely characterized by this second moment.

[31] Interestingly, for some quantities (such as the network outputs), we find that the value along the central flow is already an excellent approximation to the time-averaged value along the discrete optimizer trajectory, and eq. (22) is not necessary.

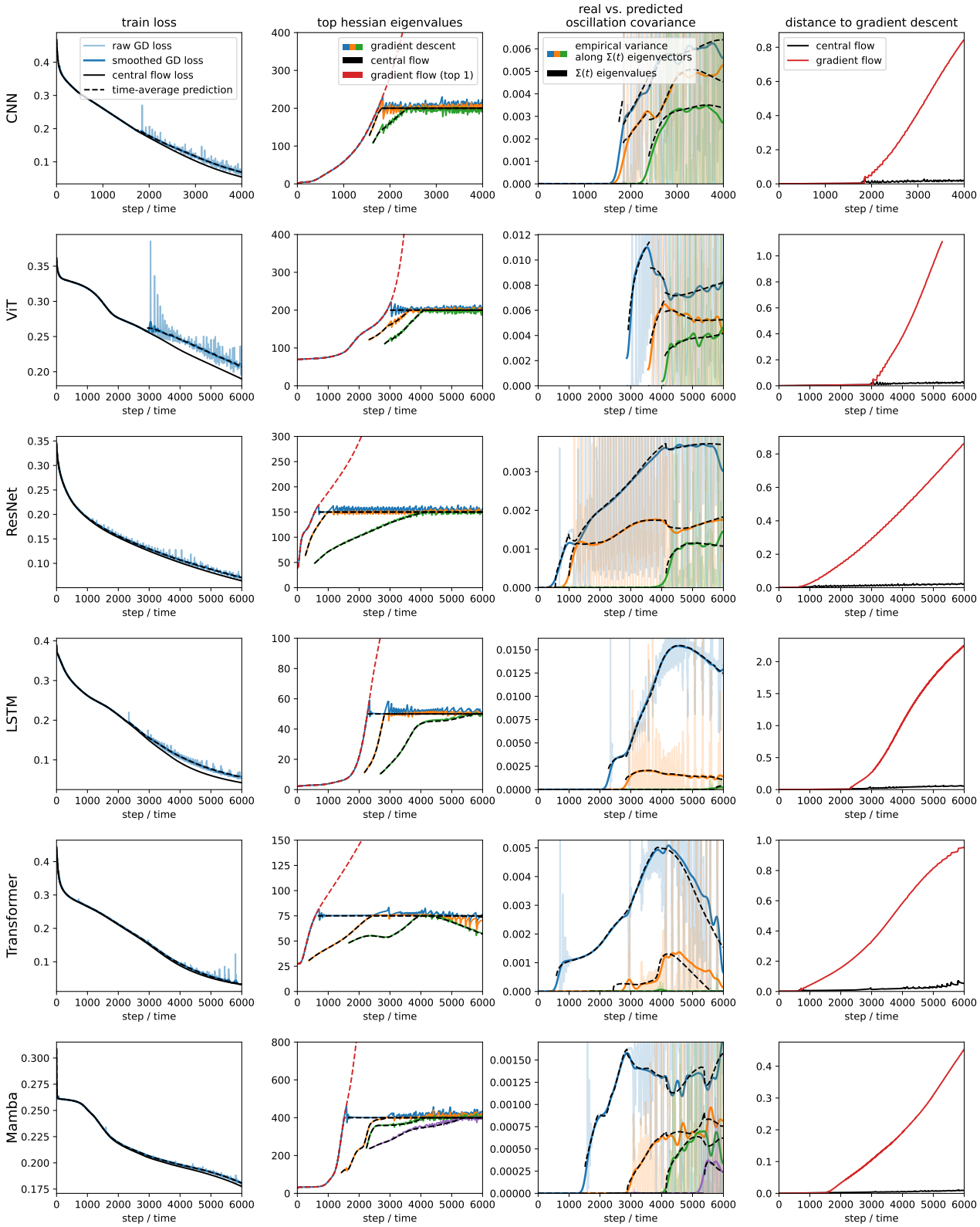


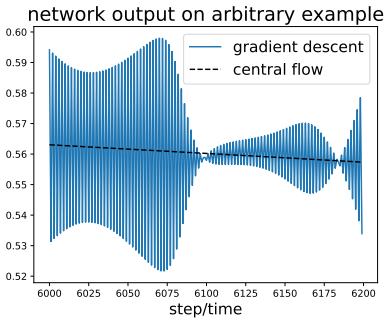
Figure 13: **Verifying the gradient descent central flow across various architectures.** Across various architectures, the central flow accurately predicts the weight-space trajectory, the covariance of the oscillations, and the time-averaged loss curve. See Section 6 for more experimental details and Appendix E for our full set of raw experiments.

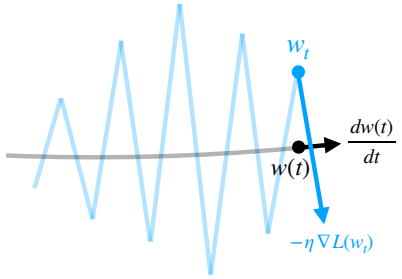
Nevertheless, our derivation relied on informal mathematical reasoning, and certain factors do empirically affect the quality of the central flow approximation. First, the central flow tends to become less accurate as the learning rate η is made increasingly large. Second, on some deep learning problems, higher-order terms cause the central flow to slightly mispredict $\Sigma(t)$, causing error to accumulate over the long run. Third, large spikes also can throw off the central flow. The latter two issues empirically seem to be more common when the loss criterion is cross-entropy rather than MSE. We discuss these points at greater length in Section 6. We hope that future work can rigorously understand the conditions under which the central flow does or does not approximate the gradient descent trajectory.

3.3 Understanding Gradient Descent via its Central Flow

We have shown that the central flow is a smooth curve that characterizes the macroscopic trajectory of gradient descent. We now explain why this makes it a useful theoretical tool for reasoning about optimization.

Averaging out oscillations reveals the underlying order At the edge of stability, gradient descent's oscillations lead to wild fluctuations in many training-related quantities, such as the network's predictions and the training loss. For example, the figure on the right shows the evolution under gradient descent of the network's prediction on an example. One can see that along the actual gradient descent trajectory (blue), training proceeds erratically. In contrast, the central flow (black) is a more coherent training process which makes steady, continuous progress over time. By averaging out the oscillations, the central flow reveals the underlying order hidden beneath the chaotic oscillatory dynamics.[32]

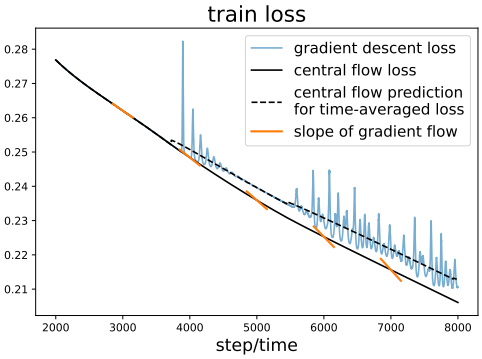




A smooth curve can be analyzed using calculus Because the central flow is a smooth curve, we can leverage calculus to reason quantitatively about the dynamics of training. Crucially, along the central flow, the time derivative $\frac{dw(t)}{dt}$ meaningfully reflects the optimizer's direction of motion over the near term (see cartoon on left). In comparison, along the gradient descent trajectory, the analogous update $-\eta \nabla L(w_t)$ is dominated by oscillations and hence does *not* meaningfully reflect the direction of motion over the near term — only over the current step.

For any quantity $f(w)$ derived from the weights w, we can use the chain rule to compute its rate of change under the central flow: $\frac{df}{dt} = \langle \nabla f(w), \frac{dw}{dt} \rangle$. We will now use this to reason about the rate of loss decrease.

Reasoning about training loss curve Consider the most basic question one can ask about an optimization algorithm: how fast is the loss going down? For the "raw" gradient descent trajectory, the loss *doesn't* always go down — instead, the loss behaves non-monotonically over short timescales, while only decreasing over long timescales. Thus, reasoning about the rate of loss decrease is challenging. In contrast, under the central flow, the loss evolves smoothly, and its rate of decrease can be quantified using the chain rule: $\frac{dL(w)}{dt} = \langle \nabla L(w), \frac{dw}{dt} \rangle$. Combining this with the projection interpretation (Definition 5), one can easily prove that the loss along the central flow $L(w(t))$ is monotonically decreasing. In other words, the central flow loss is a valid **potential function** for the optimization process:



Proposition 1. Under the central flow $w(t)$, we have $\frac{d}{dt} L(w(t)) \leq 0$.

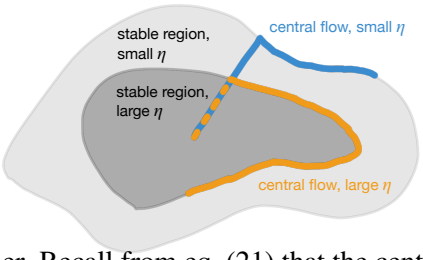
[32] Arguably, the central flow can even be viewed as the "true" training process, with the actual gradient descent trajectory being merely a noisy realization of this idealized trajectory which is computationally cheap to obtain.

See Appendix A.2.3 for the proof. The intuition is that, even after the negative gradient is projected onto the tangent cone of the stable region, it will still be negatively aligned with the gradient.

While averaging out the oscillations yields a central flow with a smoothly decreasing loss curve, the oscillations still have an effect on this loss curve through their implicit curvature reduction effect, which can be shown to slow down training. In particular, whereas the unregularized gradient flow eq. (6) decreases the loss at the speed $\frac{dL}{dt} = -\eta\|\nabla L(w)\|^2$, it is straightforward to show that the central flow optimizes at a slower speed:

Proposition 2. Under the central flow $w(t)$, we have $\frac{d}{dt}L(w(t)) \geq -\eta\|\nabla L(w)\|^2$.

See Appendix A.2.3 for the proof. The intuition is that because the central flow projects out the components of the loss gradient that would cause the sharpness to rise above $2/\eta$, it has less gradient available with which to decrease the loss. This effect is illustrated in the figure above and to the right, which shows that at various points during training, the slope of the central flow loss curve is less steep than the rate of loss decrease under the gradient flow.



Understanding the effect of hyperparameters A notorious peculiarity of deep learning is that optimizer hyperparameters affect not just the speed of training, but also the particular path that the optimizer takes through weight space (e.g. Keskar et al., 2017; Jastrzębski et al., 2019). As a result, these hyperparameters can affect many properties of the final learned model, including its robustness and generalization.[33] Such effects are *implicit* in the gradient descent update eq. (1). In contrast, the central flow renders *explicit* all effects of the learning rate hyperparameter η on the optimization process, allowing us to disentangle these effects from one another. Recall from eq. (21) that the central flow is a projected gradient flow with learning rate η that is constrained to the stable region $\mathbb{S} = \{w : S(w) \leq 2/\eta\}$. From this characterization, we see that the learning rate hyperparameter η has two distinct effects on the central flow: (1) it acts as a time rescaling, which controls the speed of optimization without affecting the overall trajectory; and (2) it determines the stable region, which affects the overall trajectory. Thus, increasing the learning rate constrains gradient descent to a smaller subset of weight space, but also allows it to traverse this set at a faster speed.[34]

Having introduced the central flows framework with an analysis of gradient descent, we will now use this methodology to understand the behavior of two adaptive optimizers.

4 Scalar RMSProp

As a stepping stone to the analysis of RMSProp in Section 5, we first study "Scalar RMSProp," a simplification of RMSProp which uses one global step size, rather than separate step sizes for each coordinate:[35][36]

$$\nu_t = \beta_2\nu_{t-1} + (1-\beta_2)\|\nabla L(w_t)\|^2, \quad w_{t+1} = w_t - \frac{\eta}{\sqrt{\nu_t}}\nabla L(w_t). \tag{24}$$

The algorithm maintains an exponential moving average (EMA), ν, of the squared gradient norm, and takes gradient steps of size $\eta/\sqrt{\nu}$, which we call the *effective step size*.[37] The EMA hyperparameter β_2 is a knob that interpolates

[33] Large learning rates are necessary for obtaining good generalization in some deep learning settings (e.g. Li et al., 2019). However, obtaining the best generalization performance usually also requires stochastic optimization with a sufficiently small batch size. Since our paper exclusively studies the deterministic setting, we decided to not focus on generalization in this paper.

[34] The learning rate η that is optimal from an optimization perspective (i.e. that will decrease the loss the fastest) will depend on the trade-off between these two effects. Empirically, we observe that for deterministic gradient descent, larger learning rates usually optimize faster (provided that training does not diverge), implying that the former effect is stronger.

[35] Note that we have re-indexed ν compared to the standard definition of RMSProp (i.e. $\nu_{t+1} \to \nu_t$). This does not affect the trajectory and just ensures the effective learning rate at step t is determined by ν_t, rather than ν_{t+1}, which simplifies the notation.

[36] This algorithm was also studied by Lyu et al. (2022). However, their analysis only applies along a manifold of global minima, as $\eta \to 0$.

[37] The terms "learning rate" and "step size" are usually interchangeable. In this paper, to avoid ambiguity, we will use the phrase "learning rate" to denote the hyperparameter, and "step size" or "effective step size" to denote the actual step sizes that are taken.

the algorithm between gradient descent when $\beta_2 = 1$ and normalized gradient descent (NGD) when $\beta_2 = 0$.[38]

While optimizers such as Scalar RMSProp are often said to utilize an "adaptive step size," it has remained unclear what precise property of the local landscape the step size is being adapted *to* (Orabona, 2020). In this section, we will use the central flows framework to answer this basic question. After describing the dynamics of Scalar RMSProp in Section 4.1 and deriving a central flow in Section 4.2, we will interpret this flow to understand the optimizer's behavior in Section 4.3. In particular:

- In Section 4.3.1, we make precise how Scalar RMSProp adapts its step size to the local loss landscape. Specifically, we show that the optimizer's dynamics implicitly set the effective step size to the value $2/S(w)$, where $S(w)$ is the current sharpness; this value is the *largest stable step size* at the current weights w.
- In Section 4.3.2, we show that step size adaptation is not the full story: Scalar RMSProp also implicitly regularizes curvature throughout training, and in fact, at EOS, the hyperparameters η, β_2 only affect the time-averaged trajectory by modulating the strength of this curvature regularization.
- Bringing it all together, in Section 4.3.3 we describe how the interplay between step size adaptation and curvature regularization gives rise to a mechanism we call *acceleration via regularization*, whereby the optimizer implicitly steers itself towards low-curvature regions where it can take larger steps. We show that this mechanism is key to the efficacy of Scalar RMSProp and to the function of its hyperparameters.

These points will generalize to RMSProp in Section 5, but are simpler to understand for Scalar RMSProp.

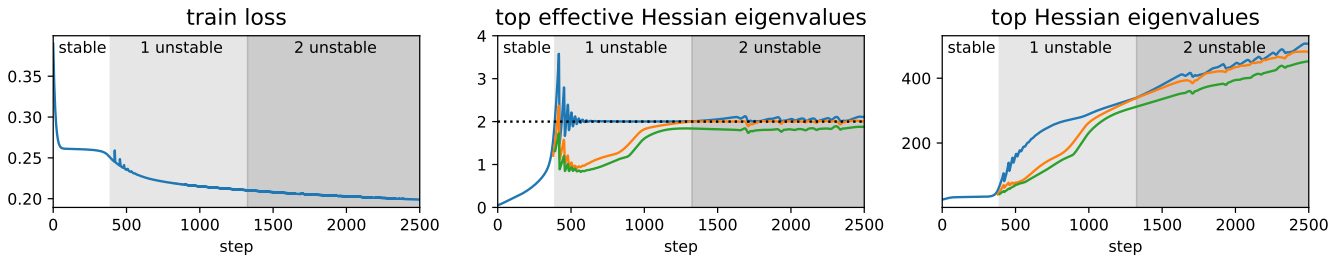


Figure 14: **A typical Scalar RMSProp trajectory**. We train a Mamba network on a sequence task using Scalar RMSProp with $\eta = 2/400$ and $\beta_2 = 0.99$. While the top eigenvalues of the "raw" Hessian $H(w)$ evolve freely (right), the top eigenvalue of the *effective* Hessian $\eta H(w)/\sqrt{\nu}$ equilibrates at the critical threshold 2 (center).

4.1 The Dynamics of Scalar RMSProp

The dynamics of Scalar RMSProp revolve around the *effective sharpness*, defined as $S^{\text{eff}} := \eta S(w)/\sqrt{\nu}$.[39] First, the effective sharpness controls the oscillations: when $S^{\text{eff}} > 2$, Scalar RMSProp oscillates with growing magnitude along high curvature direction(s). Second, such oscillations in turn trigger a reduction of effective sharpness. This occurs via a combination of two distinct mechanisms. One mechanism, shared with gradient descent, is that oscillations implicitly reduce sharpness due to eq. (9), thereby decreasing the effective sharpness via its *numerator*. The other mechanism, new to Scalar RMSProp, is that oscillations increase the gradient norm and hence ν, thereby decreasing effective sharpness via its *denominator*. These dynamics give rise to a negative feedback loop that keeps the effective sharpness automatically regulated around the value 2, as depicted in Figure 14. The fine-grained dynamics are complex and challenging to analyze, even in the case of a single oscillatory direction. Fortunately, we will see in the next section that analyzing the *time-averaged* dynamics is much simpler.

[38]When $\beta_2 = 1$, Scalar RMSProp reduces to gradient descent with learning rate $\eta/\sqrt{\nu_0}$. Conversely, when $\beta_2 = 0$, it reduces to normalized gradient descent with learning rate η: $w_{t+1} = w_t - \eta \cdot \frac{\nabla L(w_t)}{\|\nabla L(w_t)\|}$.

[39]While we could have defined the effective sharpness as $S(w)/\sqrt{\nu}$ so that it would equilibrate at $2/\eta$ rather than 2, this version makes the analysis easier.

4.2 Deriving the Scalar RMSProp Central Flow

Recall that while gradient descent trains stably, it is well-approximated by gradient flow. One can derive an analogous "stable flow" for Scalar RMSProp (Ma et al., 2022, cf.):[40]

$$\frac{dw}{dt} = -\frac{\eta}{\sqrt{\nu}}\nabla L(w), \qquad \frac{d\nu}{dt} = \frac{1-\beta_2}{\beta_2}\left[\|\nabla L(w)\|^2 - \nu\right]. \tag{25}$$

However, at the edge of stability, the trajectory of Scalar RMSProp deviates from eq. (25). We will now derive a more general *central flow* that characterizes the time-averaged trajectory even at EOS. In the main text, we will focus on the case where one eigenvalue is (and remains at) the edge of stability. See Appendix A.3 for our full derivation which accounts for multiple eigenvalues at EOS and for eigenvalues entering and leaving EOS.

In Section 3.2.1, we derived an approximation for the time-averaged gradient, $\mathbb{E}[\nabla L(w)]$. Using the first two terms of eq. (9), we can also derive a time-averaged approximation for the squared gradient norm $\mathbb{E}[\|\nabla L(w)\|^2]$:

$$\mathbb{E}[\|\nabla L(w)\|^2] \approx \|\nabla L(\overline{w})\|^2 + \cancel{2\left\langle \nabla L(\overline{w}), \overline{w} \right\rangle S(\overline{w})\,\mathbb{E}[x]} + S(\overline{w})^2\,\mathbb{E}[x^2]$$

where we again used $\mathbb{E}[x] = 0$ to ignore the middle term. This calculation makes clear that larger oscillations (i.e. higher $\mathbb{E}[x^2]$) increase the squared gradient norm on average over time. Based on these time averages, we make the ansatz that the joint dynamics of (w_t, ν_t) follow a central flow $(w(t), \nu(t))$ of the form:

$$\frac{dw}{dt} = -\frac{\eta}{\sqrt{\nu}}\Big[\underbrace{\nabla L(w) + \tfrac{1}{2}\sigma^2(t)\nabla S(w)}_{\mathbb{E}[\nabla L(w_t)]}\Big], \qquad \frac{d\nu}{dt} = \frac{1-\beta_2}{\beta_2}\Big[\underbrace{\|\nabla L(w)\|^2 + S(w)^2\sigma^2(t)}_{\mathbb{E}[\|\nabla L(w_t)\|^2]} - \nu\Big], \tag{26}$$

where $\sigma^2(t)$ is a still-unknown quantity intended to model $\mathbb{E}[x_t^2]$, the instantaneous variance of the oscillations. As in our analysis of gradient descent, there is a unique value of $\sigma^2(t)$ that maintains $S^{\text{eff}}(w, \nu) = 2$. To compute it, we expand $\frac{dS^{\text{eff}}}{dt}$ using the chain rule: $\frac{dS^{\text{eff}}}{dt} = \langle \frac{\partial S^{\text{eff}}}{\partial w}, \frac{dw}{dt} \rangle + \frac{\partial S^{\text{eff}}}{\partial \nu} \cdot \frac{d\nu}{dt}$. Plugging in $\frac{dw}{dt}$, $\frac{d\nu}{dt}$ from eq. (26) shows that $\frac{dS^{\text{eff}}}{dt}$ is linear in σ^2. Thus, there is a unique value of σ^2 that will ensure $\frac{dS^{\text{eff}}}{dt} = 0$, which is given by:

$$\sigma^2(w; \eta, \beta_2) = \frac{\beta_2 \overbrace{\langle -\nabla L(w), \nabla S(w)\rangle}^{\text{progressive sharpening}} + (1-\beta_2)\overbrace{\left[S(w)^2/4 - \|\nabla L(w)\|^2/\eta^2\right]}^{\text{effect of mean reversion on } \nu}}{\beta_2 \underbrace{\frac{1}{2}\|\nabla S(w)\|^2}_{\text{sharpness reduction}} + (1-\beta_2)\underbrace{S(w)^2/\eta^2}_{\text{effect of oscillation on } \nu}}. \tag{27}$$

The central flow for Scalar RMSProp with one unstable eigenvalue is given by eq. (26) with this value of σ^2.[41] The full central flow, derived in Appendix A.3, is given in Definition 7. Figure 15 illustrates how this central flow can accurately predict the long-term trajectory of Scalar RMSProp as well as the covariance with which Scalar RMSProp is oscillating around that trajectory. Figures 33(a) and 33(b) show, in a variety of deep learning settings, that this central flow can accurately predict the loss curve of Scalar RMSProp at different learning rates. Figure 36 shows that the central flow holds across different values of β_2. See Appendix E.2 for the full set of raw experiments.

The analysis in this section highlights the potential of our time-averaging methodology. With just a single invocation of the chain rule, we have characterized the long-term trajectory of a complex dynamical system involving mutual interactions between the oscillations, the sharpness, and the adaptive step size.

[40] The $1 - \beta_2 \to \frac{1-\beta_2}{\beta_2}$ correction is necessary for small values of β_2. For example, when $\beta_2 = 0$ (i.e. normalized gradient descent), $\nu_t = \|\nabla L(w_t)\|^2$ so in the continuous time ODE, $\nu(t)$ needs to adapt "instantly" to $\|\nabla L(w(t))\|^2$. See Appendix A.7 for additional justification for this correction term.

[41] The Scalar RMSProp central flow can be interpreted as a projected flow in the augmented space (w, ν) under a certain non-Euclidean norm. However, because this flow is not a gradient flow, it does not immediately suggest a decreasing potential function for Scalar RMSProp.

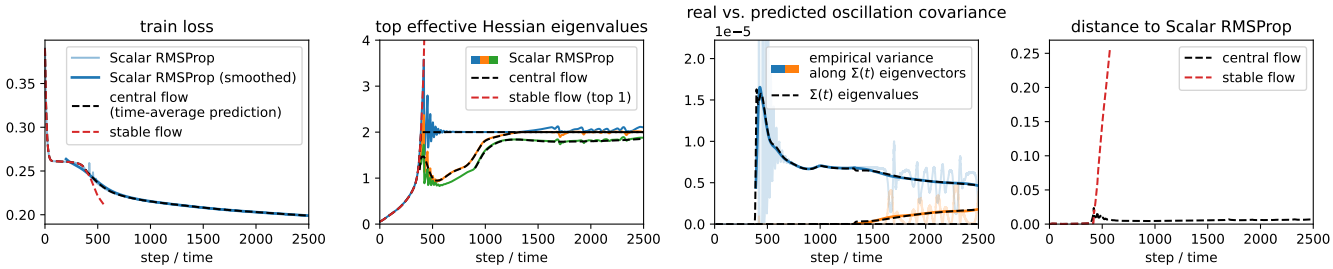


Figure 15: **Central flow for Scalar RMSProp.** The central flow (black) accurately models the time-averaged trajectory of Scalar RMSProp even at the edge of stability, whereas the naive stable flow (red) follows a different path. As with gradient descent, our analysis can predict the exact covariance Σ with which Scalar RMSProp oscillates around the central flow (third panel). The setting is the same as Figure 14.

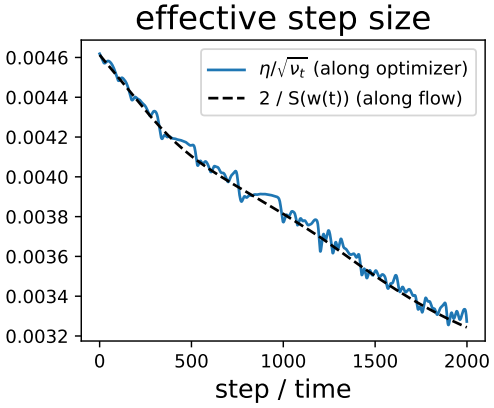
4.3 Understanding Scalar RMSProp via its Central Flow

We now interpret the Scalar RMSProp central flow to shed light on the behavior of the algorithm and the function of its hyperparameters η and β_2. Because the dynamics usually transition from stable to EOS quite early in training, we focus on interpreting the central flow in the EOS regime.[42]

4.3.1 Implicit step size selection

The central flow renders *explicit* the step size strategy that is *implicit* in the oscillatory dynamics of Scalar RMSProp. Recall that while the central flow is at EOS, the effective sharpness $S^{\text{eff}} := \eta S(w)/\sqrt{\nu}$ is fixed at 2. Indeed, this is the equilibrium condition that is automatically maintained by the dynamics of optimization. This EOS condition can be rearranged into a statement about the effective step size:

$$\eta/\sqrt{\nu} = 2/S(w). \tag{28}$$



That is, at EOS, the effective step size along the central flow is always equal to the value $2/S(w)$. Notably, the value $2/S(w)$ is the *largest stable step size* for gradient descent at location w. Thus, while Scalar RMSProp is at EOS, **the oscillatory dynamics continually adapt the effective step size to the current largest stable step size**, even as this value evolves throughout training. This is the precise sense in which Scalar RMSProp "adapts" its step size to the local loss landscape.

In principle, it would be possible for an optimizer to *manually* compute the sharpness $S(w)$ at each iteration (e.g. by using the power method), and to manually set the step size to $2/S(w)$. However, computing the sharpness would incur some computational overhead, whereas we have shown that Scalar RMSProp finds the maximum stable step size of $2/S(w)$ *efficiently*, using no more computation than is already used by gradient descent (namely, one gradient computation per iteration). This rich behavior is implicit in the algorithm's oscillatory dynamics.

Furthermore, note that even comprehending this behavior requires an appeal to some notion of time-averaging. The effective step size is usually not *exactly* at $2/S(w)$, but rather is fluctuating around $2/S(w)$. The important point is that it is $2/S(w)$ on average over time. The central flow perspective gives a way to reason about this behavior.

[42] In the stable regime ($S^{\text{eff}} < 2$), the central flow is given by the stable flow eq. (25). For this flow, $\frac{dw}{dt}$ is directly proportional to that of gradient flow, implying these flows traverse the same trajectory, just at a different speed (i.e. with a nonlinear time-rescaling). In this regime, the effective step size generally increases monotonically, so Scalar RMSProp follows gradient flow with a learning rate warmup.

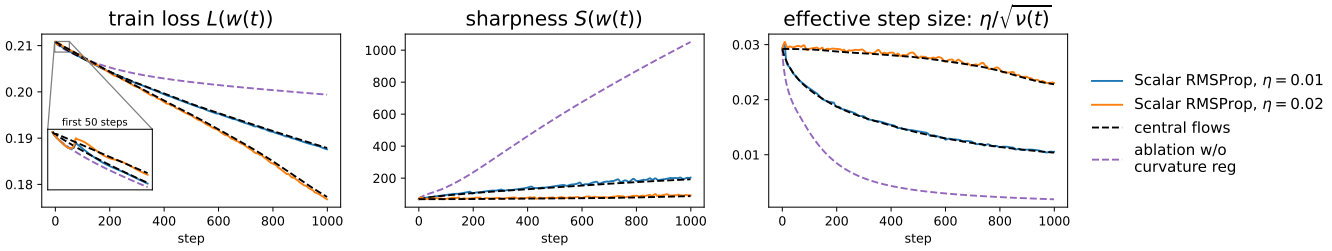


Figure 16: **Implicit curvature regularization accelerates optimization for Scalar RMSProp**. Starting from the same initial point, we run Scalar RMSProp at two different learning rates (blue and orange), alongside the corresponding central flows (black). We also run an ablated flow $\frac{dw}{dt} = -\frac{2}{S(w)}\nabla L(w)$ which has curvature regularization removed (purple). All three flows use the same step size strategy, and differ only in the strength of implicit curvature regularization. Initially (see inset), the flows with higher curvature regularization optimize slower; however, over the longer run, they take larger steps and optimize faster. This figure is in the same setting as Figure 15.[45]

4.3.2 Implicit curvature reduction

Understanding the implicit step size strategy employed by Scalar RMSProp is not sufficient to fully characterize the behavior of the algorithm. To do so, we need to return to the central flow, which additionally accounts for the curvature regularization induced by oscillations. In general, the Scalar RMSProp central flow is a joint flow over (w, ν). However, at EOS, because $\eta/\sqrt{\nu} = 2/S(w)$, we can eliminate ν from the expression for $\frac{dw}{dt}$, and write the central flow in terms of w alone:[43]

$$\frac{dw}{dt} = -\underbrace{\frac{2}{S(w)}}_{\text{effective step size}}\Big[\nabla L(w) + \underbrace{\frac{1}{2}\sigma^2(w;\eta,\beta_2)\nabla S(w)}_{\text{implicit sharpness penalty}}\Big] \tag{29}$$

where $\sigma^2(w;\eta,\beta_2)$ is given by eq. (27). In other words, the time-averaged trajectory of Scalar RMSProp at EOS is essentially equivalent to that of the following simpler-to-understand algorithm:

> At each iteration, compute the sharpness $S(w)$, and take a gradient step of size $2/S(w)$ on a sharpness-regularized objective, where the strength of the sharpness regularizer is given by eq. (27).

Interestingly, the hyperparameters η, β_2 are not used to determine the effective step size $2/S(w)$. Instead, their only role is to modulate σ^2, which controls the strength of the implicit sharpness penalty. The effect of the learning rate hyperparameter η is to *monotonically increase* σ^2 — indeed, the numerator of eq. (27) is increasing in η while the denominator is decreasing in η, which implies the overall expression for σ^2 is increasing in η. The simplest case is that of NGD, i.e. when $\beta_2 = 0$, for which eq. (27) reduces to $\sigma^2 \approx \frac{\eta^2}{4}$ (see Appendix A.3). Meanwhile, the effect of the hyperparameter β_2 is to monotonically interpolate σ^2 between that of normalized gradient descent when $\beta_2 = 0$ and that of gradient descent when $\beta_2 = 1$.[44] The interpretations of η, β_2 generalize to the setting of multiple oscillating directions, as detailed in **??**.

4.3.3 Acceleration via regularization

To fully grasp the *modus operandi* of Scalar RMSProp, it is necessary to consider the link between step size adaptation and curvature regularization. By regularizing sharpness $S(w)$, Scalar RMSProp is able to steer itself towards regions

[43] Note that at EOS we can rearrange the EOS condition as $\nu = \eta^2 S(w)^2/4$, which lets us write ν as a function of w and eliminate ν everywhere in the central flow. This was already used to derive the expression for σ^2 in eq. (27).

[44] We note that which of these is larger is situation dependent, so σ^2 can be either monotonically increasing or monotonically decreasing in β_2. That said, because when $\beta_2 = 0$, $\sigma^2(w;\eta,0) \approx \eta^2/4$ and when $\beta_2 = 1$, $\sigma^2(w;\eta,1)$ is independent of η, a general rule is that for small learning rates, σ^2 is monotonically increasing in β_2, while for large learning rates, σ^2 is monotonically decreasing in β_2.

[45] In this figure, for Scalar RMSProp, we show the train loss at the second-order midpoints between iterates (see Appendix B.1).

where the maximal locally stable step size of $2/S(w)$ is larger. In such regions, Scalar RMSProp can and does take larger steps. Thus, **by regularizing sharpness, Scalar RMSProp enables itself to take larger steps later in training**. We call this mechanism *acceleration via regularization*. Our experiments suggest that this mechanism is a critical component of the algorithm's effectiveness. In Figure 16, we compare the Scalar RMSProp central flow to an ablated version which adapts the step size to $2/S(w)$ but does not regularize sharpness. Over the long term, this ablated flow optimizes slower than the Scalar RMSProp central flow, because it traverses sharper regions of weight space in which it is forced to take smaller steps. (See Appendix D, Figure 40 for more settings.)

The mechanism of "acceleration via regularization" is also key for understanding the function of the learning rate hyperparameter η. We have seen that at EOS, the only direct effect of η on the central flow is to modulate the strength of sharpness regularization, with higher η inducing stronger sharpness regularization. Thus, counterintuitively, the *instantaneous* effect of a higher η is often to *slow down* optimization. However, as we illustrate in Figures 16 and 40, over longer timescales, higher η steers the trajectory into lower-sharpness regions, in which Scalar RMSProp's effective step size will be larger, thereby tending to *speed up* optimization. Thus, as one would expect of a learning rate hyperparameter, larger η can accelerate optimization; however they do so through this *indirect* mechanism.

5 RMSProp

We now study RMSProp (Tieleman and Hinton, 2012), which is equivalent to Adam (Kingma and Ba, 2015) without momentum. RMSProp maintains an EMA ν of the elementwise squared gradients $\nabla L(w)^{\odot 2}$, and uses *per-coordinate* effective step sizes of $\eta/\sqrt{\nu}$:[46]

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2)\nabla L(w_t)^{\odot 2}, \quad w_{t+1} = w_t - \frac{\eta}{\sqrt{\nu_t}} \odot \nabla L(w_t), \tag{30}$$

where $\odot$ represents the entrywise product. RMSProp can also be viewed as preconditioned gradient descent $w_{t+1} = w_t - P_t^{-1}\nabla L(w_t)$ with the dynamic preconditioner $P_t := \mathrm{diag}(\sqrt{\nu_t}/\eta)$.[47] While Adam employs the same dynamic preconditioner and has achieved widespread success in deep learning, it has remained unclear why this specific preconditioning strategy is so effective (Kunstner et al., 2019; Orabona, 2020; Martens, 2020). A common folklore belief is that Adam/RMSProp adapts to the local "curvature" (i.e. Hessian). However, it is a priori unclear how this can be so, since the algorithm uses the (squared) *gradient*, not the Hessian, to update its preconditioner.

In this section, we use the central flows framework to understand the behavior of RMSProp. We will show that RMSProp *does* adapt to the local Hessian after all, but the reason is inextricably tied to its oscillatory dynamics, which have not been previously studied.

We start by describing the dynamics of RMSProp in Section 5.1. We then derive a central flow in Section 5.2. Finally, in Section 5.3, we interpret this flow to understand the optimizer's behavior. In particular:

- In Section 5.3.1, we show that RMSProp's preconditioner is *implicitly* determined by the algorithm's oscillatory dynamics, and we make this preconditioner *explicit* for the first time. Specifically, we show that RMSProp computes its preconditioner by solving a convex program (eq. 35) involving the Hessian. This clarifies that RMSProp is **implicitly a second-order optimizer**, despite only accessing the loss through first-order gradients.
- In Section 5.3.2, we show that, like Scalar RMSProp, the success of RMSProp relies not only on this preconditioning strategy, but also on an *acceleration via regularization* mechanism whereby implicitly regularizing curvature allows the optimizer to take larger steps later in training.

[46] Our analysis can accommodate both bias correction and an ϵ-dampening (dividing by $\sqrt{\nu+\epsilon}$ rather than $\sqrt{\nu}$) which are used by Adam (see Appendix A.5). However, to simplify exposition, the main text focuses on this simpler version of RMSProp.

[47] Folding η into the preconditioner is unconventional, but will make the analysis clearer.

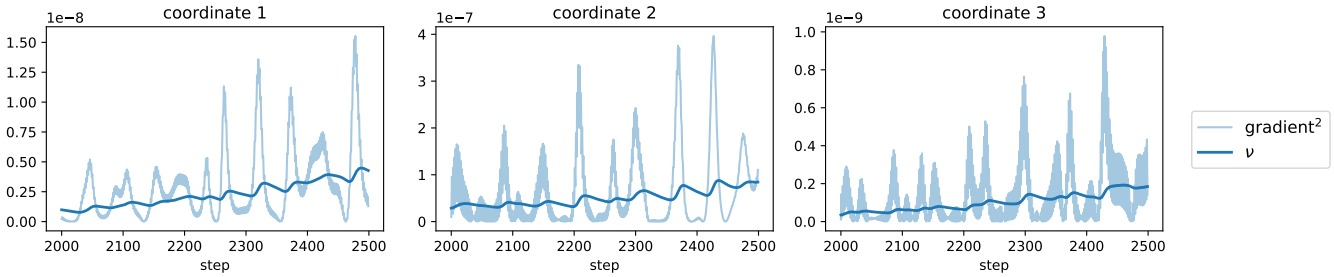


Figure 17: **RMSProp ν is determined by oscillations.** While training a network using RMSProp, we plot the squared gradient $\nabla L(w_t)^{\odot 2}$ (light blue) and its EMA ν_t (dark blue) at three coordinates (subpanels). Due to the EOS oscillations, the squared gradient fluctuates, causing the EMA ν_t to also fluctuate. Since this EMA is used to determine the effective step sizes $\eta/\sqrt{\nu_t}$, analyzing these dynamics is necessary for understanding RMSProp's adaptivity. This network is a ResNet trained on a subset of CIFAR-10 using $\eta =$ 2e-5, $\beta_2 = 0.99$ and MSE loss.

5.1 The Dynamics of RMSProp

To give some intuition into RMSProp's behavior, Figure 17 plots the dynamics of the squared gradient $\nabla L(w_t)^{\odot 2}$ and its EMA ν_t at several coordinates over a stretch of training. Observe that the entries of the squared gradient fluctuate rapidly, causing their EMA to also fluctuate. Since this EMA ν directly determines the effective step sizes $\eta/\sqrt{\nu}$, understanding the origin of this behavior is necessary to understand how RMSProp sets its effective step sizes.

These fluctuations in the gradient arise because RMSProp is operating in an oscillatory *edge of stability* regime. To understand why RMSProp oscillates, first consider running preconditioned gradient descent $w_{t+1} = w_t - P^{-1}\nabla L(w_t)$ on a quadratic function with Hessian H. The resulting dynamics are controlled by the *effective Hessian* $P^{-1}H$. Namely, if any eigenvalues of this matrix exceed the critical threshold 2, then preconditioned GD will oscillate with exponentially growing magnitude along the corresponding (right) eigenvectors.[48] For RMSProp in deep learning, both the Hessian $H(w_t)$ and the preconditioner $P_t = \mathrm{diag}(\sqrt{\nu_t}/\eta)$ can vary. However, a local quadratic Taylor approximation suggests that RMSProp will oscillate if the largest eigenvalue of the *current* effective Hessian $P_t^{-1}H(w_t)$ exceeds the critical threshold 2.[49] We refer to this quantity as the effective sharpness $S^{\text{eff}}(w_t, \nu_t)$:

$$S^{\text{eff}}(w_t, \nu_t) := \lambda_1(P_t^{-1}H(w_t)). \tag{31}$$

Paralleling the dynamics of gradient descent in deep learning, Cohen et al. (2022) observed that RMSProp typically operates in an oscillatory EOS regime that revolves around the effective sharpness eq. (31). On the one hand, oscillations ensue whenever the effective sharpness rises above the critical threshold 2.[50] On the other hand, such oscillations reduce the effective sharpness, both by inducing implicit regularization of curvature (i.e. shrinking $H(w_t)$), and by growing the gradient and hence the preconditioner P_t. The net result is that the effective sharpness equilibrates around the value 2 (as shown in Figure 18), as the optimizer oscillates along the top eigenvectors of the effective Hessian.

5.2 Deriving the RMSProp Central Flow

Similar as before, we now derive a central flow $(w(t), \nu(t))$ that jointly models the time-averaged dynamics of w_t, ν_t. We defer the full details to Appendix A.4 and sketch the argument here.

[48] On a quadratic $\frac{1}{2}w^T Hw$, this algorithm evolves via: $w_{t+1} = (I - P^{-1}H)w_t \implies w_t = (I - P^{-1}H)w_0$. If $P^{-1}H$ has any eigenvalues greater than 2, $(I - P^{-1}H)$ has eigenvalues less than -1, and the iterates diverge along the corresponding right eigenvectors.

[49] With this argument, we are also implicitly assuming that the preconditioner evolves sufficiently slowly that its movement can be neglected.

[50] For RMSProp, the effective sharpness eq. (31) tends to rise *both* because the curvature tends to rise (progressive sharpening) *and* because the gradient (and hence ν) tends to shrink. Due to the second effect, RMSProp often enters EOS sooner, and for a large range of learning rates, than gradient descent. Further, RMSProp enters EOS even on quadratics, whereas gradient descent does not.

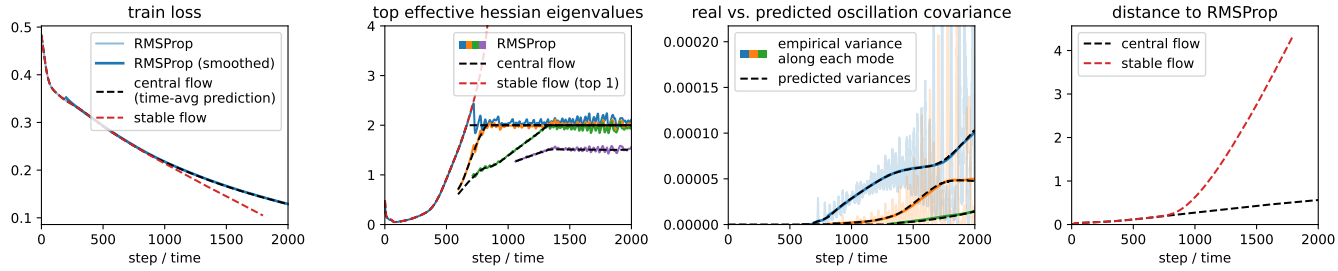


Figure 18: **Central flow for RMSProp.** The RMSProp central flow (black) accurately models the macroscopic trajectory of RMSProp even at EOS, whereas the naive stable flow (red) follows a different path. As for gradient descent and Scalar RMSProp, we are able to predict the exact covariance $\Sigma(t)$ with which RMSProp oscillates around the central flow (third panel). This figure is in the same setting as Figure 17.

If RMSProp is oscillating around its time-averaged trajectory $\{\overline{w}_t\}$, so that $w_t = \overline{w}_t + \delta_t$, then the time average of the elementwise squared gradient is approximately:

$$\underbrace{\mathbb{E}[\nabla L(w_t)^{\odot 2}]}_{\substack{\text{time-average of} \\ \text{squared gradient}}} \approx \underbrace{\nabla L(\overline{w}_t)^{\odot 2}}_{\substack{\text{squared gradient at} \\ \text{time-averaged iterate}}} + \underbrace{\operatorname{diag}[H(\overline{w}_t)\,\mathbb{E}[\delta_t \delta_t^T] H(\overline{w}_t)]}_{\text{contribution from oscillations}}. \tag{32}$$

The first term is the squared gradient at the time-averaged iterate; the second term is the contribution to the squared gradient that originates from oscillating with covariance $\mathbb{E}[\delta_t \delta_t^T]$.

If we further assume then these oscillations are contained within the right eigenspace of the effective Hessian $\operatorname{diag}(\eta/\sqrt{\nu_t})H(w_t)$ that corresponds to the eigenvalue 2, then the rightmost term simplifies as follows:

$$\underbrace{\mathbb{E}[\nabla L(w_t)^{\odot 2}]}_{\substack{\text{time-average of} \\ \text{squared gradient}}} \approx \underbrace{\nabla L(\overline{w}_t)^{\odot 2}}_{\substack{\text{squared gradient at} \\ \text{time-averaged iterate}}} + \underbrace{\frac{4\nu}{\eta^2} \odot \operatorname{diag}[\mathbb{E}[\delta_t \delta_t^T]]}_{\text{contribution from oscillations}} \tag{33}$$

Based on this calculation, and on the time-averaged gradient computed in eq. (18), we make the ansatz that the time-averaged dynamics of w_t, ν_t follow a central flow $(w(t), \nu(t))$ with the functional form:

$$\frac{dw}{dt} = -\frac{\eta}{\sqrt{\nu}} \odot \left[\underbrace{\nabla L(w) + \tfrac{1}{2}\nabla \langle \Sigma(t), H(w) \rangle}_{\mathbb{E}[\nabla L(w_t)]}\right], \quad \frac{d\nu}{dt} = \frac{1-\beta_2}{\beta_2}\left[\underbrace{\nabla L(w)^{\odot 2} + \frac{4\nu}{\eta^2} \odot \operatorname{diag}[\Sigma(t)]}_{\mathbb{E}[\nabla L(w_t)^{\odot 2}]} - \nu\right]. \tag{34}$$

To determine $\Sigma(t)$, we impose three conditions on this flow, analogous to those from Section 3.2.2. As before, it can be shown that there is a unique matrix $\Sigma(t)$ satisfying these three conditions, and this matrix can be characterized as the solution to a semidefinite complementarity problem. The RMSProp central flow is defined as eq. (34) with this value of $\Sigma(t)$. See Appendix A.4, Definition 9 for a formal statement.

Figure 18 illustrates how this central flow can accurately predict the macroscopic trajectory $w(t)$ of RMSProp, as well as the covariance $\Sigma(t)$ with which RMSProp is oscillating around that trajectory.[51] Figure 19 shows how the central flow can accurately predict the time-average of the elementwise squared gradient via eq. (33), as well as the macroscopic trajectory $\nu(t)$ of the EMA. Figure 34(a) and Figure 34(b) in Appendix D show in a variety of deep learning settings that the central flow can accurately predict the RMSProp loss curve across different learning rates. Figure 35 and Figure 37 show that the central flow can accurately predict the RMSProp trajectory at different values of β_2 and ϵ, respectively. The full set of raw RMSProp experiments can be found in Appendix E.3.

[51] To match the preconditioned geometry of the optimizer, we assess whether each eigenvalue of $P(\nu(t))^{1/2}\Sigma(t)\,P(\nu(t))^{1/2}$ accurately predicts the P-whitened variance of oscillations along the corresponding eigenvector; see eq. (120) in Appendix A.4.

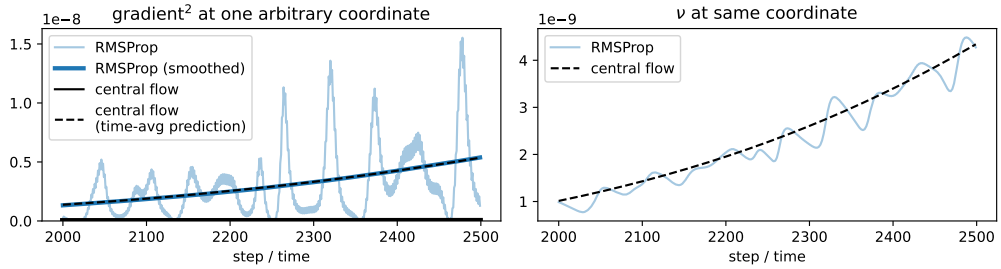


Figure 19: **Central flow can successfully predict both the time-averaged gradient^2 (left) and the EMA ν (right).** On the left, we show that although the squared gradient is fluctuating erratically (recall Figure 17), its time-average can be predicted by the central flow using eq. (33). On the right, we show that the central flow's $\nu(t)$ accurately tracks the macroscopic trajectory of the real EMA ν_t. This figure is in the same setting as Figure 17.

As with gradient descent, we find that the central flow approximation tends to become less accurate as the learning rate η grows; see Section 6 for our general discussion about the accuracy of the central flow. In addition, we observe that the central flow for RMSProp tends to be a bit less accurate overall than that for gradient descent, at least as measured by weight-space distance between the flow and the discrete optimizer. Finally, we expect the central flow for RMSProp to break down when β_2 becomes too close to zero (i.e. the sign GD limit), as then RMSProp would no longer resemble preconditioned gradient descent with a slowly-changing preconditioner.

5.3 Understanding RMSProp via its Central Flow

We now interpret the RMSProp central flow to understand the behavior of RMSProp, including how the algorithm sets its effective step sizes $\eta/\sqrt{\nu}$. Because the dynamics usually transition from stable to EOS early in training, we focus on the EOS regime.

5.3.1 The stationary preconditioner

Stationarity of ν Unfortunately, even at the edge of stability, $\nu(t)$ cannot be expressed as a closed-form function of $w(t)$ (as it could for Scalar RMSProp in Section 4), and instead remains an independent variable that must be tracked. This reflects the fact that for any w, there are potentially many values for ν that could stabilize optimization, and the actual value used by RMSProp depends on the history. Nevertheless, we will now see that under the RMSProp central flow, ν often implicitly converges to a value that depends on the current w alone.

Intuitively, the RMSProp central flow eq. (34) involves two simultaneous processes of optimization (the w dynamics) and preconditioner adaptation (the ν dynamics). Suppose that the ν dynamics of preconditioner adaptation occur *fast* relative to the w dynamics of optimization, so that ν reaches a stationary point w.r.t the current weights w. In Proposition 6 we show that for any w, there is in fact a *unique* ν that satisfies the stationarity condition $\frac{d\nu}{dt} = 0$. We call this unique ν the *stationary* ν for the weights w, denoted as $\overline{\nu}(w)$. Empirically, we observe that $\nu(t)$ usually starts to attain its stationary value $\overline{\nu}(w(t))$ at some point during training (after the dynamics have entered EOS), and continues to match $\overline{\nu}(w(t))$ thereafter, even as this value evolves. Indeed, Figure 20 illustrates how $\nu(t)$ converges to $\overline{\nu}(w(t))$ both in cosine similarity (left) and coordinate-wise (right). See Figures 42(a) to 43(b) for more settings.[52]

The stationarity of ν will allow us to reason about RMSProp's preconditioning strategy with relative ease, i.e. without needing to account for the history of ν. At any weights w, we can view the corresponding *stationary preconditioner* $\overline{P}(w) := \mathrm{diag}(\sqrt{\overline{\nu}(w)}/\eta)$ as "the RMSProp preconditioner" that is implicitly used by RMSProp at weights w. We will now interpret this preconditioner to gain insight into RMSProp's preconditioning strategy.

[52] Since the speed of the ν dynamics in eq. (34) is controlled by the β_2 hyperparameter, one might suspect that $\nu(t)$ will converge quicker to $\overline{\nu}(w(t))$ when β_2 is smaller, and we confirm this in Figure 44. Nevertheless, we emphasize that the quasistationarity of ν w.r.t w empirically holds even when β_2 is relatively large (e.g. 0.99). In keeping with our attitude throughout this paper, we do not claim to have an explanation.

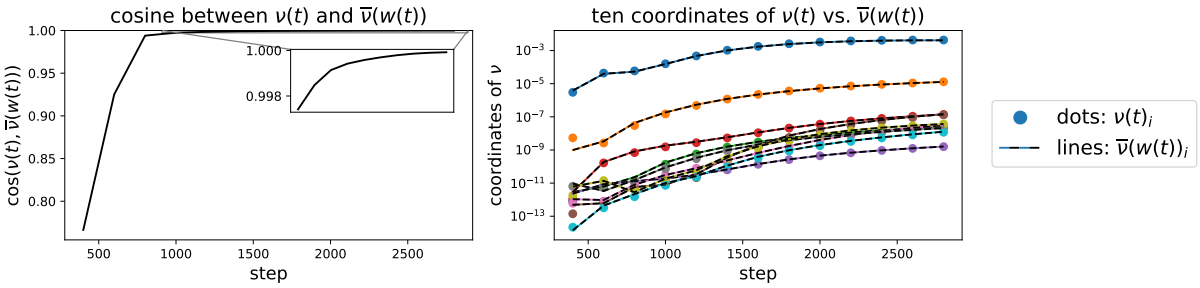


Figure 20: **The EMA ν converges to its stationary value.** While running the RMSProp central flow, we compare the actual EMA $\nu(t)$ to its stationary value $\overline{\nu}(w(t))$ w.r.t the current weights $w(t)$. On the left, we plot the cosine similarity between $\nu(t)$ and $\overline{\nu}(w(t))$; on the right, we compare ten individual coordinates (colors), spaced uniformly throughout the network. The plots begin when training enters EOS, just before step 500. Observe that after a bit of time, the cosine similarity between $\nu(t)$ and $\overline{\nu}(w(t))$ reaches high values (near 1), and the individual coordinates coincide as well. This figure depicts the same setting as Figure 17; see Figures 42(a) to 43(b) for more settings.

Interpreting the stationary preconditioner In Proposition 5, we show that this stationary preconditioner $\overline{P}(w) := \mathrm{diag}(\sqrt{\overline{\nu}(w)}/\eta)$ is, remarkably, the optimal solution to a convex optimization problem over preconditioners:

$$\overline{P}(w) \quad := \quad \underset{P \text{ diagonal, } P \succeq 0}{\arg\min} \quad \mathrm{tr}(P) + \frac{1}{\eta^2} \underbrace{\|\nabla L(w)\|_{P^{-1}}^2}_{\text{optimization speed}} \quad \text{such that} \quad \underbrace{H(w) \preceq 2P}_{\text{local stability}}. \tag{35}$$

That is, **RMSProp implicitly solves the convex program eq. (35) to compute its preconditioner**.[53] This is the precise sense in which RMSProp "adapts" its preconditioner to the local loss landscape.

We can now understand RMSProp's preconditioning strategy by interpreting the optimization problem eq. (35). The constraint $H(w) \preceq 2P$ is equivalent to $S^{\text{eff}} \leq 2$ and hence stipulates that the preconditioner P should keep RMSProp locally stable. The first term of the objective, $\mathrm{tr}(P)$, is the sum of the inverse effective step sizes. If this were the only term in the objective, RMSProp's preconditioning strategy could be simply summarized as maximizing the *harmonic mean* of the effective step sizes while maintaining local stability — a sensible preconditioning strategy. Indeed, consider a variant of eq. (35) with only the first term:

$$\hat{P}(w) := \underset{P \text{ diagonal, } P \succeq 0}{\arg\min} \quad \mathrm{tr}(P) \quad \text{such that} \quad H(w) \preceq 2P. \tag{36}$$

Figure 21 demonstrates that this preconditioner is a substantial improvement over vanilla gradient descent. (We describe in Appendix A.4.1 how we numerically solve eq. (35) and eq. (36).) Interestingly, if the diagonal constraint in eq. (36) were removed, and if $H(w)$ were PSD, then the optimization problem eq. (36) would have the closed-form solution $\hat{P}(w) = \frac{1}{2}H(w)$. That is, the preconditioner P would be a scaling of the Hessian, and preconditioned gradient descent would move in the same direction as Newton's method.[54]

However, matters are complicated by the presence of the second term in the eq. (35) objective. The quantity $\|\nabla L(w)\|_{P^{-1}}^2$ is the instantaneous rate of loss decrease under preconditioned gradient flow with preconditioner P. *Minimizing* this term necessarily acts to *slow down* optimization.[55] Indeed, Figure 21 shows that the stationary preconditioner eq. (35) underperforms the variant eq. (36) with only the first term.

Since the second term in eq. (35) is proportional to $\frac{1}{\eta^2}$, its influence diminishes as the learning rate hyperparameter η grows. Indeed, it can be seen in Figure 21 that the performance of the stationary preconditioner tends closer to

[53] Interestingly, this SDP is the dual to the max-cut SDP: $\max_{\Sigma \succeq 0} \langle \Sigma, H \rangle$ such that $\Sigma_{ii} = 1$ for all i. Thus, this preconditioning strategy could be described as solving the max-cut SDP with the Hessian as the weight matrix, and using the resulting dual variable as its preconditioner.

[54] Even if $H(w)$ were not PSD, a similar point would hold: the optimization problem eq. (36) would have the closed-form solution $\hat{P} = \frac{1}{2}\Pi_{\mathbb{S}_+} H(w)$, where $\Pi_{\mathbb{S}_+}$ denotes projection onto the cone of positive semidefinite matrices.

[55] For any w, the optimization speed $\|\nabla L(w)\|_{P^{-1}}^2$ must necessarily be smaller (worse) for eq. (36) than for eq. (35).

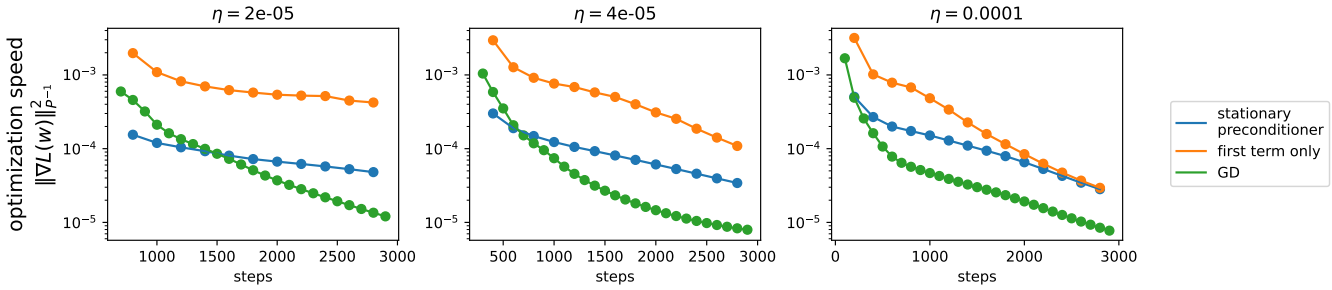


Figure 21: **Optimization speeds for various preconditioners.** Along the RMSProp central flow for various learning rates (columns), we assess the efficacy of three different preconditioners P: the RMSProp stationary preconditioner eq. (35), in blue; a variant eq. (36) with only the first term, in orange; and the preconditioner corresponding to vanilla gradient descent with the largest locally stable learning rate, i.e. $P^{-1} = (2/S(w))\, I$, in green. We assess each preconditioner P by reporting $\|\nabla L(w)\|^2_{P^{-1}} := \nabla L(w)^T P^{-1} \nabla L(w)$, the instantaneous rate of loss decrease under preconditioned gradient flow with preconditioner P. Observe that the "first term only" preconditioner (orange) is much better than the vanilla GD (green) preconditioner, and is also better than the actual stationary preconditioner (blue). The actual stationary preconditioner (blue) is usually better than vanilla GD (green), but not always, especially when η is smaller. See Figures 45(a) and 45(b) for more experimental settings.

that of eq. (36) as the learning rate hyperparameter η is made larger. In the limit of large η, the second term vanishes entirely, and the stationary preconditioner reduces completely to eq. (36). Interestingly, in this limit, the stationary preconditioner ceases to depend on η: for example, doubling η will cause $\overline{\nu}$ to quadruple in scale (due to larger oscillations), while keeping the effective step sizes $\eta/\sqrt{\overline{\nu}}$ unchanged. This parallels the situation for Scalar RMSProp in Section 4, where the effective step size at EOS was $2/S(w)$, independent of η.

The stationary flow Substituting $\overline{P}$ into the central flow, we can obtain a *stationary flow* over w alone:

$$\frac{dw}{dt} = -\underbrace{\overline{P}(w)^{-1}}_{\substack{\text{stationary} \\ \text{preconditioner}}} \Big[\nabla L(w) + \underbrace{\tfrac{1}{2}\nabla_w \langle \Sigma, H(w) \rangle}_{\text{implicit curvature penalty}} \Big]. \tag{37}$$

where $\Sigma = \Sigma(w; \eta; \beta_2)$ is defined as the solution to a certain semidefinite complementarity problem (Appendix A.4.1, Definition 10). This model assumes that the ν dynamics (preconditioner adaptation) happen *infinitely fast* relative to the w dynamics (optimization), so that we can treat the preconditioner P as always being fixed at its current stationary value $\overline{P}(w)$ (eq. 35). The appeal of this characterization is that it eliminates ν from the picture entirely, and expresses the time-averaged dynamics of RMSProp as a closed system in w alone.[56] Namely, it suggests that the time-averaged trajectory of RMSProp is equivalent to that of the following simpler-to-understand algorithm:

> At each iteration, compute the preconditioner $\overline{P}(w)$ using eq. (35) and then take a preconditioned gradient step using this preconditioner on a curvature-penalized objective.

Empirically, we find that the stationary flow eq. (37) is often a reasonable model for the RMSProp trajectory. For example, Figures 46(a) and 46(b) show that the stationary flow can accurately predict the instantaneous rate of loss decrease at various points along the central flow trajectory, even though it only has access to $w(t)$ and not $\nu(t)$. Meanwhile, Figures 47(a) and 47(b) show that the stationary flow can tolerably predict the trajectory of RMSProp over moderate timescales, although we note that its accuracy is not as high as the full central flow.[57]

[55] In this figure, for RMSProp, we show the train loss at the second-order midpoints between iterates (see Appendix B.1).
[56] This style of argument is referred to as "adiabatic elimination" in physics.
[57] As one might expect, the stationary flow tends to only be an accurate model for RMSProp once ν has reached stationarity.

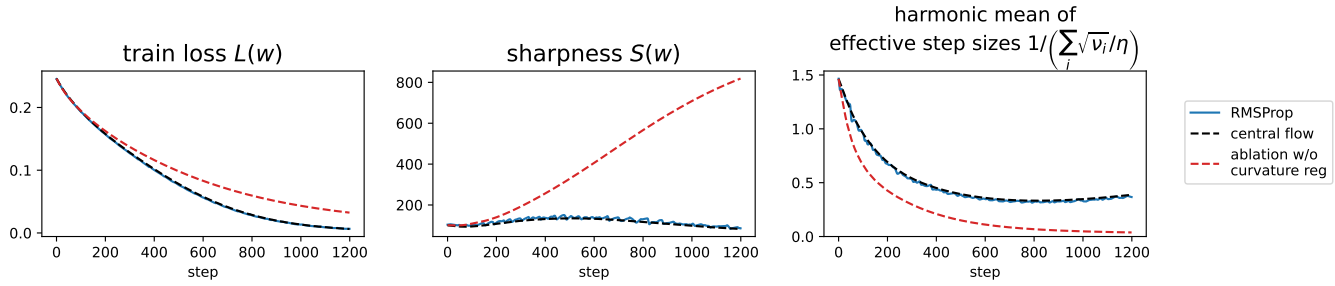


Figure 22: **Implicit curvature regularization accelerates optimization for RMSProp**. Starting from the same initial point, we compare RMSProp (blue) and its central flow (black) to an ablated flow where the implicit curvature regularization is disabled (red). Relative to this ablated flow, the central flow takes a lower-curvature trajectory (middle), in which it takes larger steps (right) and optimizes faster (left). The setting is the same as Figure 17.[58]

5.3.2 Acceleration via regularization

As with Scalar RMSProp, we find that RMSProp's implicit curvature regularization enables it to optimize faster. In Figure 22, we show that when the curvature regularization is disabled, the RMSProp central flow navigates into increasingly sharp regions, where it takes smaller steps, and optimizes slower (see Figure 41 for more settings).[59]

Establishing this claim theoretically is more difficult for RMSProp than Scalar RMSProp.[60] However, in the limit of large η and small β_2, it can be argued (Appendix A.4.1) that the stationary flow eq. (37) reduces to:

$$\frac{dw}{dt} = -\hat{P}(w)^{-1}\left[\nabla L(w) + \frac{\eta^2}{4}\nabla \operatorname{tr} \hat{P}(w)\right], \tag{38}$$

where $\hat{P}(w)$ was defined in eq. (36). This model says that RMSProp implicitly picks the diagonal preconditioner with minimal trace (equivalently, the preconditioner P where the effective learning rates P^{-1} have maximal harmonic mean), and also implicitly moves in a direction in which the trace of this preconditioner will become even smaller. The strength of the latter effect is controlled by η, and in fact, this is the only means by which the learning rate hyperparameter η affects the trajectory, since the preconditioner $\hat{P}(w)$ is independent of η. Thus, as with Scalar RMSProp, larger learning rates translate to larger steps, but only via this indirect mechanism.

6 Experiments

The goal of our experiments is to establish that each central flow accurately approximates the trajectory of its corresponding optimizer in a variety of deep learning settings, and to understand the circumstances under which this approximation breaks down. Because it is computationally costly to discretize central flows, we experiment on small-scale networks and datasets. Note that there is no evidence that scale itself fundamentally affects the dynamics of optimization in deep learning; for example, EOS dynamics have been observed at both smaller (e.g. CIFAR-10) and larger (e.g. ImageNet or WMT) scales, without noticeable differences. Therefore, we expect that the central flow approximation would similarly hold true at larger scales, if such experiments were computationally feasible.

We emphasize that the central flow is a theoretical tool for understanding optimizer behavior, not a practical optimization method. In practice, maintaining an exponential moving average of the iterates (e.g., Morales-Brotons et al., 2024) is likely a computational feasible way to estimate the optimizer's time-averaged trajectory.

Architectures We experiment on a diverse set of six architectures: a convolutional neural network (CNN), a ResNet (He et al., 2016), a Vision Transformer (ViT) (Dosovitskiy et al., 2021), an LSTM (Hochreiter and Schmidhuber, 1997),

[59] To run this ablated flow, we manually set $\nabla H(w) = 0$ both in the expression for β and in the expression for $\frac{dw}{dt}$ (see Appendix A.4).

[60] Partly, the difficulty of analysis is due to the independent ν dynamics. However, this analysis is also not easy under the stationary flow, because the second term in eq. (35) causes the effective step sizes to depend not just on the current Hessian but also on the current gradient.

a (sequence) Transformer (Vaswani et al., 2017), and a Mamba sequence model (Gu and Dao, 2024). Architectural details can be found in Appendix B.2.

Datasets We test the vision architectures (CNN, ResNet, ViT) on a subset of CIFAR-10 (Krizhevsky, 2009), and the sequence architectures (LSTM, Transformer, Mamba) on a synthetic sorting task (Karpathy, 2020). Further details on these datasets can be found in Appendix B.3. For each architecture and each dataset, we test both cross-entropy loss and MSE loss. As discussed below, the central flow tends to be somewhat more accurate with MSE loss.

Implementation Discretizing the central flows is somewhat nontrivial, as the flows are non-smooth at points where there is a change in the number of unstable eigenvalues (e.g. going from 0 to 1). We describe our solution in Appendix A.6. The time complexity of each discretization step scales quadratically with the number of eigenvalues that are at the edge of stability. Most of the computational cost arises from the need to continually re-estimate the top eigenvectors and eigenvalues of the (effective) Hessian, and to compute the necessary third derivatives (gradients of these eigenvalues). Full implementation details can be found in Appendix B.1.

Our code can be found at: `https://github.com/locuslab/central_flows`.

6.1 Experimental Results

To assess the accuracy of the central flow approximation, we run both the discrete optimizer and the central flow simultaneously, starting from the same initialization. As a baseline, we also run the corresponding stable flow (e.g. for gradient descent, the gradient flow), which we expect to poorly approximate the discrete optimizer when the latter is at the edge of stability.

Our full experimental results, which can be found in Appendix E, make clear that the central flow can accurately approximate the long-term optimization trajectory in a variety of deep learning settings. We find that the weight-space distance between the discrete optimizer and the central flow generally stays small over time, and is much smaller than the distance between the discrete optimizer and the stable flow baseline. Meanwhile, the network's predictions under the central flow generally match those of the discrete optimizer, whereas the stable flow takes a different path through function space. The central flow can also accurately predict the time-averaged train loss curve and squared gradient norm curve, as well as the covariance of the discrete optimizer's oscillations around the central flow.

That said, the central flow approximation can break down in certain circumstances, which we now describe. An interesting direction for future work would be to rigorously characterize the conditions under which the central flow does or does not approximate the real optimizer trajectory.

Sufficiently large learning rates For all three optimizers studied in this paper, we reliably observe that as the learning rate hyperparameter is made increasingly large, the real optimization trajectory tends to deviate more from the central flow, as illustrated in Figure 23. (As an extreme example, even if the optimizer is initialized stably, very large learning rates sometimes cause the real optimizer to explosively diverge in the middle of training, whereas this never happens under the central flow.) We do not know whether some corrected version of the central flow would be more successful at capturing the real optimization trajectory in these scenarios, or whether the real trajectory is simply too chaotic to be captured by any flow.

Higher-order terms Sometimes, the local curvature is not well-modeled by the cubic Taylor approximation, as is assumed by our theory. This leads the central flow to mispredict $\Sigma(t)$, causing error to accumulate over the long run. We elaborate on this failure mode in Appendix C.2.

Smoothness of architecture The smoothness of the architecture seems to affect the accuracy of the central flow approximation; non-smooth components such as ReLU or max pooling often cause the quality of the approximation to break down. For example, in Figure 38, we show that as a network's activation function is interpolated from GeLU (smooth) to ReLU (non-smooth), the accuracy of the central flow approximation degrades, both in weight space and function space. Note that it is not clear how best to precisely quantify "smoothness" in this context.

Large spikes When the EOS dynamics lead to extremely large spikes (e.g. in the gradient norm), we have found

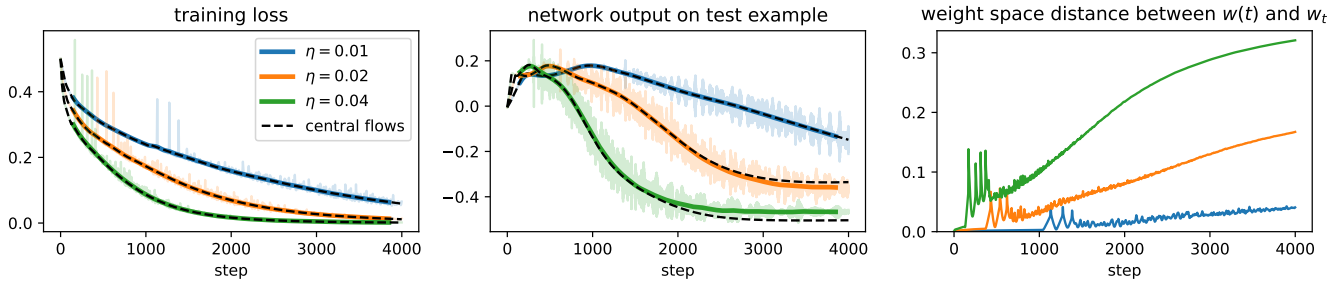


Figure 23: **Central flow approximation is less accurate at larger learning rates**. We run both gradient descent and its central flow at three learning rates (colors). The larger the learning rate, the faster the growth in the accumulated approximation error (right). Indeed, at larger learning rates, the network's output on an arbitrary test example can be visually seen to be slightly different between the central flow and gradient descent (middle). Nevertheless, the central flow approximation is still accurate enough here to accurately capture the train loss curves (left). *Details*: a CNN is trained on CIFAR-10 using MSE loss.

such spikes can cause the real trajectory to deviate from the central flow, as illustrated in Figure 39. This may be related to the large learning rate issue described above.

Interactions with loss criterion We have empirically found that the higher-order terms issue and the large spike issue are more common with cross-entropy loss than with mean squared error loss (although we do not have a satisfactory explanation for these observations). Consequently, the central flow approximation is often more accurate under the MSE loss than the cross-entropy loss.

Overall, despite these limitations, we argue that the central flow describes the behavior of the corresponding optimizer "to a first approximation." Even in cases where the central flow is not a perfect quantitative match to the discrete trajectory, it may still capture the important qualitative trends.

7 Discussion

7.1 Modeling decisions

Deterministic setting Our analysis is restricted to the simple setting of deterministic (i.e. full-batch) training, whereas practical deep learning generally involves minibatch training. We study the full-batch setting because, as the simplest special case, understanding full-batch training is a necessary prerequisite for understanding minibatch training. However, we also believe that understanding full-batch training might suffice for some practical purposes, such as designing optimizers. For example, Kunstner et al. (2024) showed that the advantage of adaptive methods over SGD grows larger with larger batch sizes, suggesting that the relevant algorithmic principles can be best understood in the deterministic setting.

An interesting direction for future research is to try to extend our central flows methodology to the stochastic setting. Like deterministic optimizers, stochastic optimizers are known to implicitly regularize the curvature along their trajectories, and in fact this effect is *stronger* in the stochastic setting (Keskar et al., 2017; Jastrzębski et al., 2020, 2021; Andreyev and Beneventano, 2024). However, extending the central flows methodology to the stochastic setting may be nontrivial; due to the randomness, it is not clear whether there exists a deterministic differential equation around which SGD oscillates. An interesting question is whether there exists a differential equation that can predict derived metrics such as network predictions or training loss curves, even if it cannot model the weight-space trajectory of SGD. Finally, while our analysis in this paper sheds light on adaptive optimizers in the deterministic setting, these optimizers could exhibit substantially different behavior in the stochastic setting.

Black-box model of the loss Our analysis treats the loss function as a black box, and never uses that the optimization

problem at hand involves training a neural network. The advantage of this approach is its generality: we expect our analysis to apply to generic deep learning architectures and learning problems, including those that do not yet exist. The disadvantage, however, is that the predictions made by our theory are at the abstraction level of the *loss landscape*, and would need to be further translated in order to make concrete claims about the network architecture or learning problem. For example, our theory tells us that the learning rate hyperparameter modulates the strength of an implicit sharpness penalty, but does not tell us how this sharpness penalty affects learning. Nor does our theory shed light on how different layers of the neural network are mechanistically implicated in progressive sharpening or sharpness reduction.

On the one hand, the loss landscape level of abstraction is in some sense "natural" — the overall path that optimizers follow really does intrinsically depend on the (effective) sharpness. But on the other hand, understanding many important aspects of optimization in deep learning will likely require cracking open the black box a bit more.

Goals of our analysis The goal of our analysis is different from the usual goals of optimization theory. Typically, the goal of an optimization analyses is to characterize the optimizer's rate of convergence to a minimum or stationary point. These can either be global rates that hold from any initialization, or local rates that hold in the immediate vicinity of a minimum. However, both of these goals have shortcomings as regards optimization in deep learning. As for local rates, when training a neural network, the vast majority of the optimization process cannot be usefully regarded as being within the vicinity of the final minimum, yet is of interest to the practitioner. As for global rates, is not currently possible to meaningfully characterize the global rate of convergence of any optimization algorithm in realistic deep learning settings.

Thus, our paper instead pursues a different aim: characterizing the *local dynamics* of the optimizer *throughout training*. As we have shown, these local dynamics are important; they are more mathematically interesting than may have been assumed (even vanilla gradient descent gives rise to rich dynamics); and they are empirically consistent across different deep learning settings, which suggests that general theory is feasible.

7.2 Takeaways from our analysis

The unreasonable effectiveness of time-averaging Prior works on EOS show that it is challenging to analyze the oscillatory EOS dynamics in fine-grained detail. Our work shows that, perhaps surprisingly, simple heuristics allow us to analyze the *time-averaged* trajectory with excellent numerical accuracy. Interestingly, the success of this time-averaging approach seems to imply that the oscillations only affect the macroscopic trajectory in an ergodic sense, i.e. via their *covariance* rather than via their fine-grained details. An promising direction for future work is to identify realistic conditions under which our heuristic time-averaging arguments can be made rigorous.

Necessity of third-order Taylor expansions While optimization theory generally relies on second-order Taylor expansions of the loss, Damian et al. (2023) showed that a *third-order* Taylor expansion is necessary for understanding the convergence of gradient descent; such a Taylor expansion reveals that oscillations implicitly trigger curvature reduction, a form of negative feedback which stabilizes optimization. In this work, we have shown that a third-order Taylor expansion is similarly necessary for understanding the *acceleration via mechanism* which underlies the success of adaptive optimizers. Thus, our work further underscores the necessity of a third-order Taylor expansion when analyzing optimization in deep learning.

Oscillatory first-order methods are implicitly second-order methods Over the last decade, optimizers that explicitly use Hessian information have failed to outperform first-order adaptive optimizers which employ only gradient information. Our work demystifies this observation. We have shown that that when first-order optimizers oscillate, they implicitly leverage second order information. Thus, even though RMSProp is a first-order optimizer, it implicitly employs a second-order preconditioning strategy, detailed in Section 5.3.1. Further, this preconditioning strategy is efficient, requiring no more gradient queries than gradient descent does. An exciting direction for future work is to *intentionally* design first-order adaptive methods with such implicit preconditioners in mind.

Adapting to curvature is not enough Traditional optimization theory views the curvature of the loss as a pre-existing feature of the optimization problem, and views the job of an optimizer as *adapting* to this pre-existing

curvature. We have shown that the adaptive optimizers that we study do not merely passively adapt to the curvature; they also actively *shape* the curvature along their trajectory, by steering away from high-curvature regions where they would need to take small steps. Further, we have shown that this effect is crucial for their optimization efficacy. Thus, our work suggests that *acceleration via regularization* is a vital design principle for adaptive optimizers.

8 Conclusion

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

A. Agarwala, F. Pedregosa, and J. Pennington. Second-order regression models exhibit progressive sharpening to the edge of stability. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

K. Ahn, S. Bubeck, S. Chewi, Y. T. Lee, F. Suarez, and Y. Zhang. Learning threshold neurons via edge of stability. *Advances in Neural Information Processing Systems*, 36, 2024.

A. Andreyev and P. Beneventano. Edge of stochastic stability: Revisiting the edge of stability for sgd. *arXiv preprint arXiv:2412.20553*, 2024.

S. Arora, Z. Li, and A. Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pages 948–1024. PMLR, 2022.

Z. Bai, Z. Zhou, J. Zhao, X. Li, Z. Li, F. Xiong, H. Yang, Y. Zhang, and Z.-Q. J. Xu. Adaptive preconditioners trigger loss spikes in adam. *arXiv preprint arXiv:2506.04805*, 2025.

D. Barrett and B. Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=3q5IqUrkcF`.

L. Beyer, X. Zhai, and A. Kolesnikov. Better plain vit baselines for imagenet-1k. *arXiv preprint arXiv:2205.01580*, 2022.

L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

G. Blanc, N. Gupta, G. Valiant, and P. Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Annual Conference Computational Learning Theory*, 2019. URL `https://api.semanticscholar.org/CorpusID:125944013`.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

S. Burer and R. D. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005.

M. D. Cattaneo, J. M. Klusowski, and B. Shigida. On the implicit bias of Adam. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

C. Chen, L. Shen, F. Zou, and W. Liu. Towards practical adam: Non-convexity, convergence theory, and mini-batch acceleration. *Journal of Machine Learning Research*, 23(229):1–47, 2022.

L. Chen and J. Bruna. Beyond the edge of stability via two-step gradient updates. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1x-x309tm.

Z. Chen, Z. Yuan, J. Yi, B. Zhou, E. Chen, and T. Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=Syx5V2CcFm.

J. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

J. M. Cohen, B. Ghorbani, S. Krishnan, N. Agarwal, S. Medapati, M. Badura, D. Suo, D. Cardoze, Z. Nado, G. E. Dahl, and J. Gilmer. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

E. M. Compagnoni, L. Biggio, A. Orvieto, F. N. Proske, H. Kersting, and A. Lucchi. An sde for modeling sam: Theory and insights. In *International Conference on Machine Learning*, pages 25209–25253. PMLR, 2023.

E. M. Compagnoni, T. Liu, R. Islamov, F. N. Proske, A. Orvieto, and A. Lucchi. Adaptive methods through the lens of SDEs: Theoretical insights on the role of noise. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ww3CLRhF1v.

B. Cornet. Existence of slow solutions for a class of differential inclusions. *Journal of Mathematical Analysis and Applications*, 96(1):130–147, Oct. 1983. ISSN 0022-247X. doi:10.1016/0022-247X(83)90032-X.

M. Crawshaw, M. Liu, F. Orabona, W. Zhang, and Z. Zhuang. Robustness to unbounded smoothness of generalized signsgd. *Advances in Neural Information Processing Systems*, 35:9955–9968, 2022.

A. Damian, T. Ma, and J. D. Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

A. Damian, E. Nichani, and J. D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nhKHA59gXz.

Y. Dauphin, A. Agarwala, and H. Mobahi. How hessian structure explains mysteries in sharpness regularization. *Advances in Neural Information Processing Systems*, 37, 2024.

M. K. de Carli Silva and L. Tunçel. Strict complementarity in maxcut sdp, 2018. URL https://arxiv.org/abs/1806.01173.

A. Défossez, L. Bottou, F. Bach, and N. Usunier. A simple convergence proof of adam and adagrad. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=ZPQhzTSWA7.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

O. Elkabetz and N. Cohen. Continuous vs. discrete optimization of deep neural networks. *Advances in Neural Information Processing Systems*, 34:4947–4960, 2021.

M. Even, S. Pesme, S. Gunasekar, and N. Flammarion. (s)gd over diagonal linear networks: implicit bias, large stepsizes and edge of stability. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, 2024.

J. Geiping, M. Goldblum, P. Pope, M. Moeller, and T. Goldstein. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZBESeIUB5k.

A. Ghosh, H. Lyu, X. Zhang, and R. Wang. Implicit regularization in heavy-ball momentum accelerated stochastic gradient descent. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ZzdBhtEH9yB.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang. A novel convergence analysis for algorithms of the adam family. *arXiv preprint arXiv:2112.03459*, 2021.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Y. Hong and J. Lin. On convergence of adam for stochastic optimization under relaxed assumptions. *Advances in Neural Information Processing Systems*, 37:10827–10877, 2024.

F. Hübler, J. Yang, X. Li, and N. He. Parameter-agnostic optimization under relaxed smoothness. In *International Conference on Artificial Intelligence and Statistics*, pages 4861–4869. PMLR, 2024.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

S. Jastrzębski, Z. Kenton, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkgEaj05t7.

S. Jastrzębski, M. Szymczak, S. Fort, D. Arpit, J. Tabor, K. Cho*, and K. Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

S. Jastrzębski, D. Arpit, O. Astrand, G. B. Kerg, H. Wang, C. Xiong, R. Socher, K. Cho, and K. J. Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *International Conference on Machine Learning*, pages 4772–4784. PMLR, 2021.

A. Karpathy. mingpt - demo.ipynb. https://github.com/karpathy/minGPT/blob/master/demo.ipynb, 2020.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

A. Khaled, K. Mishchenko, and C. Jin. Dowg unleashed: An efficient universal parameter-free gradient descent method. *Advances in Neural Information Processing Systems*, 36:6748–6769, 2023.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.

I. Kreisler, M. S. Nacson, D. Soudry, and Y. Carmon. Gradient descent monotonically decreases the sharpness of gradient flow solutions in scalar networks and beyond. In *International Conference on Machine Learning*, pages 17684–17744. PMLR, 2023.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

F. Kunstner, P. Hennig, and L. Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in Neural Information Processing Systems*, 32, 2019.

F. Kunstner, R. Yadav, A. Milligan, M. Schmidt, and A. Bietti. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. In *Neural Information Processing Systems*, 2024.

A. Lewkowycz, Y. Bahri, E. Dyer, J. Sohl-Dickstein, and G. Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

H. Li and Z. Lin. On the O($\sqrt{d}/t^{1/4}$) convergence rate of RMSProp and its momentum extension measured by l_1 norm: Better dependence on the dimension. *arXiv preprint arXiv:2402.00389*, 2024.

H. Li, A. Rakhlin, and A. Jadbabaie. Convergence of adam under relaxed assumptions. *Advances in Neural Information Processing Systems*, 36, 2024.

Q. Li, C. Tai, and W. E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

X. Li, Z.-Q. J. Xu, and Z. Zhang. Loss spike in training neural networks. *arXiv preprint arXiv:2305.12133*, 2023.

Y. Li, C. Wei, and T. Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Z. Li, S. Malladi, and S. Arora. On the validity of modeling SGD with stochastic differential equations (SDEs). In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=goEdyJ_nVQI.

Z. Li, T. Wang, and S. Arora. What happens after SGD reaches zero loss? –a mathematical framework. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=siCt4xZn5Ve.

Z. Li, Z. Wang, and J. Li. Analyzing sharpness along gd trajectory: Progressive sharpening and edge of stability. In *Neural Information Processing Systems*, 2022b.

Y. Liu, Z. Liu, and J. Gore. Focus: First order concentrated updating scheme. *arXiv preprint arXiv:2501.12243*, 2025a.

Z. Liu, Y. Liu, J. Gore, and M. Tegmark. Neural thermodynamic laws for large language model training, 2025b. URL https://arxiv.org/abs/2505.10559.

LucidRains. Vision transformer - pytorch. https://github.com/lucidrains/vit-pytorch, 2024.

K. Lyu, Z. Li, and S. Arora. Understanding the generalization benefit of normalization layers: Sharpness reduction. *Advances in Neural Information Processing Systems*, 35:34689–34708, 2022.

C. Ma, L. Wu, and E. Weinan. A qualitative study of the dynamic behavior for adaptive gradient algorithms. In *Mathematical and Scientific Machine Learning*, pages 671–692. PMLR, 2022.

S. Malladi, K. Lyu, A. Panigrahi, and S. Arora. On the sdes and scaling rules for adaptive gradient algorithms. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 7697–7711. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/32ac710102f0620d0f28d5d05a44fe08-Paper-Conference.pdf.

J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21 (146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

A. Mishkin, A. Khaled, Y. Wang, A. Defazio, and R. M. Gower. Directional smoothness and gradient methods: Convergence and adaptivity. *Advances in Neural Information Processing Systems*, 2024.

D. Morales-Brotons, T. Vogels, and H. Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=2M9CUnYnBA.

J.-J. Moreau. Décomposition orthogonale d'un espace hilbertien selon deux cones mutuellement polaires. *Comptes Rendus de l'Académie des Sciences*, 255:238–240, 1962.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

F. Orabona. Neural networks (maybe) evolved to make adam the best optimizer. https://parameterfree.com/2020/12/06/neural-network-maybe-evolved-to-make-adam-the-best-optimizer/, 2020. Accessed: October 17, 2024.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

M. Rosca, Y. Wu, C. Qin, and B. Dherin. On a continuous time model of gradient descent dynamics and instability in deep learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=EYrRzKPinA.

V. Roulet, A. Agarwala, J.-B. Grill, G. M. Swirszcz, M. Blondel, and F. Pedregosa. Stepping on the edge: Curvature aware learning rate tuners. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=SEflLHIhhJ.

M. Sandler, A. Zhmoginov, M. Vladymyrov, and N. Miller. Training trajectories, mini-batch losses and the curious role of the learning rate, 2023. URL https://arxiv.org/abs/2301.02312.

N. Shi, D. Li, M. Hong, and R. Sun. RMSprop converges with proper hyper-parameter. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3UDSdyIcBDA.

S. L. Smith, B. Dherin, D. Barrett, and S. De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

M. Song and C. Yun. Trajectory alignment: Understanding the edge of stability phenomenon via bifurcation theory. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=PnJaA0A8Lr.

D. E. Stewart. *Dynamics with Inequalities*. Society for Industrial and Applied Mathematics, 2011. doi:10.1137/1.9781611970715. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611970715.

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.

A. Torres-Leguet. mamba.py. https://github.com/alxndrTL/mamba.py, 2024.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

B. Wang, J. Fu, H. Zhang, N. Zheng, and W. Chen. Closing the gap between the upper bound and lower bound of adam's iteration complexity. *Advances in Neural Information Processing Systems*, 36, 2024a.

B. Wang, H. Zhang, Q. Meng, R. Sun, Z.-M. Ma, and W. Chen. On the convergence of adam under non-uniform smoothness: Separability from sgdm and beyond. *arXiv preprint arXiv:2403.15146*, 2024b.

B. Wang, Y. Zhang, H. Zhang, Q. Meng, R. Sun, Z.-M. Ma, T.-Y. Liu, Z.-Q. Luo, and W. Chen. Provable adaptivity of adam under non-uniform smoothness. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2024c. ISBN 9798400704901.

M. Wang, J. Wang, H. He, Z. Wang, G. Huang, F. Xiong, Z. li, W. E, and L. Wu. Improving generalization and convergence by enhancing implicit regularization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d. URL https://openreview.net/forum?id=cjM2bhLOiC.

K. Wen, Z. Li, J. S. Wang, D. L. W. Hall, P. Liang, and T. Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=m51BgoqvbP.

J. Wu, V. Braverman, and J. D. Lee. Implicit bias of gradient descent for logistic regression at the edge of stability. *Advances in Neural Information Processing Systems*, 36, 2024.

L. Wu, C. Ma, and W. E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/6651526b6fb8f29a00507de6a49ce30f-Paper.pdf.

Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

C. Xing, D. Arpit, C. Tsirigotis, and Y. Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

J. Yang, X. Li, I. Fatkhullin, and N. He. Two sides of one coin: the limits of untuned sgd and the power of adaptive methods. *Advances in Neural Information Processing Systems*, 36, 2024.

M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Q. Zhang, Y. Zhou, and S. Zou. Convergence guarantees for RMSProp and adam in generalized-smooth non-convex optimization with affine noise variance. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=QIzRdjIWnS.

Y. Zhang, C. Chen, N. Shi, R. Sun, and Z.-Q. Luo. Adam can converge without any modification on update rules. *Advances in Neural Information Processing Systems*, 35:28386–28399, 2022.

X. Zhu, Z. Wang, X. Wang, M. Zhou, and R. Ge. Understanding edge-of-stability training dynamics with a minimalist example. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=p7EagBsMAEO.

F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11127–11135, 2019.

Contents

A Central Flow Derivations

In this appendix, we derive central flows for the three optimizers studied in this paper: gradient descent (Appendix A.2), Scalar RMSProp (Appendix A.3), and RMSProp (Appendix A.4). We reiterate that these derivations rely on informal mathematical reasoning; our claim that the central flow accurately matches the optimizer trajectory is ultimately supported *empirically* by the experiments described in Section 6. An interesting direction for future work is to prove that, under realistic conditions, the discrete optimizer follows the central flow.

A.1 Preliminaries

A.1.1 Notation

We use $L(w)$ to denote the training objective, a function of weights $w \in \mathbb{R}^d$. We will assume that L is three-times differentiable. We will frequently use $H(w)$ as a shorthand for $\nabla^2 L(w)$, the Hessian matrix at w.

We use $\langle A, B \rangle$ to denote the Frobenius inner product $\operatorname{tr}\left(A^\top B\right)$ between two matrices A and B, equivalent to flattening the matrices into vectors and taking the dot product. We use $\ker$ and span to denote the kernel and span of a matrix, and $\dim$ to denote the dimension of a vector space.

We use $\operatorname{Sym}(\mathbb{R}^d)$ to denote the set of $d \times d$ symmetric matrices. For a k-dimensional subspace $\mathcal{U} \subseteq \mathbb{R}^d$, we use $\operatorname{Sym}(\mathcal{U})$ to denote the set of $d \times d$ symmetric matrices whose span is contained within $\mathcal{U}$. Equivalently, this is the set of matrices that can be written as $UXU^\top$, where $U \in \mathbb{R}^{d \times k}$ is a basis for $\mathcal{U}$ and $X \in \operatorname{Sym}(\mathbb{R}^k)$.

For a subspace $\mathcal{U} \subseteq \mathbb{R}^d$ and a matrix $A \in \mathbb{R}^{d \times d}$, we use $A \big|_{\mathcal{U}}$ to denote the *restriction* of A to $\mathcal{U}$, that is,

$$A \big|_{\mathcal{U}} := \Pi_{\mathcal{U}} \, A \, \Pi_{\mathcal{U}}, \tag{39}$$

where $\Pi_{\mathcal{U}}$ is the matrix that projects onto $\mathcal{U}$, e.g. $\Pi_{\mathcal{U}} = UU^\top$ for any orthogonal basis U of $\mathcal{U}$.

For a subspace $\mathcal{U} \subseteq \mathbb{R}^d$ and a matrix $A \in \operatorname{Sym}(\mathbb{R}^d)$, we write $A \succeq_{\mathcal{U}} 0$ to denote that A is positive semidefinite (PSD) over the subspace $\mathcal{U}$, i.e. $u^\top A u \geq 0 \; \forall u \in \mathcal{U}$, or equivalently, $A \big|_{\mathcal{U}} \succeq 0$. We analogously define $A \preceq_{\mathcal{U}} 0$ to mean that A is negative semidefinite (NSD) over $\mathcal{U}$.

We will need to work with higher-order tensors, though we will try our best to keep the tensor notation at a minimum. For an order-k tensor T and an order-s tensor X, with $s < k$, we define the contraction $T[X]$ as the order-$(k-s)$ tensor obtained by multiplying T and X componentwise along the last s coordinates of T and all coordinates of X, and then summing over those coordinates.

$$(T[X])_{i_1,\ldots,i_{k-s}} := \sum_{j_1,\ldots,j_s} T_{i_1,\ldots,i_{k-s},j_1,\ldots,j_s} \, X_{j_1,\ldots,j_s}.$$

We will also write $T[u_1, \ldots, u_s]$ for the result of contracting T sequentially with the vectors $u_1, \ldots, u_s$ one at a time, equivalent to $T[u_1 \otimes \ldots \otimes u_s]$. When T is fully symmetric, permuting $u_1, \ldots, u_s$ does not change the result.

It can be helpful to reshape a higher-order tensor into a matrix; this viewpoint lets us work with higher-order tensors while using the familiar language of linear algebra. For example, consider $\nabla^3 L(w)$, the order-3 tensor of third derivatives. By flattening together the first two dimensions, we could view this as a matrix of shape $\mathbb{R}^{d^2 \times d}$. In tensor product notation, we are identifying $\nabla^3 L$ as an element of $\mathbb{R}^{d^2} \otimes \mathbb{R}^d$. (We note that understanding tensor product notation is not necessary for understanding this paper.) One could similarly identify $\nabla^3 L$ as an element of $\mathbb{R}^{d \times d} \otimes \mathbb{R}^d$ or $\operatorname{Sym}(\mathbb{R}^d) \otimes \mathbb{R}^d$. These views naturally correspond to linear operators; for example, an element of the tensor product space $\operatorname{Sym}(\mathbb{R}^d) \otimes \mathbb{R}^d$ can be treated as a linear operator $\mathbb{R}^d \to \operatorname{Sym}(\mathbb{R}^d)$. We will switch freely among the full-tensor, tensor-product, and operator views as convenient.

In particular, we will use the notation $\nabla H(w) \in \operatorname{Sym}(\mathbb{R}^d) \otimes \mathbb{R}^d$ to denote the reshaping of $\nabla^3 L(w)$ that collects

together the first two indices (i.e. the "gradient of the Hessian"):

$$\nabla H(w)_{ij,p} := \nabla^3 L(w)_{ijp} = \frac{\partial H_{ij}(w)}{\partial w_p}.$$

Intuitively, for any direction $v \in \mathbb{R}^d$, the operator $\nabla H(w) : \mathbb{R}^d \to \mathrm{Sym}(\mathbb{R}^d)$ returns the directional derivative of the Hessian $H(w)$ when moving in the direction v:

$$[\nabla H(w)[v]]_{ij} = \sum_{p=1}^{d} \frac{\partial H_{ij}(w)}{\partial w_p} v_p.$$

Meanwhile, for any matrix $\Sigma \in \mathrm{Sym}(\mathbb{R}^d)$, the transpose operator $\nabla H(w)^\top : \mathrm{Sym}(\mathbb{R}^d) \to \mathbb{R}^d$ returns the gradient of the Σ-weighted Hessian $\langle \Sigma, H(w) \rangle$:

$$\left[\nabla H(w)^\top[\Sigma]\right]_p = \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial H_{ij}(w)}{\partial w_p}\Sigma_{ij} \implies \nabla H(w)^\top[\Sigma] = \nabla_w \left\langle \Sigma, H(w) \right\rangle.$$

For a vector space V, we abbreviate $V \otimes V$ as $V^{\otimes 2}$, e.g. we abbreviate $\mathrm{Sym}(\mathbb{R}^d) \otimes \mathrm{Sym}(\mathbb{R}^d)$ as $\mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$.

A.1.2 Third-order Taylor expansions

If $L : \mathbb{R}^d \to \mathbb{R}$ is k-times differentiable, then Taylor's theorem for the k-th order Taylor expansion of L is:

$$L(w+\delta) = \sum_{j=0}^{k} \frac{1}{j!} \nabla^j L(w)[\delta^{\otimes j}] + o(\|\delta\|^k).$$

In particular, the third-order Taylor expansion of L around any $w \in \mathbb{R}^d$ is given by:

$$L(w+\delta) = L(w) + \nabla L(w)[\delta] + \tfrac{1}{2}\nabla^2 L(w)[\delta,\delta] + \tfrac{1}{6}\nabla^3 L(w)[\delta,\delta,\delta] + o(\|\delta\|^3). \tag{40}$$

Likewise, the second-order Taylor expansion of the *gradient* ∇L around w is:

$$\nabla L(w+\delta) = \nabla L(w) + \nabla^2 L(w)[\delta] + \tfrac{1}{2}\nabla^3 L(w)[\delta,\delta] + o(\|\delta\|^2). \tag{41}$$

Since $\nabla^3 L(w)[\delta,\delta] = \nabla^3 L(w)[\delta\delta^\top]$, and recalling our H, ∇H notation from the preceding section, we can equivalently write this Taylor expansion in the form:

$$\nabla L(w+\delta) = \nabla L(w) + H(w)[\delta] + \tfrac{1}{2}\nabla H(w)^\top[\delta\delta^\top] + o(\|\delta\|^2). \tag{42}$$

This is the form that we will directly use in our central flow derivations.

A.1.3 Complementarity

A *complementarity relation* is a constraint on two non-negative variables which enforces that at least one of the variables is zero, i.e. both cannot be strictly positive. An example is the following set of three conditions over the two scalar-valued variables $x, y \in \mathbb{R}$:

$$x \geq 0, \quad y \geq 0, \quad xy = 0. \tag{43}$$

By convention, such a condition is often abbreviated using the shorthand notation:

$$0 \leq x \perp y \geq 0. \tag{44}$$

Complementarity relations can be extended to vectors and matrices. We will be particularly interested in the matrix case. For two symmetric matrices $X, Y \in \mathrm{Sym}(\mathbb{R}^d)$, consider the following complementarity relation:

$$X \succeq 0, \quad Y \succeq 0, \quad \langle X, Y \rangle = 0, \tag{45}$$

which we will often abbreviate using the shorthand:

$$0 \preceq X \perp Y \succeq 0. \tag{46}$$

This condition is equivalent to X, Y being PSD with orthogonal spans (Appendix A.5, Fact 2). It is also equivalent to X, Y being PSD with $\mathrm{span}\, X \subseteq \ker Y$ (Appendix A.5, Corollary 1).

A.1.4 Semidefinite complementarity problems

The *semidefinite complementarity problem* (SDCP) will be a recurring primitive throughout this work. An SDCP asks to find a matrix $\Sigma \in \mathrm{Sym}(\mathbb{R}^d)$ that is complementary to an affine function of itself. Namely, let $\alpha \in \mathrm{Sym}(\mathbb{R}^d)$ be a symmetric matrix and let $\beta \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$ be a tensor, viewed as a linear operator over symmetric matrices $\mathrm{Sym}(\mathbb{R}^d) \to \mathrm{Sym}(\mathbb{R}^d)$. The semidefinite complementarity problem is to find a matrix $\Sigma \in \mathrm{Sym}(\mathbb{R}^d)$ such that:

$$0 \preceq \Sigma \perp \alpha + \beta[\Sigma] \succeq 0. \tag{47}$$

This is a generalization of the well-studied linear complementarity problem from vectors in the non-negative orthant to matrices in the positive semidefinite cone.

Remark 1. It is easily verified that if $\beta^{-1}[-\alpha] \succeq 0$, then the linear inverse $\Sigma = \beta^{-1}[-\alpha]$ is a solution to the SDCP eq. (47). Interestingly, along the central flows, this will be the solution to the SDCP at almost all times.

Remark 2. In the scalar-valued case, where $\alpha, \beta \in \mathbb{R}$, one can verify by case-checking that the SDCP

$$0 \leq \sigma^2 \perp \alpha + \beta\sigma^2 \geq 0 \tag{48}$$

has the closed-form solution $\sigma^2 = \max(-\frac{\alpha}{\beta}, 0)$ provided that $\beta > 0$.

It will be useful to restrict the domain of an SDCP to an arbitrary subspace $\mathcal{U} \subseteq \mathbb{R}^d$. Recall that we use $\mathrm{Sym}(\mathcal{U})$ to denote the set of symmetric matrices whose span is contained within $\mathcal{U}$. We thus have the following definition:

Definition 1 (Semidefinite Complementarity Problem)**.** For a subspace $\mathcal{U} \subseteq \mathbb{R}^d$, matrix $\alpha \in \mathrm{Sym}(\mathcal{U})$, and tensor $\beta \in \mathrm{Sym}(\mathcal{U})^{\otimes 2}$, we define the solution set of the SDCP as:

$$\mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta) := \{\Sigma \in \mathrm{Sym}(\mathcal{U}) \ : \ 0 \preceq \Sigma \perp \alpha + \beta[\Sigma] \succeq_{\mathcal{U}} 0\}. \tag{49}$$

A priori, it is unclear whether the SDCP has zero, one, or many solutions. The following lemma shows that the SDCP always has one unique solution if β is symmetric and positive definite as an operator over $\mathrm{Sym}(\mathcal{U})$, i.e if:

$$\beta[\Sigma, \Sigma'] = \beta[\Sigma', \Sigma] \quad \forall\, \Sigma, \Sigma' \in \mathrm{Sym}(\mathcal{U}) \quad \text{and} \quad \beta[\Sigma, \Sigma] > 0 \quad \forall\, \Sigma \in \mathrm{Sym}(\mathcal{U}) \backslash \{0\}.$$

Lemma 1. If β is symmetric and positive definite over $\mathrm{Sym}(\mathcal{U})$, then the cardinality of the solution set satisfies $|\mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)| = 1$.

Proof. Consider the following quadratic program with a semidefinite constraint:

$$\min_{\Sigma \in \mathrm{Sym}(\mathcal{U})} \langle \alpha, \Sigma \rangle + \tfrac{1}{2}\beta[\Sigma, \Sigma] \quad \text{subject to} \quad \Sigma \succeq 0. \tag{50}$$

As $\beta \succ 0$, the objective is strictly convex so there is a unique minimizer $\Sigma^\star$. The KKT conditions for $\Sigma^\star$ are exactly $0 \preceq \Sigma^\star \perp \alpha + \beta[\Sigma^\star] \succeq 0$ and $\Sigma \in \mathrm{Sym}(\mathcal{U})$, so $\Sigma^\star \in \mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)$. Similarly if $\Sigma \in \mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)$ then Σ satisfies the KKT conditions for the strictly convex semidefinite quadratic program, so $\Sigma = \Sigma^\star$. □

Note that this lemma is simply the minor adaptation of a standard argument for linear complementarity problems. In the case where the lemma applies and the solution to the SDCP is unique, we will overload notation and use $\mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)$ to denote this unique solution.

Efficient computation Fortunately, solving $\mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)$ does not actually require materializing $\alpha \in \mathrm{Sym}(\mathbb{R}^d)$ and $\beta \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$ in full. Let $k := \dim \mathcal{U}$ denote the dimension of $\mathcal{U}$, which is typically $\ll d$, and let $U \in \mathbb{R}^{d \times k}$ denote a basis for $\mathcal{U}$. Then any $\Sigma \in \mathrm{Sym}(\mathcal{U})$ can be expressed as $\Sigma = UXU^\top$ for some $X \in \mathrm{Sym}(\mathbb{R}^k)$. The SDCP condition eq. (49) then reduces to a k-dimensional SDCP over $\mathbb{R}^k$:

$$0 \preceq X \perp \alpha_U + \beta_U[X] \succeq 0 \iff X \in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U, \beta_U), \tag{51}$$

where the matrix $\alpha_U \in \mathrm{Sym}(\mathbb{R}^k)$ is defined as:

$$\alpha_U := U^\top \alpha U \qquad \iff \qquad (\alpha_U)_{ij} = u_i^T \alpha u_j \tag{52}$$

and the tensor $\beta_U \in \mathrm{Sym}(\mathbb{R}^k)^{\otimes 2}$ is defined via its action as:

$$\beta_U[X] := U^\top \beta[UXU^\top]U \qquad \iff \qquad (\beta_U)_{ijpq} = \beta[u_i, u_j, u_p, u_q]. \tag{53}$$

Thus, to solve the original d-dimensional problem $\Sigma \in \mathrm{SDCP}_{\mathcal{U}}(\alpha, \beta)$, one can instead solve the k-dimensional problem $X \in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U, \beta_U)$, and then represent $\Sigma = UXU^\top$.

To solve $\mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U, \beta_U)$, we use a standard convex solver to solve the convex program eq. (50):

$$\min_{X \in \mathrm{Sym}(\mathbb{R}^k)} \langle \alpha_U, X \rangle + \tfrac{1}{2}\beta_U[X, X] \quad \text{subject to} \quad X \succeq 0. \tag{54}$$

A.1.5 On local time averaging

We intentionally do not specialize to a specific notion of "local time-average". The only properties of the local time-averaging operator $\mathbb{E}$ that we use are:

1. linearity, i.e. $\mathbb{E}[f + g] = \mathbb{E}[f] + \mathbb{E}[g]$ and $\mathbb{E}[cf] = c\,\mathbb{E}[f]$ for any constant c
2. the local time average of a constant c is itself: $\mathbb{E}[c] = c$
3. in the EOS regime when the sharpness fluctuates around $2/\eta$, the time-average is coarse enough to smooth out these fluctuations so that $S(\mathbb{E}[w_t]) = 2/\eta$

One reason why we do not further define the time-averaging operator is that even the appropriate timescale for the averaging operation (e.g. window size or kernel bandwidth) depends nontrivially on the local dynamics. Recall that in the relatively simple setting of one unstable eigenvalue that was analyzed in Damian et al. (2023), the EOS dynamics consists of consecutive cycles where the sharpness rises above, then falls below, the critical threshold $2/\eta$. In this setting, it is natural to choose an averaging timescale so as to average over a cycle. Yet, the analysis of Damian et al. (2023) shows that the length of the cycle depends on the initial position of the iterate along the top Hessian eigenvector at the instant where the sharpness crosses $2/\eta$ (the closer the iterate is to the directionwise optimum, the longer the cycle). Hence, even the choice of timescale is very nontrivial.

A.1.6 Smoothness of the central flows

Central flows are ordinary differential equations of the form: $\frac{dw(t)}{dt} = f(w)$, where f is not continuous everywhere. Thus, the ODE should be interpreted in the sense of Carathéodory: $w(t)$ is only differentiable at *almost* all t, and can have points of non-differentiability where the left and right derivatives differ. For example, at the instant when the gradient descent central flow first reaches EOS, the right and left derivatives of $w(t)$ will differ, as the left derivative is $-\eta \nabla L(w)$ while the right derivative is $-\eta \Pi^{\perp}_{\nabla S(w)} \nabla L(w)$. However, the central flow is *right*-differentiable for all t. Therefore, when we write $\frac{d}{dt}$ (e.g. in Proposition 1), it can either be interpreted as holding for almost all t, or it can be alternatively interpreted as a statement about the right derivative.

A.2 Gradient Descent

We now derive the central flow for gradient descent. In Section 3.2.1 we considered the special case where one eigenvalue is at the edge of stability, and is continuing to remain there. The complete central flow, derived here, applies in the more general setting where multiple eigenvalues are potentially at the edge of stability. It also allows eigenvalues to enter and leave the edge of stability when appropriate.

This section is structured as follows:

1. First, in Appendix A.2.1, we formulate the central flow as a *differential complementarity problem* (DCP): a dynamical system defined implicitly by combining differential equations with complementarity constraints.
2. Next, in Appendix A.2.2, we show that this DCP can be re-formulated into an ordinary differential equation with an explicit right-hand side that involves the solution to a semidefinite complementarity problem.
3. In Appendix A.2.3, we show that the central flow can be equivalently formulated as a *projected gradient flow* that projects the negative gradient onto the tangent cone of the stable region. Leveraging this projection interpretation, we prove that the central flow decreases the loss monotonically (Proposition 1), but at a slower rate than the unregularized gradient flow (Proposition 2).
4. Finally, in Appendix A.2.4, we describe how to discretize the central flow in practice.

A.2.1 The Differential Complementarity Problem (DCP) formulation

The central flow $w(t)$ will model the time-averaged trajectory of gradient descent $\{w_t\}$:

$$w(t) := \mathbb{E}[w_t]. \tag{55}$$

Let $\delta_t := w_t - w(t)$ denote the displacement between gradient descent and the central flow (i.e. "the oscillation") at step/time t. From the definition of $w(t)$ as the time-average, it follows that $\mathbb{E}[\delta_t] = 0$. Let $\Sigma(t) := \mathbb{E}[\delta_t \delta_t^\top]$ denote the covariance of these oscillations. That is, we are modeling the gradient descent trajectory as:

$$w_t = w(t) + \delta_t, \quad \text{where} \quad \mathbb{E}[\delta_t] = 0 \quad \text{and} \quad \mathbb{E}[\delta_t \delta_t^\top] = \Sigma(t). \tag{56}$$

Recall that gradient descent oscillates along the eigenvectors that are at the edge of stability. When the Hessian has multiple eigenvalues at $2/\eta$, the corresponding eigenvectors are not individually identifiable, since any linear combination of eigenvectors is also an eigenvector. Instead, what is identifiable is the corresponding eigenspace, i.e. the linear subspace comprising all such eigenvectors. We refer to this eigenspace as the *critical subspace*:

Definition 2 (Critical subspace for gradient descent)**.** Given weights $w \in \mathbb{R}^d$, the *critical subspace* $\mathcal{U}(w) \subseteq \mathbb{R}^d$ is defined as the Hessian's eigenspace corresponding to the eigenvalue $2/\eta$:

$$\mathcal{U}(w) := \ker \left[H(w) - \tfrac{2}{\eta} I \right] = \left\{ u \in \mathbb{R}^d : H(w)\, u = \tfrac{2}{\eta} u \right\}. \tag{57}$$

Thus, we assume that the oscillations $\{\delta_t\}$ are fully contained within the critical subspace:

$$\delta_t \in \mathcal{U}(w(t)) \iff \operatorname{span}[\Sigma(t)] \subseteq \mathcal{U}(w(t)). \tag{58}$$

We will now derive the central flow. By Taylor expansion of ∇L around $w(t)$ (see Appendix A.1.2, eq. (42)), the gradient at step t is:

$$\nabla L(w_t) \approx \nabla L(w(t)) + H(w(t))\delta_t + \tfrac{1}{2} \nabla H(w(t))^\top [\delta_t \delta_t^\top]. \tag{59}$$

Time-averaging both sides and using that $\mathbb{E}[\delta_t] = 0$ and $\mathbb{E}[\delta_t \delta_t^\top] = \Sigma(t)$ gives:

$$\mathbb{E}[\nabla L(w_t)] \approx \nabla L(w(t)) + \tfrac{1}{2} \nabla H(w(t))^\top [\Sigma(t)]. \tag{60}$$

Therefore, we make the ansatz that the central flow $w(t)$ takes the form:

$$\frac{dw}{dt} = -\eta\Big[\nabla L(w) + \tfrac{1}{2}\nabla H(w)^\top[\Sigma(t)]\Big], \tag{61}$$

for some unknown $\Sigma(t)$ which we will now determine.

To solve for $\Sigma(t)$, we impose three conditions for all times t:

1. **PSD:** As a covariance matrix, $\Sigma(t)$ is positive semidefinite (PSD): $\Sigma(t) \succeq 0$.
2. **Stability:** The sharpness remains bounded by $2/\eta$: $H(w(t)) \preceq (2/\eta)I$.
3. **Complementarity:** The oscillations are contained within the critical subspace: $\mathrm{span}[\Sigma(t)] \subseteq \mathcal{U}(w(t))$.

It will now be helpful to define the "residual" matrix $A(w)$ as:

$$A(w) := \tfrac{2}{\eta}I - H(w). \tag{62}$$

Notably, $\ker A(w)$ is precisely the critical subspace $\mathcal{U}(w)$, i.e. the eigenspace of $H(w)$ with eigenvalue $2/\eta$. With this notation, Conditions 1-3 can be expressed as:

$$\underbrace{\Sigma(t) \succeq 0}_{\text{PSD}}, \quad \underbrace{A(w(t)) \succeq 0}_{\text{stability}}, \quad \underbrace{\mathrm{span}[\Sigma(t)] \subseteq \ker A(w(t))}_{\text{complementarity}}. \tag{63}$$

As discussed in Appendix A.1.3, these three conditions are equivalent to the complementarity relation:

$$\Sigma(t) \succeq 0, \quad A(w(t)) \succeq 0, \quad \Sigma(t) \perp A(w(t)), \tag{64}$$

which we write more compactly as:

$$0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0. \tag{65}$$

We say $(w(t), \Sigma(t))$ follow the central flow if they satisfy eq. (61) along with this complementarity relation:[61]

Definition 3 (Gradient Descent Central Flow, DCP Formulation)**.** We say that $\{(w(t), \Sigma(t))\}_{t\geq 0}$ follow the gradient descent central flow if, for almost all t, they satisfy eq. (61) along with the conditions: $0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0$, where $A(w) := \frac{2}{\eta}I - H(w)$.

Definition 3 is an example of a *differential complementarity problem* (DCP) (Stewart, 2011), which are described in more detail in Appendix A.6. In a DCP, $\Sigma(t)$ is not defined explicitly, but is rather defined *implicitly* via a complementarity relation that the trajectory $(w(t), \Sigma(t))$ is required to satisfy.

A priori, it is not clear that a feasible $\Sigma(t)$ exists or is unique. To give an explicit expression for $\Sigma(t)$, and thereby turn Definition 3 into an ODE with an explicit right-hand side, we will next show that for almost all times t, $\Sigma(t)$ must be the unique solution to a certain semidefinite complementary problem.

A.2.2 The Ordinary Differential Equation (ODE) formulation

Before deriving the ODE formulation of the central flow, let us first explain why the DCP formulation, Definition 3, fails to immediately specify $\Sigma(t)$. At any instant t, there can be multiple Σ's which satisfy $0 \preceq \Sigma \perp A(w(t)) \succeq 0$; for example, the trivial choice $\Sigma = 0$ always works. Yet, most of these Σ's would cause the stability constraint $A(w(t+\epsilon)) \succeq 0$ to be violated if the dynamics eq. (61) are run for an infinitesimal amount of time ϵ. Intuitively, we need to meld together the static constraint $0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0$ with the dynamics eq. (61).

[61] We only require $w(t), \Sigma(t)$ to satisfy eq. (61) for "almost all t" as $w(t)$ may not be differentiable when an eigenvalue enters or leaves EOS. This is in line with the standard definition of a differential complementarity problem.

To do so, we appeal to Lemma 5 in Appendix A.6. This result essentially "differentiates" the complementarity relation $0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0$, to yield a new complementarity relation between $\Sigma(t)$ and the time derivative $\frac{d}{dt}A(w(t))$. In particular, Lemma 5 implies that under the flow defined by Definition 3, we must have:

$$0 \preceq \Sigma(t) \perp \tfrac{d}{dt}A(w(t)) \succeq_{\mathcal{U}(w)} 0. \tag{66}$$

We have thus turned a "position-level" constraint on the residual $A(w(t))$ into a "velocity-level" constraint on its time derivative $\frac{d}{dt}A(w(t))$. We now expand $\frac{d}{dt}A(w(t))$ to reveal its dependence on $\Sigma(t)$:

$$\begin{aligned}
\tfrac{d}{dt}A(w(t)) &= \nabla A(w)\left[\tfrac{dw}{dt}\right] && \text{(chain rule)} \\
&= -\nabla H(w)\left[\tfrac{dw}{dt}\right] && \text{(definition of } A\text{)} \\
&= -\nabla H(w)\Big[-\eta\Big[\nabla L(w) + \tfrac{1}{2}\nabla H(w)^\top[\Sigma(t)]\Big]\Big] && \text{(form of } \tfrac{dw}{dt}\text{)} \\
&= \underbrace{\eta\nabla H(w)[\nabla L(w)]}_{=:\alpha(w)} + \underbrace{\tfrac{1}{2}\eta\nabla H(w)\nabla H(w)^\top}_{=:\beta(w)}[\Sigma(t)] && \text{(linearity)}
\end{aligned}$$

This reveals that $\frac{d}{dt}A(w(t))$ is *affine* in $\Sigma(t)$. Namely, if we define the matrix $\alpha(w)$ and tensor $\beta(w)$ as:

$$\alpha(w) := \eta\nabla H(w)[\nabla L(w)] \in \mathrm{Sym}(\mathbb{R}^d), \quad \beta(w) := \tfrac{\eta}{2}\nabla H(w)\nabla H(w)^\top \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}, \tag{67}$$

then $\frac{d}{dt}A(w(t))$ is given by the affine expression:

$$\tfrac{d}{dt}A(w(t)) = \alpha(w) + \beta(w)[\Sigma(t)]. \tag{68}$$

Substituting this into eq. (66) implies that $\Sigma(t)$ must satisfy the complementarity relation:

$$0 \preceq \Sigma(t) \perp \alpha(w) + \beta(w)[\Sigma(t)] \succeq_{\mathcal{U}(w)} 0. \tag{69}$$

Since $\Sigma(t) \in \mathrm{Sym}(\mathcal{U}(w))$, this is precisely an SDCP (Appendix A.1.4) defined over the critical subspace $\mathcal{U}(w)$:

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w)}\left(\alpha(w), \beta(w)\right). \tag{70}$$

Thus, we are now ready to state the ODE formulation of the central flow.

Definition 4 (Gradient Descent Central Flow, ODE Formulation)**.** We say $\{w(t)\}_{t\geq 0}$ follows the gradient descent central flow if for almost all $t \geq 0$, $w(t)$ satisfies eq. (61) for some $\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w(t))}\left(\alpha(w(t)), \beta(w(t))\right)$.

This DCP-to-ODE conversion can be straightforwardly generalized to a broader class of DCPs, and this is done in Appendix A.6, Lemma 6. Our subsequent central flow derivations will directly invoke Lemma 6.

Existence and uniqueness of SDCP solution $\Sigma(t)$ Recall from Appendix A.1.4, Lemma 1 that the SDCP eq. (69) will have a unique solution $\Sigma(t)$ when $\beta(w)$ is symmetric and positive definite as a linear operator over $\mathrm{Sym}(\mathcal{U})$. Due to its outer product structure, $\beta(w)$ is always symmetric and positive *semi*-definite. If β is also full rank as an operator acting on $\mathrm{Sym}(\mathcal{U}(w))$, then it is positive *definite*, and $\Sigma(t)$ is unique. Empirically, in our experiments, we always do observe that this full-rank condition is satisfied (it is equivalent to full-rankness of $\beta_U(w)$ defined below in eq. (80)), and thus $\Sigma(t)$ is unique. Note that existence and uniqueness of $\Sigma(t)$ does not necessarily imply existence and uniqueness of the central flow ODE.

Existence and uniqueness of central flow Provided that $\beta(w)$ is full-rank as an operator on $\mathrm{Sym}(\mathcal{U}(w))$ for all w, prior results imply existence and uniqueness for the gradient descent central flow (see Appendix A.6).

One unstable eigenvalue As a sanity check, we now verify that when there is one eigenvalue at the edge of stability (i.e. when the critical subspace has dimension 1), Definition 4 recovers the central flow defined in Section 3.2.1.

In general, a subspace $\mathcal{U}$ of dimension 1 has the form $\mathcal{U} = \operatorname{span} u$ for some $u \in \mathbb{R}^d$, so $\operatorname{Sym}(\mathcal{U}) = \{\sigma^2\, uu^\top : \sigma^2 \in \mathbb{R}\}$, and $\operatorname{SDCP}_{\mathcal{U}}(\alpha, \beta)$ reduces to a 1-dimensional SDCP:

$$\operatorname{SDCP}_{\mathcal{U}}(\alpha, \beta) = \sigma^2 uu^\top, \quad \sigma^2 = \operatorname{SDCP}_{\mathbb{R}}(\alpha_u, \beta_u), \quad \underbrace{\alpha_u := u^\top \alpha\, u}_{\in \mathbb{R}}, \quad \beta_u := \underbrace{u^\top \beta[uu^\top] u}_{\in \mathbb{R}}.$$

Thus, σ^2 has the closed-form solution described in Remark 2:

$$\sigma^2 = \max\left(-\frac{\alpha_u}{\beta_u}\right).$$

Therefore, when there is one eigenvalue at the edge of stability, $\Sigma(t)$ from eq. (70) becomes:

$$\Sigma(t) = \sigma^2 uu^\top, \quad \sigma^2 = \max\left(-\frac{\alpha_u(w)}{\beta_u(w)}\right), \quad \alpha_u(w) = u^\top \alpha(w) u, \quad \beta_u(w) = u^\top \beta(w)[uu^\top] u,$$

where $u \in \mathbb{R}^d$ is the top eigenvector of $H(w)$ at w, and $\alpha(w), \beta(w)$ were defined in eq. (67). These simplify to:

$$\alpha_u(w) = \eta \nabla L(w)^\top \nabla S(w), \quad \beta_u(w) = \tfrac{\eta}{2} \|\nabla S(w)\|^2, \tag{71}$$

where we recall that $S(w)$ denotes the top eigenvalue of $H(w)$ at w. Therefore:

$$\sigma^2 = \max\left(\frac{-2\nabla L(w)^\top \nabla S(w)}{\|\nabla S(w)\|^2}, 0\right). \tag{72}$$

When $\langle -\nabla L(w), \nabla S(w) \rangle > 0$, i.e. when progressive sharpening holds, this recovers eq. (13). Else, $\sigma^2 = 0$.

Finally, since $\Sigma(t) = \sigma^2 uu^\top$, eq. (61) reduces to:

$$\frac{dw}{dt} = -\eta\big[\nabla L(w) + \tfrac{1}{2}\sigma^2 \nabla S(w)\big],$$

which recovers eq. (14).

Predicting time-averages The central flow can predict the time-average of various quantities, such as the loss or squared gradient norm, along the gradient descent trajectory. For any quantity $f(w)$, we write $\bar{f}(t)$ for the central flow's prediction for $\mathbb{E}[f(w_t)]$ at step t.

For example, the central flow's prediction $\bar{L}(t)$ for the time-averaged loss $\mathbb{E}[L(w_t)]$ at step t is given by:

$$\begin{aligned}
\mathbb{E}[L(w_t)] &= \mathbb{E}[L(w(t) + \delta_t)] \\
&\approx \mathbb{E}\left[L(w(t)) + \nabla L(w(t))^\top \delta_t + \tfrac{1}{2}\delta_t^\top H(w(t))\delta_t\right] && \text{(Taylor expansion)} \\
&= L(w(t)) + \tfrac{1}{2}\left\langle H(w(t)), \Sigma(t)\right\rangle && (\mathbb{E}[\delta_t] = 0) \\
&= L(w(t)) + \tfrac{1}{2}\operatorname{tr}\left[\tfrac{2}{\eta}\Sigma\right] && (H\Sigma = \tfrac{2}{\eta}\Sigma) \\
&= L(w(t)) + \tfrac{1}{\eta}\operatorname{tr}\Sigma(t) \\
&:= \bar{L}(t). && (73)
\end{aligned}$$

Similarly, the prediction for the time-averaged squared gradient norm $\mathbb{E}[\|\nabla L(w_t)\|^2]$ at step t is:

$$\begin{aligned}
\mathbb{E}[\|\nabla L(w_t)\|^2] &\approx \mathbb{E}\left[\|\nabla L(w(t)) + H(w(t))\delta_t\|^2\right] \\
&= \|\nabla L(w(t))\|^2 + \left\langle H^2(w(t)), \Sigma(t)\right\rangle \\
&= \|\nabla L(w(t))\|^2 + \tfrac{4}{\eta^2}\operatorname{tr}\Sigma(t). \\
&=: \overline{\|\nabla L(t)\|^2} && (74)
\end{aligned}$$

In general, for any function $f(w)$, the central flow predicts that the time-average of $f(w_t)$ at step t is:

$$\begin{aligned}\mathbb{E}[f(w_t)] &\approx \mathbb{E}\left[f(w(t)) + \nabla f(w(t))^\top \delta_t + \tfrac{1}{2}\delta_t^\top \nabla^2 f(w(t))\delta_t\right] \\ &= f(w(t)) + \tfrac{1}{2}\left\langle \nabla^2 f(w(t)), \Sigma(t)\right\rangle \\ &:= \bar{f}(t)\end{aligned} \tag{75}$$

(Note: our prediction eq. (74) for the squared gradient norm does not fit this template, as we choose to do a first-order of expansion of ∇L and then take the norm, rather than do a second-order expansion of $f(w) = \|\nabla L(w)\|^2$.)

The central flow can also predict the covariance of the oscillations. Let $\Sigma(t) = V(t)\,\Lambda(t)\,V(t)^\top$ be the (reduced) eigenvalue decomposition of the rank-k matrix $\Sigma(t)$, where $V(t) \in \mathbb{R}^{d\times k}$ and $\Lambda(t) \in \mathrm{diag}(\mathbb{R}^k)$. Define $x_t := V(t)^\top(w_t - w(t)) \in \mathbb{R}^k$ as the displacement of gradient descent from the central flow along these eigenvector directions. Then the central flow predicts that the covariance of these displacements is:

$$\mathbb{E}[x_t x_t^\top] = V(t)^\top\,\mathbb{E}[\delta_t\delta_t^\top]V(t) = V(t)^\top\Sigma(t)V(t) = \Lambda(t).$$

In particular, if we consider the i-th diagonal entry, the central flow predicts that the variance of oscillations along the i-th eigenvector of $\Sigma(t)$ should be equal to the i-th eigenvalue of $\Sigma(t)$:

$$\mathbb{E}\left[\Big(v_i(t)^\top(w_t - w(t))\Big)^2\right] = \lambda_i(t). \tag{76}$$

Basis-dependent version Naively computing the central flow's $\frac{dw}{dt}$ would be impractical, as storing $\Sigma(t)$ and $\alpha(w)$ would require $O(d^2)$ space, and storing $\beta(w)$ would require $O(d^4)$ space. Fortunately, because all necessary quantities are supported on the low-rank critical subspace, the central flow's $\frac{dw}{dt}$ can be computed efficiently using only $O(k^2d + k^4)$ space, where $k = \dim\mathcal{U}(w)$ is the dimension of the critical subspace, which is typically $\ll d$.

In particular, fix t, and let $U \in \mathbb{R}^{d\times k}$ be a basis for the critical subspace $\mathcal{U}(w(t))$. Then recall from Appendix A.1.4 that $\Sigma(t)$ can be represented as $\Sigma(t) = UXU^\top$ for some low-dimensional matrix $X \in \mathrm{Sym}(\mathbb{R}^k)$ that solves $X \in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U(w), \beta_U(w))$, where $\alpha_U \in \mathrm{Sym}(\mathbb{R}^k)$ and $\beta_U \in \mathrm{Sym}(\mathbb{R}^k)^{\otimes 2}$ were defined in eqs. (52) and (53).

Now we define $H_U(w) := U^\top H(w)U \in \mathrm{Sym}(\mathbb{R}^k)$ and its gradient $\nabla H_U(w) \in \mathrm{Sym}(\mathbb{R}^k) \otimes \mathbb{R}^d$:

$$H_U(w)_{ij} := u_i^\top H(w)u_j \quad \text{and} \quad \nabla H_U(w)_{ij} := \nabla_w[u_i^\top H(w)u_j]. \tag{77}$$

The tensor $\nabla H_U(w)$ only requires $O(k^2d)$ space to store, and can be computed in $O(k^2d)$ time by looping over all pairs (u_i, u_j) of columns of U and computing the third derivative $\nabla_w\left[u_i^\top H(w)u_j\right] \in \mathbb{R}^d$. Crucially, computing $\frac{dw}{dt}$ only requires access to the smaller $\nabla H_U(w)$ rather than the full $\nabla H(w)$. To see this, note that:

$$\nabla H_U(w)[v] = U^\top\nabla H(w)[v]U \quad \text{and} \quad \nabla H_U(w)^\top[X] = \nabla H(w)^\top[UXU^\top]. \tag{78}$$

Thus, the central flow eq. (61) takes the form:

$$\frac{dw}{dt} = -\eta\Big[\nabla L(w) + \tfrac{1}{2}\nabla H_U^\top(w)[X]\Big], \tag{79}$$

and $\alpha_U(w) \in \mathrm{Sym}(\mathbb{R}^k), \beta_U(w) \in \mathrm{Sym}(\mathbb{R}^k)^{\otimes 2}$ take the form:

$$\alpha_U(w) = \eta\nabla H_U(w)[\nabla L(w)], \quad \beta_U(w) = \tfrac{\eta}{2}\nabla H_U(w)\,\nabla H_U(w)^\top. \tag{80}$$

Thus, to compute $\frac{dw}{dt}$, we can compute $\nabla H_U(w)$, then use this to compute $\alpha_U(w)$ and $\beta_U(w)$ via eq. (80), then solve the k-dimensional problem $X \in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U(w), \beta_U(w))$, and then compute $\frac{dw}{dt}$ via eq. (79).

In practice, due to the non-smoothness of the central flow, we do not discretize the central flow by computing $\frac{dw}{dt}$ and taking an Euler step; instead, we directly discretize the DCP formulation, as described in Appendix A.6.1.

The time-averaged predictions can also be computed efficiently given a basis. If we pick U to be orthonormal ($U^\top U = I$), then the central flow's prediction eq. (73) for the time-averaged training loss at step t is:

$$\begin{aligned}\bar{L}(t) &:= L(w(t)) + \tfrac{1}{\eta}\operatorname{tr}\left[UXU^\top\right] \\ &= L(w(t)) + \tfrac{1}{\eta}\operatorname{tr}\left[XU^\top U\right] \\ &= L(w(t)) + \tfrac{1}{\eta}\operatorname{tr} X. \end{aligned} \tag{81}$$

Similarly, the prediction eq. (74) for the time-averaged squared gradient norm at step t is:

$$\begin{aligned}\overline{\|\nabla L(t)\|^2} &:= \|\nabla L(w(t))\|^2 + \tfrac{4}{\eta^2}\operatorname{tr}\left[UXU^\top\right]. \\ &= \|\nabla L(w(t))\|^2 + \tfrac{4}{\eta^2}\operatorname{tr} X. \end{aligned} \tag{82}$$

In general, for any function $f(w)$, the prediction eq. (75) can be computed as:

$$\bar{f}(t) = f(w(t)) + \tfrac{1}{2}\left\langle U^\top \nabla^2 f(w(t))U, X\right\rangle.$$

As for the oscillation covariance, we can evaluate both sides of eq. (76) without needing to materialize $\Sigma(t)$ in full. If $X = U_X\,\Lambda(t)\,U_X^\top$ denotes the eigenvalue decomposition of X, and if we define $V(t) = UU_X$, then $\Sigma(t) = V(t)\,\Lambda(t)\,V(t)^\top$ is the eigenvalue decomposition of $\Sigma(t)$. Note that U_X and X will depend on the basis U, while $V(t)$ and $\Lambda(t)$ are independent of U.

Smoothness of the central flow At a finite set of times, a new eigenvalue enters or leaves the edge of stability. We refer to these instants as *breakpoints*. In between the breakpoints, $\Sigma(t)$ is continuous and $w(t)$ is differentiable. Moreover, the SDCP is solved by the linear inverse $\Sigma = -U\beta_U^{-1}[\alpha_U]U^\top$ where α_U, β_U are defined in eq. (80) (see Remark 1). Further, $\frac{d}{dt}A(w(t))\,\big|_{\mathcal{U}(w)} = 0$, i.e. all Hessian eigenvalues that are at EOS remain fixed at $2/\eta$. However, at the breakpoints, $\Sigma(t)$ is discontinuous and $w(t)$ is not differentiable (although they are still right-continuous and right-differentiable, respectively).

A.2.3 The Projection Formulation

In this section we will show that the gradient descent central flow (Definition 3 and Definition 4) can be equivalently interpreted as projected gradient flow constrained to the *stable set* $\mathbb{S}_\eta$, i.e. the subset of weight space where gradient descent is locally stable:

$$\mathbb{S}_\eta := \{w \;:\; S(w) \le 2/\eta\}. \tag{83}$$

For general constrained optimization problems, a projected gradient flow projects the negative gradient onto the *tangent cone* of the constraint set before taking an infinitesimal step. The tangent cone consists of the set of allowable directions that would not cause any constraints to be violated.

In our case, the tangent cone $T_{\mathbb{S}_\eta(w)}$ of the stable set $\mathbb{S}_\eta$ at the point $w \in \mathbb{S}_\eta$ is the set of directions that, to first order, would not increase the sharpness if we moved in that direction from w. This tangent cone is given by:

$$T_{\mathbb{S}_\eta(w)} = \{z \in \mathbb{R}^d : \nabla H(w)[z] \preceq_{\mathcal{U}(w)} 0\}. \tag{84}$$

Note that this is a convex cone, since it is closed under linear combinations with non-negative weights.

We use $\operatorname{proj}_M(\cdot)$ to denote the usual Euclidean projection onto a set $M \subseteq \mathbb{R}^d$:

$$\operatorname{proj}_M(v) = \operatorname*{arg\,min}_{z \in M} \|v - z\|_2^2. \tag{85}$$

Projecting a vector onto the tangent cone of the stable set involves solving a certain SDCP:

Lemma 2. The projection of a vector $v \in \mathbb{R}^d$ onto the tangent cone of $\mathbb{S}_\eta$ at $w \in \mathbb{S}_\eta$ is given by:

$$\mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[v] = v - \tfrac{1}{2}\nabla H(w)^\top[\Sigma] \quad \text{where} \quad \Sigma \in \mathrm{SDCP}_{\mathcal{U}(w)}\left(-\nabla H(w)[v], \tfrac{1}{2}\nabla H(w)\nabla H(w)^\top\right), \tag{86}$$

where $\mathcal{U}(w) := \ker\left[H(w) - \frac{2}{\eta}I\right]$ is the critical subspace (Definition 2).

Proof. Recall that the tangent cone of $\mathbb{S}_\eta$ is the set: $\{z \in \mathbb{R}^d \ : \ \nabla H(w)[z] \preceq_{\mathcal{U}(w)} 0\}$. Therefore, the projection of v onto this set is given by:

$$\mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[v] = v + \delta^*, \tag{87}$$

where the perturbation δ^* is the optimal solution to the optimization problem:

$$\min_\delta \|\delta\|^2 \quad \text{such that} \quad \nabla H(w)[v+\delta] \preceq_{\mathcal{U}(w)} 0. \tag{88}$$

This is a quadratic program with a semidefinite constraint. Introducing a dual variable $\Sigma \in \mathrm{Sym}(\mathcal{U}(w))$, the KKT conditions for this optimization problem are:

$$\underbrace{\delta = -\tfrac{1}{2}\nabla H(w)^\top[\Sigma]}_{\text{stationarity}}, \quad \underbrace{\langle \Sigma, \nabla H(w)[v+\delta]\rangle = 0}_{\text{complementary slackness}}, \quad \underbrace{\nabla H(w)[v+\delta] \preceq_{\mathcal{U}(w)} 0}_{\text{primal feasibility}}, \quad \underbrace{\Sigma \succeq 0}_{\text{dual feasibility}} . \tag{89}$$

Substituting the first condition into the middle two yields the following three conditions:

$$0 \preceq \Sigma \perp -\nabla H(w)[v] + \tfrac{1}{2}\nabla H(w)\nabla H(w)^\top[\Sigma] \succeq_{\mathcal{U}(w)} 0. \tag{90}$$

We recognize these as precisely the characterization of an SDCP:

$$\Sigma \in \mathrm{SDCP}_{\mathcal{U}(w)}\left(-\nabla H(w)[v], \tfrac{1}{2}\nabla H(w)\nabla H(w)^\top\right). \tag{91}$$

Therefore, if Σ satisfies eq. (91), then $\delta = -\frac{1}{2}\nabla H(w)^\top[\Sigma]$ is an optimal solution to the optimization problem eq. (88), and $v + \delta$ is the desired projection. □

We are now ready to state the projection formulation of the gradient descent central flow:

Definition 5 (Gradient Descent Central Flow, Projection Formulation)**.** We say that $\{w(t)\}_{t\geq 0}$ follows the gradient descent central flow if for almost all t,

$$\frac{dw}{dt} = \mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[-\eta\nabla L(w)] \quad \text{where} \quad \mathbb{S}_\eta := \{w \ : \ S(w) \leq 2/\eta\}. \tag{92}$$

The equivalence between the projection formulation (Definition 5) and the ODE formulation (Definition 4) follows from applying Lemma 2 to the vector $v = -\eta\nabla L(w)$ and noting $\mathrm{SDCP}(\alpha, \beta) = \mathrm{SDCP}(c\alpha, c\beta)$ for $c > 0$.

Understanding the projection formulation We now give intuition for the projection formulation:

- When $S(w) < 2/\eta$, w is in the interior of $\mathbb{S}_\eta$ so $\mathcal{U}(w) = \emptyset$, the tangent cone is the entire space, and the projection is the identity map. Therefore eq. (92) reduces to gradient flow.
- When there is a single eigenvalue at $2/\eta$, w is on the boundary of $\mathbb{S}_\eta$ and the tangent cone is given by the halfspace: $T_{\mathbb{S}_\eta}(w) = \{v : \langle \nabla S(w), v\rangle \leq 0\}$. If the negative gradient lies outside this halfspace (i.e. if gradient flow threatens to increase the sharpness above $2/\eta$), then the projection onto the halfspace is given by the projection onto the hyperplane: $-\eta\Pi^\perp_{\nabla S(w)}\nabla L(w)$ (see Figure 24). But, if the negative gradient already lies in the halfspace, the projection is the identity map, so the central flow follows gradient flow and leaves EOS.

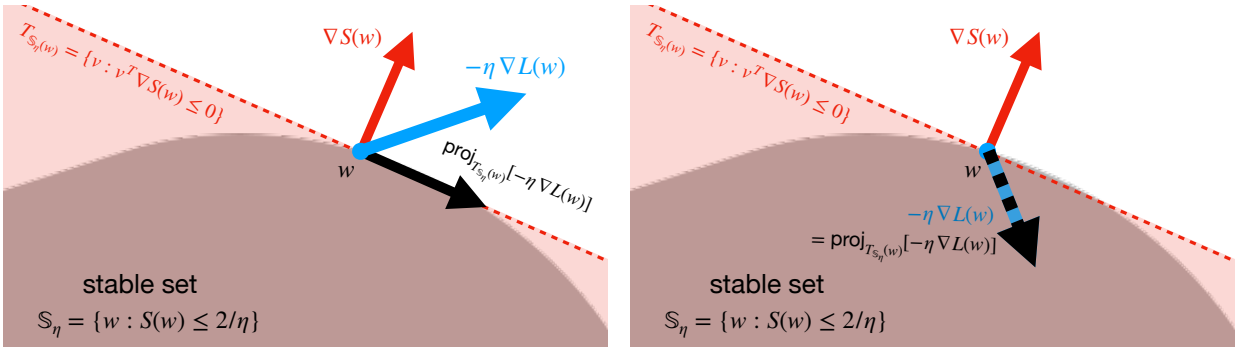


Figure 24: This cartoon illustrates projecting onto the tangent cone of the stable set $T_{\mathbb{S}_\eta(w)}$ in the case where one eigenvalue at the edge of stability. The iterate w is on the border of the stable set (grey blob). The tangent cone is the half-space $\{v : v^\top \nabla S(w) \le 0\}$ (shaded red). **Left**: on the one hand, if the negative gradient (blue arrow) points *out of* the stable set, then the projection (black arrow) removes the component aligned with $S(w)$ (red arrow). **Right**: on the other hand, if the negative gradient (blue arrow) already points *into* the stable set, then the projection (black arrow) does nothing.

- In general, computing the projection onto $T_{\mathbb{S}_\eta(w)}$ requires solving a semidefinite quadratic program for which Σ is the Lagrangian dual variable. The KKT conditions of this quadratic program are equivalent (up to a constant) to the SDCP that defines Σ above.

Properties of projection This projection formulation is helpful because Euclidean projection onto a convex cone shares some useful properties with Euclidean projection onto a linear subspace. We will use these properties below to reason about the rate of loss decrease.

First, projection onto a convex cone C is positive homogeneous: for any scalar $c > 0$ we have:

$$\text{proj}_C[cv] = c\,\text{proj}_C[v]. \tag{93}$$

In our case, this can also be seen directly by combining the characterization of the projection in Lemma 2, with the identity $X \in \text{SDCP}(\alpha, \beta) \iff cX \in \text{SDCP}(c\alpha, \beta)$.

Second, by the Moreau decomposition (Moreau, 1962), any vector v can be orthogonally decomposed into the projection onto a convex cone C and the projection onto its dual cone C^*:

$$v = \text{proj}_C[v] + \text{proj}_{C^*}[v] \quad \text{where} \quad \langle \text{proj}_C[v], \text{proj}_{C^*}[v] \rangle = 0. \tag{94}$$

In particular, this implies that:

$$\langle \text{proj}_C[v], v - \text{proj}_C[v] \rangle = 0. \tag{95}$$

and:

$$\langle v, \text{proj}_C[v] \rangle = \|\text{proj}_C[v]\|^2. \tag{96}$$

In our case, C is the tangent cone to the stable set at w, its dual cone C^* is the so-called *normal cone* to the stable set at w: $\{\nabla H(w)^\top[\Sigma] : \Sigma \in \text{Sym}(\mathcal{U}(w)), \Sigma \succeq 0\}$, and eq. (95) can be proved by rearranging the complementarity relation in eq. (90).

Rate of loss decrease We now use the projection formulation (Definition 5) to reason about the rate of loss decrease under the central flow. We first show the following helper lemma:

Lemma 3. Under the gradient descent central flow (Definition 5), for almost all t we have

$$\frac{dL(w)}{dt} = -\eta \left\| \text{proj}_{T_{\mathbb{S}_\eta}(w)}[-\nabla L(w)] \right\|^2. \tag{97}$$

Proof. By the chain rule, we have

$$\begin{aligned}\frac{dL(w)}{dt} &= \left\langle \nabla L(w), \frac{dw}{dt} \right\rangle \\ &= \left\langle \nabla L(w), \mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[-\eta \nabla L(w)] \right\rangle \\ &= -\eta \left\langle -\nabla L(w), \mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[-\nabla L(w)] \right\rangle \\ &= -\eta \left\| \mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[-\nabla L(w)] \right\|^2,\end{aligned}$$

where the first line is the chain rule, the second line is the projection formulation of the central flow, the third line is due to positive homogeneity of the projection operation eq. (93), and the last line is due to the orthogonality of the projection eq. (96). □

A simple corollary is that the training loss monotonically decreases under the gradient descent central flow:

Proposition 1 (Restated)**.** Under the GD central flow (Definition 5), for almost all t, the loss curve $L(w(t))$ is monotonically decreasing:

$$\frac{dL(w(t))}{dt} \leq 0. \tag{98}$$

Proof. The claim follows by combining Lemma 3 with the fact that a norm is always non-negative. □

Another simple corollary is that the central flow decreases the loss at a less steep rate than would gradient flow. In other words, the oscillations induce a slowdown in the rate of loss decrease:

Proposition 2 (Restated)**.** Under the GD central flow (Definition 5), for almost all t, the slope of the loss curve is less steep than that of gradient flow:

$$-\eta \|\nabla L(w(t))\|^2 \leq \frac{dL(w(t))}{dt}. \tag{99}$$

Proof. Projecting a vector v onto a convex cone C always makes the norm smaller:

$$\begin{aligned}\|v\|^2 &= \|\mathrm{proj}_C[v] + [v - \mathrm{proj}_C[v]]\|^2 \\ &= \|\mathrm{proj}_C[v]\|^2 + \|v - \mathrm{proj}_C[v]\|^2 \\ &\geq \|\mathrm{proj}_C[v]\|^2,\end{aligned}$$

where the second line is due to the orthogonality of $\mathrm{proj}_C[v]$ and $v - \mathrm{proj}_C[v]$ i.e. eq. (95), and the third line is due to the non-negativity of a square.

Thus, recalling Lemma 3, we have:

$$\begin{aligned}\frac{dL(w)}{dt} &= -\eta \left\| \mathrm{proj}_{T_{\mathbb{S}_\eta}(w)}[-\nabla L(w)] \right\|^2 \\ &\geq -\eta \| - \nabla L(w)\|^2 \\ &= -\eta \|\nabla L(w)\|^2.\end{aligned}$$

□

Why not start with the projection formulation? Since the projection formulation of the central flow is arguably the simplest one, one might ask why we first went through the DCP/ODE formulations before arriving at the projection formulation. After all, since the sharpness equilibrates at $2/\eta$ at EOS, one might think that a projected gradient flow constrained to the set $\{w : S(w) \leq 2/\eta\}$ is already a natural approximation. The trouble with this thinking is that there are actually an infinite number of flows which keep the sharpness locked at $2/\eta$, moving within the tangent cone of the stable set. Among these, the significance of the central flow (Definition 5) is that it follows the particular vector within the tangent cone that is *closest* (in Euclidean distance) to the negative gradient; that is, it makes the *smallest perturbation* to the negative gradient that will force it inside the tangent cone. A priori, there is no reason why this should be the case, and thus jumping straight to the projection formulation would be arbitrary. In fact, we will see in our analyses of Scalar RMSProp and RMSProp that the central flows *do not* pick the closest tangent vector in Euclidean distance and their central flows cannot be interpreted as a projected gradient flow.

A.2.4 Discretizing the gradient descent central flow

Discretizing the central flow is nontrivial, because the flow is nonsmooth at points where the dimension of the critical subspace (i.e. the number of unstable eigenvalues) undergoes a change. To discretize the flow, we directly discretize the DCP formulation (Definition 3) rather than going through the ODE formulation (Definition 4). We describe our general procedure for discretizing DCPs in Appendix A.6.1. Let us now describe how this general procedure specializes to the gradient descent case.

We use $w^{(t)}$, $\Sigma^{(t)}$ to denote our estimate for the central flow's $w(t)$, $\Sigma(t)$. Let $\epsilon > 0$ be the discretization step size, e.g. $\epsilon = 0.25$. For some tolerance $\tau > 0$, e.g. $\tau = \frac{0.05}{\eta}$, we will regard Hessian eigenvalues greater than $\frac{2}{\eta} - \tau$ as those which might become unstable in the next discretization time step.

At each discretization step, we first compute all Hessian eigenvalues that are greater than $\frac{2}{\eta} - \tau$, as well as the corresponding eigenvectors. Let k be the number of such eigenvalues, let $D \in \mathbb{R}^{k \times k}$ be a diagonal matrix containing such eigenvalues on the diagonal, and let $U \in \mathbb{R}^{d \times k}$ be the corresponding orthonormal eigenvectors. Then, we compute the tensor ∇H_U as in eq. (77), though note that U now refers to a basis of eigenvectors whose eigenvalues are *almost* 2 rather than exactly equal to 2. Then, we compute α_U and β_U as in eq. (80). Then, we solve the following k-dimensional SDCP:

$$X^{(t)} = \text{SDCP}_{\mathbb{R}^k}(\tfrac{2}{\eta} I - D + \epsilon\, \alpha_U, \epsilon\, \beta_U), \tag{100}$$

so that $\Sigma^{(t)} = U X^{(t)} U^\top$. Then, we update the weights via:

$$w^{(t+\epsilon)} = w^{(t)} - \epsilon\, \eta \Big[\nabla L(w^{(t)}) + \tfrac{1}{2} \nabla H_U^\top [X^{(t)}] \Big]. \tag{101}$$

To predict the time-average of the train loss, the squared gradient, and the covariance of the oscillations, we use eq. (81), eq. (82), and eq. (76), respectively.

A.3 Scalar RMSProp

We model the Scalar RMSProp iterates $\{w_t\}$ as oscillating around a central flow $w(t)$ with mean zero and covariance $\Sigma(t)$. That is, if $\delta_t := w_t - w(t)$ denotes the displacement ("the oscillation"), then $\mathbb{E}[\delta_t] = 0$ and $\mathbb{E}[\delta_t \delta_t^\top] = \Sigma(t)$. Let $\nu(t) := \mathbb{E}[v_t]$ model the time-averaged ν_t, and we will frequently neglect the distinction between the two, implicitly assuming that ν_t concentrates tightly around $\nu(t)$.

A similar argument as for gradient descent implies that the time-averaged gradient is approximately:

$$\mathbb{E}[\nabla L(w_t)] \approx \nabla L(w(t)) + \tfrac{1}{2} \nabla H(w(t))^\top [\Sigma(t)]. \tag{102}$$

Meanwhile, we approximate the time-average of the squared gradient norm as:[62]

$$\begin{aligned}\mathbb{E}[\|\nabla L(w_t)\|^2] &\approx \mathbb{E}[\|\nabla L(w(t)) + H(w(t))\,\delta_t\|^2] \\ &= \|\nabla L(w(t))\|^2 + \mathbb{E}[\|H(w(t))\,\delta_t\|^2]\end{aligned} \tag{103}$$

where the first line is a first-order Taylor expansion of ∇L around $w(t)$ and the second line is because $\mathbb{E}[\delta_t] = 0$.

We define the critical subspace $\mathcal{U}(w, \nu)$ as the eigenspace of the effective Hessian $\frac{\eta}{\sqrt{\nu}}H(w)$ that corresponds to the eigenvalue 2:

$$\mathcal{U}(w, \nu) := \ker\Big[\tfrac{\eta}{\sqrt{\nu}}H(w) - 2I\Big]. \tag{104}$$

We model Scalar RMSProp as oscillating within the critical subspace, i.e. we assume that $\delta_t \in \mathcal{U}(w(t), \nu(t))$.

This implies that $H(w(t))\,\delta_t = \frac{2\sqrt{\nu(t)}}{\eta}\delta_t$, which lets us simplify eq. (103) as:

$$\begin{aligned}\mathbb{E}[\|\nabla L(w_t)\|^2] &\approx \|\nabla L(w(t))\|^2 + \mathbb{E}[\|H(w(t))\,\delta_t\|^2] \\ &= \|\nabla L(w(t))\|^2 + \tfrac{4\nu(t)}{\eta^2}\,\mathbb{E}[\|\delta_t\|^2] \\ &= \|\nabla L(w(t))\|^2 + \tfrac{4\nu(t)}{\eta^2}\,\operatorname{tr}\Sigma(t),\end{aligned} \tag{105}$$

where the final line is because $\mathbb{E}[\|\delta_t\|^2] = \mathbb{E}[\operatorname{tr}\big(\delta_t\delta_t^\top\big)] = \operatorname{tr}\Sigma(t)$.

Based on the time averages eq. (102) and eq. (105), we make the ansatz that the joint dynamics of (w_t, ν_t) can be modeled by a central flow $(w(t), \nu(t))$ of the form:

$$\begin{aligned}\frac{dw}{dt} &= -\frac{\eta}{\sqrt{\nu}}\Big[\nabla L(w) + \tfrac{1}{2}\nabla H(w)^\top[\Sigma(t)]\Big] \\ \frac{d\nu}{dt} &= \tfrac{1-\beta_2}{\beta_2}\Big[\|\nabla L(w)\|^2 + \tfrac{4\nu}{\eta^2}\operatorname{tr}(\Sigma(t)) - \nu\Big].\end{aligned} \tag{106}$$

As with gradient descent, to determine $\Sigma(t)$ we will impose three conditions for all times t:

1. **PSD:** As a covariance matrix, $\Sigma(t)$ is positive semidefinite (PSD), i.e. $\Sigma(t) \succeq 0$.
2. **Stability:** The effective sharpness remains bounded by 2, i.e. $\frac{\eta}{\sqrt{\nu}}H(w) \preceq 2I$.
3. **Complementarity:** The oscillations are contained within the critical subspace, i.e $\operatorname{span}\Sigma(t) \subseteq \mathcal{U}(w(t), \nu(t))$.

To write these concisely, it will be convenient to define the matrix-valued "residual" function $A(w, \nu)$ by:

$$A(w, \nu) := \tfrac{2\sqrt{\nu}}{\eta}I - H(w), \tag{107}$$

so that stability is equivalent to $A(w, \nu) \succeq 0$ and the critical subspace is precisely $\mathcal{U}(w, \nu) = \ker A(w, \nu)$. With this notation, we can concisely express the above three conditions as a semidefinite complementarity relation:

$$0 \preceq \Sigma(t) \perp A(w(t), \nu(t)) \succeq 0.$$

We say that $(w(t), \Sigma(t))$ follow the Scalar RMSProp central flow if they follow eq. (106) along with this semidefinite complementarity relation:

Definition 6 (Scalar RMSProp Central Flow, DCP Formulation)**.** We say that $\{(w(t), \nu(t), \Sigma(t))\}_{t\geq 0}$ satisfy the Scalar RMSProp central flow if they satisfy eq. (106) and $0 \preceq \Sigma(t) \perp A(w(t), \nu(t)) \succeq 0$ for almost all t.

[62] We note here that this is not a "faithful" second-order Taylor expansion of the gradient $\nabla L(w_t)$ around $w(t)$. We are implicitly assuming that $\|\mathbb{E}[\nabla L(w_t)]\|^2 \approx \|\nabla L(\mathbb{E}[w_t])\|^2$ which neglects the term $2\nabla^3 L(w(t))[\nabla L(w(t)), \Sigma(t)]$. This omission simplifies the expressions and our numerical experiments still successfully predict $\Sigma(t)$ across a variety of datasets and architectures which justifies this omission.

Since this DCP can be expressed in the general form described in Appendix A.6, we can use Lemma 6 to convert it into an equivalent ODE. In particular, if $\widetilde{w} := [w, \nu]^\top$ denotes the augmented state, we can write eq. (106) as:

$$\frac{d\widetilde{w}}{dt} = f(\widetilde{w}) + B(\widetilde{w})[\Sigma] \quad \text{where} \quad f(\widetilde{w}) := \begin{bmatrix} -\frac{\eta}{\sqrt{\nu}}\nabla L(w) \\ \frac{1-\beta_2}{\beta_2}[\|\nabla L(w)\|^2 - \nu] \end{bmatrix} \quad \text{and} \quad B(\widetilde{w})[\Sigma] := \begin{bmatrix} -\frac{\eta}{2\sqrt{\nu}}\nabla H(w)^\top[\Sigma] \\ \frac{1-\beta_2}{\beta_2} \cdot \frac{4\nu}{\eta^2} \operatorname{tr}\Sigma \end{bmatrix},$$

and we can write the complementarity relation as $0 \preceq \Sigma(t) \perp A(\widetilde{w}(t)) \succeq 0$.

Therefore, Lemma 6 implies that

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w,\nu)}\left(\alpha(w,\nu), \beta(w,\nu)\right),$$

where the matrix $\alpha(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)$ and tensor $\beta(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$ are defined by:

$$\begin{aligned}
\alpha(w,\nu) &:= \nabla A(\widetilde{w})[f(\widetilde{w})] \\
&= \begin{bmatrix} -\nabla H(w) & \frac{1}{\eta\sqrt{\nu}} I \end{bmatrix} \begin{bmatrix} -\frac{\eta}{\sqrt{\nu}}\nabla L(w) \\ \frac{1-\beta_2}{\beta_2}\|\nabla L(w)\|^2 - \nu \end{bmatrix} \\
&= \frac{\eta}{\sqrt{\nu}}\nabla H(w)[\nabla L(w)] + \frac{1}{\eta\sqrt{\nu}}\frac{1-\beta_2}{\beta_2}\Big[\|\nabla L(w)\|^2 - \nu\Big] I
\end{aligned}$$

and

$$\begin{aligned}
\beta(w,\nu)[\Sigma] &:= \nabla A(\widetilde{w}) B(\widetilde{w})[\Sigma] \\
&= \begin{bmatrix} -\nabla H(w) & \frac{1}{\eta\sqrt{\nu}} I \end{bmatrix} \begin{bmatrix} -\frac{\eta}{2\sqrt{\nu}}\nabla H(w)^\top[\Sigma] \\ \frac{1-\beta_2}{\beta_2} \cdot \frac{4\nu}{\eta^2} \operatorname{tr}\Sigma \end{bmatrix} \\
&= \frac{\eta}{2\sqrt{\nu}}\nabla H(w)\nabla H(w)^\top[\Sigma] + \frac{4\sqrt{\nu}}{\eta^3} \cdot \frac{1-\beta_2}{\beta_2} \operatorname{tr}[\Sigma] I.
\end{aligned}$$

Note that $\beta(\widetilde{w})$ is indeed symmetric PSD, as for gradient descent:

$$\begin{aligned}
\beta(\tilde{w})[\Sigma, \Sigma'] &= \frac{\eta}{2\sqrt{\nu}}\left\langle \nabla H(w)^\top[\Sigma], \nabla H(w)^\top[\Sigma'] \right\rangle + \frac{4\sqrt{\nu}}{\eta^3} \cdot \frac{1-\beta_2}{\beta_2} \operatorname{tr}[\Sigma] \operatorname{tr}\big[\Sigma'\big] \\
\implies \beta(\tilde{w})[\Sigma, \Sigma] &= \frac{\eta}{2\sqrt{\nu}}\Big\|\nabla H(w)^\top[\Sigma]\Big\|^2 + \frac{4\sqrt{\nu}}{\eta^3} \cdot \frac{1-\beta_2}{\beta_2} \operatorname{tr}[\Sigma]^2 \geq 0.
\end{aligned}$$

This gives us the ODE formulation of the Scalar RMSProp central flow:

Definition 7 (Scalar RMSProp Central Flow, ODE Formulation)**.** We say that $\{w(t), \nu(t)\}_{t\geq 0}$ follow the Scalar RMSProp central flow if for almost all t, they satisfy eq. (106) for some

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w(t),\nu(t))}\left(\alpha(w(t),\nu(t)), \beta(w(t),\nu(t))\right).$$

One unstable eigenvalue As a sanity check, we now verify that this formulation recovers eq. (27) when there is one eigenvalue at the edge of stability. Just as with gradient descent (see above), in this setting $\Sigma(t)$ reduces to:

$$\Sigma(t) = \sigma^2 uu^\top, \quad \sigma^2 = \max\left(-\frac{\alpha_u(w,\nu)}{\beta_u(w,\nu)}\right), \quad \alpha_u(w,\nu) = u^\top\alpha(w,\nu)u, \quad \beta_u(w,\nu) = u^\top\beta(w,\nu)[uu^\top]u,$$

where $u \in \mathbb{R}^d$ is the unit-normalized top eigenvector of $H(w)$ at w. Simplifying, we have:

$$\begin{aligned}
\alpha_u(w,\nu) &= \frac{\eta}{\sqrt{\nu}}\left\langle \nabla L(w), \nabla S(w) \right\rangle + \frac{1}{\eta\sqrt{\nu}} \cdot \frac{1-\beta_2}{\beta_2}\Big[\|\nabla L(w)\|^2 - \nu\Big] \\
\beta_u(w,\nu) &= \frac{\eta}{2\sqrt{\nu}}\|\nabla S(w)\|^2 + \frac{4\sqrt{\nu}}{\eta^3} \cdot \frac{1-\beta_2}{\beta_2}.
\end{aligned}$$

Then, when $\alpha_u(w, \nu)$ is negative, we have:

$$\begin{aligned}
\sigma^2 &= -\frac{\alpha_u(w,\nu)}{\beta_u(w,\nu)} \\
&= -\frac{\frac{\eta}{\sqrt{\nu}}\left\langle \nabla L(w), \nabla S(w)\right\rangle + \frac{1}{\eta\sqrt{\nu}}\cdot\frac{1-\beta_2}{\beta_2}\left[\|\nabla L(w)\|^2 - \nu\right]}{\frac{\eta}{2\sqrt{\nu}}\|\nabla S(w)\|^2 + \frac{4\sqrt{\nu}}{\eta^3}\cdot\frac{1-\beta_2}{\beta_2}} && \text{(definition of } \alpha_u, \beta_u\text{)} \\
&= \frac{\beta_2\left\langle -\nabla L(w), \nabla S(w)\right\rangle + \frac{1}{\eta^2}\cdot(1-\beta_2)\left[\nu - \|\nabla L(w)\|^2\right]}{\beta_2\cdot\frac{1}{2}\|\nabla S(w)\|^2 + (1-\beta_2)\frac{4\nu}{\eta^4}} && \text{(simplify)} \\
&= \frac{\beta_2\left\langle -\nabla L(w), \nabla S(w)\right\rangle + (1-\beta_2)\left[\frac{S(w)^2}{4} - \frac{\|\nabla L(w)\|^2}{\eta^2}\right]}{\beta_2\cdot\frac{1}{2}\|\nabla S(w)\|^2 + (1-\beta_2)S(w)^2/\eta^2}. && (\nu = \tfrac{\eta^2 S(w)^2}{4} \text{ at EOS})
\end{aligned}$$

which indeed recovers eq. (27). On the other hand, when $\alpha_u(w, \nu)$ is positive, $\sigma^2 = 0$ and the central flow reduces to the stable flow eq. (25).

Normalized gradient descent As $\beta_2 \to 0$, Scalar RMSProp becomes normalized gradient descent, and the formula eq. (27) for $\sigma^2(w; \eta, \beta_2)$ reduces to:

$$\sigma^2(w;\eta) = \frac{\eta^2}{4} - \frac{\|\nabla L(w)\|^2}{S(w)^2}.$$

In practice, we observe that the first term generally dominates the second term. Thus, for NGD with one unstable eigenvalue, we have approximately:

$$\sigma^2(w;\eta) \approx \frac{\eta^2}{4}. \tag{108}$$

This clearly illustrates how σ^2 grows monotonically with η.

Does the loss decrease? Whereas the gradient descent central flow decreases the loss monotonically (Proposition 1), this is not true for the Scalar RMSProp central flow. Indeed, we have seen that with one unstable eigenvalue, the Scalar RMSProp central flow takes the form:

$$\frac{dw}{dt} = -\frac{2}{S(w)}\left[\nabla L(w) + \frac{1}{2}\sigma^2(w;\eta,\beta_2)\nabla S(w)\right].$$

Thus, by the chain rule, the rate of change in the loss is given by:

$$\begin{aligned}
\frac{dL}{dt} &= \left\langle \nabla L(w), \frac{dw}{dt}\right\rangle \\
&= -\frac{2}{S(w)}\left[\|\nabla L(w)\|^2 + \frac{1}{2}\sigma^2(w;\eta,\beta_2)\left\langle \nabla L(w), \nabla S(w)\right\rangle\right].
\end{aligned}$$

If progressive sharpening holds, i.e. if $\langle \nabla L(w), \nabla S(w)\rangle < 0$, then the second term is acting to *increase* L, and for sufficiently large values of σ^2, this increase will outweigh the decrease from the first term, causing L to go up. Indeed, if progressive sharpening holds, then $\frac{dL}{dt} > 0$ so long as:

$$\sigma^2(w;\eta,\beta_2) > \frac{2\|\nabla L(w)\|^2}{-\left\langle \nabla L(w), \nabla S(w)\right\rangle}.$$

This can indeed occur. For instance, if we consider the case of normalized gradient descent ($\beta_2 \to 0$) and if we make the approximation described in eq. (108), then $\frac{dL}{dt} > 0$ so long as the learning rate η satisfies:

$$\eta > \sqrt{\frac{8\|\nabla L(w)\|^2}{-\left\langle \nabla L(w), \nabla S(w)\right\rangle}}. \tag{109}$$

Thus, for sufficiently large learning rates η, the train loss will go up rather than down under the central flow.

The effect of the hyperparameters η, β_2

Predicting time-averages As with gradient descent, the Scalar RMSProp central flow can predict the time-average of various quantities, such as the loss or squared gradient norm, along the Scalar RMSProp trajectory. Recall that for a function $f(w)$, we use $\bar{f}(t)$ to denote the central flow's prediction for the time-average $\mathbb{E}[f(w_t)]$ at step t.

The prediction $\bar{L}(t)$ for the time-averaged loss at step t is:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \tfrac{\sqrt{\nu}(t)}{\eta} \operatorname{tr} \Sigma(t) =: \bar{L}(t). \tag{110}$$

The prediction for the time-averaged squared gradient norm at step t is:

$$\mathbb{E}[\|\nabla L(w_t)\|^2] \approx \|\nabla L(w(t))\|^2 + \tfrac{4\nu(t)}{\eta^2} \operatorname{tr} \Sigma(t) =: \overline{\|\nabla L(t)\|^2}. \tag{111}$$

These can be derived along similar lines as eqs. (73) and (74) for gradient descent, but using the Scalar RMSProp condition $H(w)\,\Sigma(t) = \frac{2\sqrt{\nu}(t)}{\eta}\Sigma(t)$.

As for predicting the covariance of the oscillations, let $\Sigma(t) = V(t)\Lambda(t)V(t)^\top$ be the (reduced) eigenvalue decomposition of the rank-k matrix $\Sigma(t)$, where $V(t) \in \mathbb{R}^{d\times k}$ and $\Lambda(t) \in \operatorname{diag}(\mathbb{R}^k)$. Then the central flow predicts that the variance of oscillations along the i-th eigenvector of $\Sigma(t)$ should be equal to the i-th eigenvalue of $\Sigma(t)$:

$$\mathbb{E}\left[\left(v_i(t)^\top (w_t - w(t))\right)^2\right] = \lambda_i(t). \tag{112}$$

Practical implementation When implementing the Scalar RMSProp central flow, we treat Scalar RMSProp as an instance of a more general class of adaptive preconditioned methods that is described in Appendix A.5. Please refer to that section for details on how we discretize the central flow in practice.

A.4 RMSProp

We will begin by describing the stability condition for preconditioned gradient descent on a quadratic. Consider optimizing the quadratic $L(w) = \frac{1}{2}w^\top H w$ using preconditioned gradient descent with preconditioner $P \succ 0$:

$$w \leftarrow w - P^{-1}\nabla L(w) = w - P^{-1}Hw = (I - P^{-1}H)w. \tag{113}$$

The matrix $P^{-1}H$ is non-symmetric, but is similar to the symmetric matrix $P^{-1/2}HP^{-1/2}$, and thus has the same eigenvalues, which are necessarily real. If all eigenvalues of $P^{-1}H$ are contained in $(0, 2)$ then all eigenvalues of $(I - P^{-1}H)$ are contained in $(-1, 1)$, and hence $w \to 0$ exponentially fast. Otherwise, the dynamics will diverge along the *right eigenvectors* of $P^{-1}H$ with eigenvalues outside this range.[63]

We now derive the RMSProp central flow. We model the RMSProp iterates $\{w_t\}$ as oscillating around a central flow $w(t)$ with mean zero and covariance $\Sigma(t)$. That is, if $\delta_t := w_t - w(t)$ denotes the displacement between RMSProp and the central flow ("the oscillation"), then $\mathbb{E}[\delta_t] = 0$ and $\mathbb{E}[\delta_t\delta_t^\top] = \Sigma(t)$. Let $\nu(t) := \mathbb{E}[v_t]$ model the time-averaged ν_t, and we will frequently neglect the distinction between the two, implicitly assuming that ν_t concentrates tightly around $\nu(t)$.

A similar argument as for gradient descent implies that the time-averaged gradient is approximately:

$$\mathbb{E}[\nabla L(w_t)] \approx \nabla L(w(t)) + \tfrac{1}{2}\nabla H(w(t))^\top[\Sigma(t)]. \tag{114}$$

[63] Each right eigenvector u of $P^{-1}H$ corresponds to an eigenvector v of $P^{-1/2}HP^{-1/2}$ via the relation $v = P^{1/2}u$.

Meanwhile, we approximate the time-average of the elementwise squared gradient as:

$$\begin{aligned}
\mathbb{E}\left[\nabla L(w_t)^{\odot 2}\right] &= \mathbb{E}\left[\nabla L(w(t)+\delta_t)^{\odot 2}\right] && (w_t = w(t)+\delta_t)\\
&\approx \mathbb{E}\left[(\nabla L(w(t)) + H(w(t))\delta_t)^{\odot 2}\right] && \text{(Taylor expansion)}\\
&= \mathbb{E}\left[\nabla L(w(t))^{\odot 2} + 2\nabla L(w)\odot H(w(t))\delta_t + (H(w(t))\delta_t)^{\odot 2}\right] && \text{(expand the square)}\\
&= \nabla L(w(t))^{\odot 2} + \mathbb{E}\left[(H(w(t))\delta_t)^{\odot 2}\right] && (\mathbb{E}[\delta_t]=0)\\
\implies \mathbb{E}[\nabla L(w_t)^{\odot 2}] &\approx \nabla L(w(t))^{\odot 2} + \mathbb{E}\left[(H(w(t))\delta_t)^{\odot 2}\right]. && (115)
\end{aligned}$$

Recall that RMSProp can be viewed as preconditioned gradient descent with the dynamic preconditioner:

$$P(\nu) := \operatorname{diag}(\sqrt{\nu}/\eta). \tag{116}$$

We therefore define stability for RMSProp by the condition $P(\nu)^{-1}H(w) \preceq 2I$, and we define the critical subspace $\mathcal{U}(w,\nu)$ as the eigenspace of the effective Hessian $P(\nu)^{-1}H(w)$ corresponding to the eigenvalue 2:

$$\mathcal{U}(w,\nu) := \ker\left[P(\nu)^{-1}H(w) - 2I\right].$$

We model RMSProp as oscillating within the critical subspace, i.e. we assume that $\delta_t \in \mathcal{U}(w(t),\nu(t))$. This allows us to simplify eq. (115) as:

$$\begin{aligned}
\mathbb{E}\left[\nabla L(w_t)^{\odot 2}\right] &\approx \nabla L(w(t))^{\odot 2} + \mathbb{E}\left[(H(w(t))\delta_t)^{\odot 2}\right]\\
&= \nabla L(w(t))^{\odot 2} + 4\,\mathbb{E}\left[(P(\nu(t))\delta_t)^{\odot 2}\right] && (H(w)\delta_t = 2P(\nu)\delta_t)\\
&= \nabla L(w(t))^{\odot 2} + \frac{4}{\eta^2}\nu(t)\odot \mathbb{E}\left[\delta_t^{\odot 2}\right] && (P(\nu)=\operatorname{diag}[\nu^{1/2}/\eta])\\
&= \nabla L(w(t))^{\odot 2} + \frac{4}{\eta^2}\nu(t)\odot \mathbb{E}\left[\operatorname{diag}[\delta_t\delta_t^\top]\right] && (v^{\odot 2}=\operatorname{diag}[vv^\top])\\
&= \nabla L(w(t))^{\odot 2} + \frac{4}{\eta^2}\nu(t)\odot \operatorname{diag}[\Sigma(t)]. && (\mathbb{E}[\delta_t\delta_t^\top]=\Sigma(t))
\end{aligned}$$

Based on these time averages, we make the central flow ansatz:

$$\begin{aligned}
\frac{dw}{dt} &= -\frac{\eta}{\sqrt{\nu}}\odot\left[\nabla L(w) + \tfrac{1}{2}\nabla H(w)^\top[\Sigma(t)]\right]\\
\frac{d\nu}{dt} &= \frac{1-\beta_2}{\beta_2}\left[\nabla L(w)^{\odot 2} + \frac{4\nu}{\eta^2}\odot\operatorname{diag}[\Sigma(t)] - \nu\right].
\end{aligned} \tag{117}$$

As with the previous optimizers, to determine $\Sigma(t)$ we impose three conditions for all times t:

1. **PSD:** As a covariance matrix, $\Sigma(t)$ is positive semidefinite (PSD), i.e. $\Sigma(t) \succeq 0$.
2. **Stability:** The effective sharpness remains bounded by 2, i.e. $P(\nu)^{-1}H(w) \preceq 2I$.
3. **Complementarity:** The oscillations are contained within the critical subspace, i.e. $\operatorname{span}\Sigma(t) \subseteq \mathcal{U}(w(t),\nu(t))$.

To express these conditions more concisely, we define the matrix-valued function $A(w,\nu)$ as:

$$A(w,\nu) := 2P(\nu) - H(w),$$

so that stability is $A(w,\nu)\succeq 0$, and the critical subspace is precisely $\mathcal{U}(w,\nu) = \ker A(w,\nu)$. Using this notation, the above conditions can be summarized more compactly as the semidefinite complementarity relation:

$$0 \preceq \Sigma(t) \perp A(w(t),\nu(t)) \succeq 0.$$

Definition 8 (RMSProp Central Flow, Differential Complementarity Problem)**.** We say that $\{w(t), \nu(t)\}_{t\geq 0}$ follow the RMSProp central flow if for almost all $t \geq 0$, they satisfy eq. (117) along with the complementarity relation: $0 \preceq \Sigma(t) \perp A(w(t), \nu(t)) \succeq 0$.

Since this DCP can be expressed in the general form described in Appendix A.6, we can use Lemma 6 to convert it into an equivalent ODE. In particular, if $\widetilde{w} := [w, \nu]^\top$ denotes the augmented state, we can write eq. (117) as:

$$\frac{d\widetilde{w}}{dt} = f(\widetilde{w}) + B(\widetilde{w})[\Sigma] \quad \text{where} \quad f(\widetilde{w}) := \begin{bmatrix} -P(\nu)^{-1}\nabla L(w) \\ \frac{1-\beta_2}{\beta_2}[\nabla L(w)^{\odot 2} - \nu] \end{bmatrix} \quad \text{and} \quad B(\widetilde{w})[\Sigma] := \begin{bmatrix} -\frac{1}{2}P(\nu)^{-1}\nabla H(w)^\top[\Sigma] \\ \frac{1-\beta_2}{\beta_2} \cdot 4P(\nu)^2 \operatorname{diag}\Sigma \end{bmatrix},$$

and we can write the complementarity relation as $0 \preceq \Sigma(t) \perp A(\widetilde{w}(t)) \succeq 0$, where:

$$\nabla A(\widetilde{w}) = [-\nabla H(w), 2\nabla_\nu P(\nu)] \quad \text{where} \quad \nabla_\nu P(\nu)[z] = \operatorname{diag}\left[\frac{1}{2\eta\sqrt{\nu}} \odot z\right] = \frac{1}{2\eta^2}\operatorname{diag}[P(\nu)^{-1}z] \quad \forall z.$$

Therefore, Lemma 6 implies that

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w,\nu)}\left(\alpha(w,\nu), \beta(w,\nu)\right)$$

where the matrix $\alpha(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)$ and tensor $\beta(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$ are defined by:

$$\begin{aligned}
\alpha(w,\nu) &:= \nabla A(\widetilde{w})[f(\widetilde{w})] \\
&= \begin{bmatrix} -\nabla H(w) & 2\nabla_\nu P(\nu) \end{bmatrix} \begin{bmatrix} -P(\nu)^{-1}\nabla L(w) \\ \frac{1-\beta_2}{\beta_2}[\nabla L(w)^{\odot 2} - \nu] \end{bmatrix} \\
&= \nabla H(w)[P(\nu)^{-1}\nabla L(w)] + \frac{1-\beta_2}{\beta_2} \cdot \frac{1}{\eta^2} \cdot \operatorname{diag}\left[P(\nu)^{-1}\left(\nabla L(w)^{\odot 2} - \nu\right)\right]
\end{aligned}$$

and

$$\begin{aligned}
\beta(w,\nu)[\Sigma] &:= \nabla A(\widetilde{w})B(\widetilde{w})[\Sigma] \\
&= \begin{bmatrix} -\nabla H(w) & 2\nabla_\nu P(\nu) \end{bmatrix} \begin{bmatrix} -\frac{1}{2}P(\nu)^{-1}\nabla H(w)^\top[\Sigma] \\ \frac{1-\beta_2}{\beta_2} \cdot 4P(\nu)^2 \operatorname{diag}\Sigma \end{bmatrix} \\
&= \tfrac{1}{2}\nabla H(w)P(\nu)^{-1}\nabla H(w)^\top[\Sigma] + \frac{1-\beta_2}{\beta_2} \cdot \frac{4}{\eta^2} \cdot \operatorname{diag}[P(\nu)\operatorname{diag}[\Sigma]].
\end{aligned}$$

Note that $\beta(\widetilde{w})$ is indeed symmetric PSD, as for gradient descent and Scalar RMSProp:

$$\begin{aligned}
\beta(\tilde{w})[\Sigma, \Sigma'] &= \tfrac{1}{2}\left\langle \nabla H(w)^\top[\Sigma], \nabla H(w)^\top[\Sigma'] \right\rangle_{P(\nu)^{-1}} + \frac{1-\beta_2}{\beta_2} \cdot \frac{4}{\eta^2}\left\langle \operatorname{diag}[\Sigma], \operatorname{diag}[\Sigma'] \right\rangle_{P(\nu)} \\
\implies \beta(\tilde{w})[\Sigma, \Sigma] &= \tfrac{1}{2}\left\| \nabla H(w)^\top[\Sigma] \right\|^2_{P(\nu)^{-1}} + \frac{1-\beta_2}{\beta_2} \cdot \frac{4}{\eta^2}\|\operatorname{diag}[\Sigma]\|^2_{P(\nu)} \geq 0.
\end{aligned}$$

This gives us the ODE formulation of the RMSProp central flow:

Definition 9 (RMSProp Central Flow, ODE Formulation)**.** We say that $\{w(t), \nu(t)\}_{t\geq 0}$ follow the RMSProp central flow if, for almost all t, they satisfy eq. (117) with

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w(t),\nu(t))}\left(\alpha(w(t),\nu(t)), \beta(w(t),\nu(t))\right).$$

Predicting time-averages As with the previous optimizers, the RMSProp central flow can predict the time-average of various quantities, such as the loss or squared gradient norm, along the RMSProp trajectory. Recall that for a function $f(w)$, we use $\bar{f}(t)$ to denote the central flow's prediction for the time-average $\mathbb{E}[f(w_t)]$ at step t. See Appendix A.5 for the derivation of the following statements. We abbreviate $P(t) := P(\nu(t))$.

The prediction $\bar{L}(t)$ for the time-averaged loss at step t is:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \operatorname{tr}\left[P(t)\,\Sigma(t)\right] =: \bar{L}(t). \tag{118}$$

The prediction for the time-averaged squared gradient norm at step t is:

$$\mathbb{E}[\|\nabla L(w_t)\|^2] \approx \|\nabla L(w(t))\|^2 + 4\operatorname{tr}\left[P(t)\,\Sigma(t)\,P(t)^\top\right] =: \overline{\|\nabla L(t)\|^2}. \tag{119}$$

As for predicting the covariance of the oscillations, let $\Sigma(t) = V(t)\,\Lambda(t)\,V(t)^\top$ be the (reduced) eigenvalue decomposition of the rank-k matrix $P^{1/2}(t)\,\Sigma(t)\,P^{1/2}(t)$, where $V(t) \in \mathbb{R}^{d\times k}$ and $\Lambda(t) \in \operatorname{diag}(\mathbb{R}^k)$. Then the central flow predicts that the P-whitened variance of oscillations along the i-th eigenvector of $P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}$ should be equal to the i-th eigenvalue of that matrix:

$$\mathbb{E}\left[\left(v_i(t)^\top P(t)^{1/2}(w_t - w(t))\right)^2\right] = \lambda_i(t). \tag{120}$$

See Appendix A.5 for a discussion of why we predict the P-whitened covariance of oscillations.

Practical implementation When implementing the RMSProp central flow, we treat RMSProp as an instance of a more general class of adaptive preconditioned methods that is described in Appendix A.5. Please refer to that section for details on how we discretize the central flow in practice.

A.4.1 Stationarity analysis

In this appendix, we provide supporting derivations for our analysis of RMSProp's stationary preconditioner.

Stationary preconditioner The RMSProp central flow eq. (117) is a joint flow over (w, ν). However, suppose that the ν dynamics occur "fast" relative to the w dynamics, so that ν is always "fully caught up" to the current w. For any fixed w, solving eq. (117) for the stationarity condition $\frac{d\nu}{dt} = 0$ gives the condition:

$$\nu = \nabla L(w)^{\odot 2} + \frac{4}{\eta^2}\,\nu \odot \operatorname{diag}[\Sigma]. \tag{121}$$

In addition, by the PSD, stability, and complementarity conditions, we have:

$$0 \preceq \Sigma \perp 2P(\nu) - H(w) \succeq 0 \quad \text{where} \quad P(\nu) := \operatorname{diag}\left(\tfrac{\sqrt{\nu}}{\eta}\right). \tag{122}$$

We will now show that for any w, there is a *unique* pair ν, Σ that satisfies eqs. (121) and (122). We will denote this pair as $\overline{\nu}(w)$ and $\overline{\Sigma}(w)$. We will further show that the corresponding preconditioner $\operatorname{diag}(\sqrt{\overline{\nu}(w)}/\eta)$ is the unique optimum to the following convex optimization problem over diagonal preconditioners:

$$\underset{P \text{ diagonal},\, P\succeq 0}{\arg\min} \quad \operatorname{tr}(P) + \tfrac{1}{\eta^2}\|\nabla L(w)\|^2_{P^{-1}} \quad \text{such that} \quad H(w) \preceq 2P, \tag{123}$$

where $\|v\|^2_{P^{-1}} := v^\top P^{-1} v$. We will denote this preconditioner as $\overline{P}(w)$.

We show this in two parts. First, in Proposition 5 we prove that if ν, Σ satisfy eqs. (121) and (122), then $P(\nu)$ is an optimum for eq. (123). Then, in Proposition 6 we show that the solution to eq. (123) is unique, implying that this must be the *unique* optimum. At the end of this section, we describe how to numerically compute $\overline{P}(w)$ and $\bar{\nu}(w)$ when $H(w)$ is so large that it can only be feasibly accessed via matrix-vector products.

Proposition 5. Define $P(\nu) := \operatorname{diag}(\sqrt{\nu}/\eta)$. For any $g \in \mathbb{R}^d, H \in \operatorname{Sym}(\mathbb{R}^d)$, if $\nu \in \mathbb{R}^d, \Sigma \in \operatorname{Sym}(\mathbb{R}^d)$ satisfy:

$$\nu = g^{\odot 2} + \tfrac{4}{\eta^2}\nu \odot \operatorname{diag}(\Sigma) \tag{124}$$

$$0 \preceq \Sigma \perp 2P(\nu) - H \succeq 0, \tag{125}$$

then $P(\nu)$ is an optimum for the convex program:

$$\underset{P \text{ diagonal},\, P\succeq 0}{\arg\min} \quad \operatorname{tr}(P) + \tfrac{1}{\eta^2} g^T P^{-1} g \quad \text{such that} \quad H \preceq 2P. \tag{126}$$

Proof. Parameterizing $P = \mathrm{diag}(p)$ for a vector $p \in \mathbb{R}^d$, the convex program eq. (126) can be written as:

$$\min_{p \in \mathbb{R}^d,\, p \geq 0} \sum_{i=1}^{d} p_i + \frac{1}{\eta^2} \frac{g_i^2}{p_i} \quad \text{such that} \quad H \preceq 2\,\mathrm{diag}(p). \tag{127}$$

Introducing a dual variable $Z \succeq 0$ for the semidefinite constraint, the Lagrangian is:

$$L(p, Z) = \sum_{i=1}^{d} \left[p_i + \frac{1}{\eta^2} \frac{g_i^2}{p_i} \right] - \langle Z, 2\,\mathrm{diag}(p) - H \rangle \,. \tag{128}$$

Therefore, the KKT conditions are:

$$\underbrace{1 - \frac{1}{\eta^2} \frac{g_i^2}{p_i^2} - 2Z_{ii} = 0 \quad \forall i}_{\text{stationarity}}, \quad \underbrace{H \preceq 2\,\mathrm{diag}(p)}_{\text{primal feasibility}}, \quad \underbrace{Z \succeq 0}_{\text{dual feasibility}}, \quad \underbrace{2\,\mathrm{diag}(p) - H \perp Z}_{\text{complementary slackness}}, \tag{129}$$

as well as $p \geq 0$. We claim that if (v, Σ) satisfy eqs. (124) and (125), then $(p, Z) = (\frac{\sqrt{\nu}}{\eta}, \frac{2}{\eta^2}\Sigma)$ solve these KKT conditions. First, elementwise dividing both sides of eq. (124) by ν gives:

$$1 - \frac{g^{\odot 2}}{\nu_i} - \left(\frac{4}{\eta^2} \right) \Sigma_{ii} = 0 \quad \forall i \tag{130}$$

which is equivalent to the stationary condition in eq. (129) after substituting $p = \sqrt{\nu}/\eta$ and $Z = \frac{2}{\eta^2}\Sigma$. Next, the remaining parts of eq. (129) are implied by eq. (125). Finally, we must have $\sqrt{\nu} \geq 0$ or $P(\nu)$ would be imaginary. $\square$

We now prove that the solution to the optimization problem eq. (126) is unique. A custom proof is needed because, while both the objective and constraints of eq. (126) are convex, the objective is not *strictly* convex.

Proposition 6. For any $g \in \mathbb{R}^d, H \in \mathrm{Sym}(\mathbb{R}^d)$, the solution to eq. (123) is unique.

Proof. Assume there are two minimizers P, P' and let $p := \mathrm{diag}(P), \delta := \mathrm{diag}(P' - P)$. Then by convexity, $\mathrm{diag}[p + \epsilon\delta]$ also minimizes eq. (35) for any $\epsilon \leq 1$. Therefore, differentiating the objective function in this direction gives:

$$\sum_i \delta_i \left[1 - \frac{1}{\eta^2} \frac{g_i^2}{p_i^2} \right] = 0. \tag{131}$$

Taking another derivative implies that:

$$\sum_i \frac{g_i^2}{p_i^3} \delta_i^2 = 0. \tag{132}$$

This implies that $\delta_i = 0$ in any direction where $g_i \neq 0$. Let I be the set of indices for which $g_i \neq 0$, and for any vector p, let p_I denote the vector p restricted to the indices in I. Define the linear map g by

$$g[v_I]_i := \begin{cases} v_i & i \in I \\ p_i & i \notin I \end{cases}. \tag{133}$$

In other words, g takes a reduced vector v_I and fills in the missing entries with p. Next, define the operator $\mathcal{A}$ by

$$\mathcal{A}^T[v_I] = \mathrm{diag}[g[v_I]] \oplus \mathrm{diag}[v_I] \tag{134}$$

where $\oplus$ represents the direct sum. Then both p_I, p'_I minimize the following reduced SDP:

$$\min_{p_I} \sum_{i \in I} p_i \quad \text{such that} \quad \tfrac{1}{2} H(w) \oplus 0_{|I| \times |I|} \preceq \mathcal{A}^T(p). \tag{135}$$

Now we apply de Carli Silva and Tunçel (2018, Proposition 1) with $(\mathcal{A}, \mathbf{1}_{|I|})$. First, note that $\mathcal{A}[I_{d+|I|}] = 2\mathbf{1}_{|I|}$ which satisfies the first condition. Next, for any $y \neq 0$, we can take $z = |y|$ to satisfy the second condition, as in the proof of (de Carli Silva and Tunçel, 2018, Corollary 2) Therefore $p_I = p'_I$, and as we have already shown equality on I^c, we must have $p = p'$. □

Stationary flow Suppose that the ν dynamics (preconditioner adaptation) happen *infinitely fast* relative to the w dynamics (optimization), so that we can treat ν as always being fixed at its current stationary value $\overline{\nu}(w)$. This motivates the stationary flow:

Definition 10 (RMSProp Stationary Flow)**.** We say that $\{w(t)\}_{t \geq 0}$ follow the RMSProp stationary flow if, for almost all t, they satisfy

$$\frac{dw}{dt} = -\frac{\eta}{\sqrt{\overline{\nu}(w)}} \odot \left[\nabla L(w) + \tfrac{1}{2} \nabla H(w)^\top [\Sigma(t)]\right] \tag{136}$$

with

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w(t), \overline{\nu}(w(t)))} \left(\alpha(w(t), \overline{\nu}(w(t))), \beta(w(t), \overline{\nu}(w(t)))\right).$$

Note that $\Sigma(t)$ is defined as the solution to an SDCP, and is *not*, in general, equal to $\bar{\Sigma}(w(t))$. That said, in the limit of $\beta_2 \to 0$, $\Sigma(t)$ does become $\bar{\Sigma}(w(t))$. To see this, note that as $\beta_2 \to 0$, we have:

$$\begin{aligned}
\alpha(w, \bar{\nu}(w)) &\to \frac{1-\beta_2}{\beta_2} \frac{1}{\eta^2} \operatorname{diag}\left[P(\bar{\nu}(w))^{-1} \left(\nabla L(w)^{\odot 2} - \bar{\nu}(w)\right)\right] && (\beta_2 \to 0) \\
&= \frac{1-\beta_2}{\beta_2} \frac{1}{\eta^2} \operatorname{diag}\left[P(\bar{\nu}(w))^{-1} \left(-\frac{4\nu}{\eta^2} \odot \operatorname{diag}[\bar{\Sigma}]\right)\right] && \text{(stationarity, i.e. eq. (121))} \\
&= -\frac{1-\beta_2}{\beta_2} \frac{4}{\eta^3} \sqrt{\overline{\nu}(w)} \odot \operatorname{diag}[\bar{\Sigma}(w)]. && (P^{-1}(\nu) = \eta/\sqrt{\nu})
\end{aligned}$$

$$\begin{aligned}
\beta(w, \bar{\nu}(w))[\bar{\Sigma}(w)] &\to \frac{1-\beta_2}{\beta_2} \frac{4}{\eta^2} P(\bar{\nu}(w)) \operatorname{diag}[\bar{\Sigma}(w)] && (\beta_2 \to 0) \\
&= \frac{1-\beta_2}{\beta_2} \frac{4}{\eta^3} \sqrt{\overline{\nu}(w)} \odot \operatorname{diag}[\bar{\Sigma}(w)]. && (P(\nu) = \sqrt{\nu}/\eta)
\end{aligned}$$

This implies that:

$$\alpha(w, \bar{\nu}(w)) + \beta(w, \bar{\nu}(w))[\bar{\Sigma}(w)] \to 0.$$

Thus, $\bar{\Sigma}(w)$ is the solution to the SDCP that defines $\Sigma(t)$. This suggests the approximation:

$$\frac{dw}{dt} = -\frac{\eta}{\sqrt{\overline{\nu}(w)}} \odot \left[\nabla L(w) + \tfrac{1}{2} \nabla H(w)^\top [\bar{\Sigma}(w)]\right]. \tag{137}$$

Note that we expect this to be a useful model even when β_2 is far from the $\beta_2 = 0$ limit.

Limit of large η In the limit of large η, the second term in the objective eq. (123) vanishes, and the stationary preconditioner $\overline{P}(w)$ tends towards the *minimum-trace, diagonal stable preconditioner*, which we denote $\hat{P}(w)$:

$$\hat{P}(w) := \operatorname*{arg\,min}_{P \text{ diagonal}, P \succeq 0} \operatorname{tr}(P) \quad \text{such that} \quad H(w) \preceq 2P. \tag{138}$$

Note that the dual to this semidefinite program is the following semidefinite program:

$$\hat{Z}(w) := \arg\max_{Z \succeq 0} \langle Z, H(w) \rangle \quad \text{subject to} \quad Z_{ii} \leq \tfrac{1}{2} \tag{139}$$

and a primal/dual optimal pair $\hat{P}(w), \hat{Z}(w)$ must satisfy the KKT conditions:

$$\hat{Z}_{ii}(w) = \tfrac{1}{2}, \quad 0 \preceq \hat{Z}(w) \perp 2\hat{P}(w) - H(w) \succeq 0. \tag{140}$$

In the limit of large η, the stationary EMA $\bar{\nu}(w)$ and the stationary oscillation covariance $\bar{\Sigma}(w)$ become:

$$\bar{\nu}(w) \to \eta \operatorname{diag}[\hat{P}(w)]^{\circ 2} \quad \text{and} \quad \bar{\Sigma}(w) \to \frac{\eta^2}{2}\hat{Z}(w). \tag{141}$$

As a result, the approximation eq. (137) can be shown to be equivalent to:

$$\frac{dw}{dt} = -\hat{P}(w)^{-1} \odot \Big[\nabla L(w) + \tfrac{\eta^2}{4}\nabla \operatorname{tr} \hat{P}(w)\Big]. \tag{142}$$

This is because $\operatorname{tr} \hat{P}(w) = \left\langle \hat{Z}(w), H(w) \right\rangle$ by duality, so by Danskin's theorem:

$$\begin{aligned} \nabla \operatorname{tr} \hat{P}(w) &= \nabla H(w)^\top [\hat{Z}(w)] \\ &= \tfrac{2}{\eta^2} \nabla H(w)^\top [\bar{\Sigma}(w)]. \end{aligned}$$

Hence, $\frac{\eta^2}{4}\nabla \operatorname{tr} \hat{P}(w) = \frac{1}{2}\nabla H(w)^\top [\bar{\Sigma}(w)]$, and eq. (142) follows.

Connection to MaxCut Interestingly, the SDP eq. (139) is precisely the SDP relaxation of MaxCut (Goemans and Williamson, 1995) where the Laplacian matrix of the graph is given by $\frac{1}{2}H(w)$. Meanwhile, the SDP eq. (138) that defines $\hat{P}$ is the dual to the MaxCut SDP relaxation.

Numerically solving for the stationary preconditioner When the problem dimension d is small, the optimization problem eq. (123) can be solved exactly using a standard convex solver, e.g. cvxpy. But when d is large (e.g. the number of weights in a reasonably sized neural network), solving eq. (123) exactly is not practical, as it is not even practical to materialize the matrix $H \in \mathbb{R}^{d \times d}$. Therefore, we instead solve eq. (123) using a fixed point iteration which only requires access to H using matrix-vector products.

We parameterize Σ in the factorized form $\Sigma = DD^T$ where $D \in \mathbb{R}^{d \times r}$ and r is intended to be at least as large as the rank of Σ. This is similar to the Burer-Monteiro factorization (Burer and Monteiro, 2005). We start for a random initial guess for D and then iteratively update D and ν by:

$$\nu \leftarrow g^{\odot 2} + (HD)^{\odot 2}\mathbf{1}, \tag{143}$$

$$D \leftarrow \frac{\eta}{2} \operatorname{diag}[\nu^{-1/2}] HD. \tag{144}$$

where the second update uses the ν that was just computed in the first update.

If this update scheme reaches a fixed point (D, ν), then we have:

$$\nu = g^{\odot 2} + (HD)^{\odot 2}\mathbf{1}, \tag{145}$$

$$HD = \tfrac{2}{\eta} \operatorname{diag}[\nu^{1/2}] D. \tag{146}$$

If (D, ν) satisfy these two conditions, as well as the stability condition $H \preceq 2\operatorname{diag}(\sqrt{\nu}/\eta)$, then it can be shown that $\Sigma = DD^T$ and ν satisfy eqs. (124) and (125). Indeed, $\Sigma \succeq 0$ holds by construction, eq. (124) follows by substituting

eq. (146) into eq. (145), and eq. (146) implies $[2\,\mathrm{diag}(\sqrt{\nu}/\eta) - H]D = 0$ which implies $2\,\mathrm{diag}(\sqrt{\nu}/\eta) - H \perp \Sigma$ provided that the stability condition holds.

Thus, if the update scheme reaches a fixed point eqs. (145) and (146), and if the stability condition $H \preceq 2\,\mathrm{diag}(\sqrt{\nu}/\eta)$ is also satisfied there, then we know that $P = \mathrm{diag}(\sqrt{\nu}/\eta)$ solves the optimization problem eq. (123).

Empirically, we observe that this update scheme does reach a fixed point in practice. We moreover observe that if r is sufficiently large (in particular, if it is as large as the rank of the true Σ), then the stability condition is satisfied at this fixed point, implying that the corresponding preconditioner indeed solves eq. (123). On the other hand, we observe that if r is too small (less than the rank of the true Σ), then while the update scheme converges to a fixed point, the stability condition is not satisfied there.

Algorithm 1: Solving for the Stationary Preconditioner

Input: Gradient $g \in \mathbb{R}^d$, Hessian-vector oracle $v \mapsto Hv$, learning rate η, rank parameter r, number of steps nsteps, tolerance parameter tol_ν

Output: ν, P, and D

Initialize $D \in \mathbb{R}^{d\times r}$ with standard normal entries;

for $i = 1$ **to** *nsteps* **do**

  if *i > 1* **then**

    $\nu_{\text{prev}} \leftarrow \nu$;

  end

  $\nu \leftarrow g^{\odot 2} + \texttt{sum\_rows}((HD)^{\odot 2})$;

  $D \leftarrow \frac{\eta}{2}\mathrm{diag}[\nu^{-1/2}]HD$;

end

if $\frac{\|\nu - \nu_{prev}\|_2}{\|\nu\|_2} \geq tol_\nu$ **then**

  return "Error: more steps needed";

end

$P \leftarrow \sqrt{\nu}/\eta$;

if $\lambda_1(P^{-1/2}HP^{-1/2}) > 2$ **then**

  return "Error: higher r needed";

end

return ν, P, D;

A.5 General Class of Adaptive Preconditioned Methods

In this section, we derive a central flow for a general class of adaptive preconditioned methods that subsumes gradient descent, Scalar RMSProp, and RMSProp as special cases. In these special cases, this flow will reduce to the central flows that we have already derived, and whose accuracy we have verified empirically. However, we do *not* claim that the central flow derived in this section will be empirically accurate for *any* method within this class. Rather, we include this section because it allows us treat all three considered optimizers in a unified manner, and to easily generalize our central flows to minor variants of the same algorithms (e.g. gradient descent with a learning rate schedule, RMSProp with bias correction). Our implementation in code is based on this formulation.

We consider methods which update some "optimizer state" $\nu \in \mathbb{R}^{d_\nu}$ based on the current gradient, and then take a preconditioned gradient step using some preconditioner $P(\nu)$ that is derived from this state:

$$\nu_t = \nu_{t-1} + G(\nu_{t-1}, \nabla L(w_t)), \quad w_{t+1} = w_t - P(\nu_t)^{-1}\nabla L(w_t). \tag{147}$$

Here, $G : \mathbb{R}^{d_\nu} \times \mathbb{R}^d \to \mathbb{R}^{d_\nu}$ determines how the optimizer state $\nu \in \mathbb{R}^{d_\nu}$ is updated based on the gradient, and $P : \mathbb{R}^{d_\nu} \to \mathrm{Sym}(\mathbb{R}^d)$ determines how the optimizer state affects the preconditioner.

This formulation is very general and includes a wide variety of optimizers including:

- *Vanilla GD:* ignore ν and set $P = \eta^{-1} I$.
- *GD with a learning rate schedule $\eta(t)$:* Set $G(\nu, w) = 1$ so that $\nu(t) = t$, and $P(t) = \eta(t)^{-1} I$
- *Vanilla RMSProp:*[64] Set $G(\nu, g) = (1 - \beta_2)[g^{\odot 2} - \nu]$ and $P(\nu) = \mathrm{diag}[\sqrt{\nu}/\eta]$
- *RMSProp with ϵ, bias correction, and learning rate schedule $\eta(t)$:* Set $\nu = [v, t]$, $G([v, t], w) = [(1 - \beta_2)[g^{\odot 2} - v], 1]$ and define

$$P([v, t]) = \frac{1}{\eta(t)} \mathrm{diag}\left[\sqrt{\frac{v}{1 - \beta_2^t}} + \epsilon\right].$$

Note that this trick of embedding t into the state variable ν allows us to automatically derive central flows for any smooth hyperparameter schedule (e.g. $\eta(t), \beta_2(t), \epsilon(t)$) as a simple corollary.

Remark 3. Not all algorithms of the form eq. (147) are necessarily sensible optimizers. Indeed, we will see below that the central flow is only well-defined if G and P satisfy a certain condition (Remark 4). Thus, it may make sense for future work to further restrict the formulation eq. (147).

The stability of the algorithm eq. (147) requires $P(\nu)^{-1} H(w) \preceq 2I$ or equivalently $H(w) \preceq 2P(\nu)$. We define

$$A(w, \nu) := 2P(\nu) - H(w), \tag{148}$$

so that stability is equivalent to $A(w, \nu) \succeq 0$. We define the critical subspace by $\mathcal{U}(w, \nu) := \ker A(w, \nu)$.

To derive the central flow, we model $w_t = w(t) + \delta_t$ with $\mathbb{E}[\delta_t] = 0$, $\mathbb{E}[\delta_t \delta_t^\top] = \Sigma(t)$, and $\delta_t \in \ker A(w(t), \nu(t))$. For ease of notation, we will sometimes use $g(w)$ as a shorthand for the gradient $\nabla L(w)$.

We will first compute the time-average of G, i.e. $\mathbb{E}[G(\nu, g(w_t)] = \mathbb{E}[G(\nu, g(w(t) + \delta_t)]$. A first-order Taylor expansion of g around any point w yields:

$$g(w + \delta) \approx g(w) + H(w)\, \delta.$$

Meanwhile, a second-order Taylor expansion of G in its second argument yields:

$$G(v, g + \Delta g) \approx G(\nu, g) + \nabla_g G(\nu, g)^\top \Delta g + \tfrac{1}{2} \nabla_g^2 G(\nu, g)[\Delta g\, \Delta g^\top], \tag{149}$$

where $\nabla_g^2 G(v, \Delta g) \in \mathbb{R}^{d_\nu} \otimes \mathrm{Sym}(\mathbb{R}^d)$ is a tensor that represents the Hessian of each entry of G. Putting these together, with $\Delta g = H(w)\delta$, we have:

$$G(\nu, g(w + \delta)) \approx G(\nu, g(w)) + \nabla_g G(\nu, g)^\top H(w)\delta + \tfrac{1}{2} \nabla_g^2 G(\nu, g)[H(w)\, \delta\delta^\top\, H(w)],$$

Taking the time-average over δ with $\mathbb{E}[\delta] = 0$ and $\mathbb{E}[\delta\delta^\top] = \Sigma$ yields:

$$\mathbb{E}[G(v, g(w + \delta)] \approx G(\nu, g(w)) + \nabla_g^2 G(\nu, g(w))[H(w)\, \Sigma\, H(w)]. \tag{150}$$

Since $H(w)\Sigma = 2P(\nu)\Sigma$, we have $H(w)\Sigma H(w) = 4P(\nu)\Sigma P(\nu)$, and therefore the second term becomes:

$$\mathbb{E}[G(v, g(w + \delta)] \approx G(\nu, g(w)) + 4\nabla_g^2 G(\nu, g(w))[P(\nu)\, \Sigma\, P(\nu)]. \tag{151}$$

These motivate the central flow ansatz:

$$\begin{aligned} \frac{dw}{dt} &= -P(\nu)^{-1}\Big[\nabla L(w) + \tfrac{1}{2} \nabla H(w)^\top [\Sigma]\Big] \\ \frac{d\nu}{dt} &= G(\nu, g(w)) + 2\nabla_g^2 G(\nu, g(w))[P(\nu)\, \Sigma\, P(\nu)]. \end{aligned} \tag{152}$$

[64] This formalism doesn't directly handle our small β_2 correction of $\beta_2 \to \frac{1-\beta_2}{\beta_2}$. We view this correction as "orthogonal" in the sense that it comes when deriving a stable flow analogue of the discrete time update $\nu_t = \nu_{t-1} + G(\nu_{t-1}, \nabla L(w_t))$ when this takes the form of an EMA.

We say that $\{w(t), \nu(t), \Sigma(t)\}_{t\geq 0}$ satisfy the DCP formulation of the central flow if for almost all $t \geq 0$ they satisfy eq. (152) along with the complementarity relation:

$$0 \preceq \Sigma(t) \perp A(w(t), \nu(t)) \succeq 0.$$

As for gradient descent, Scalar RMSProp, RMSProp, we can invoke Lemma 6 to obtain an equivalent ODE formulation that makes $\Sigma(t)$ explicit. We begin by writing eq. (152) as:

$$\frac{d\widetilde{w}}{dt} = f(\widetilde{w}) + B(\widetilde{w})[\Sigma] \quad \text{where} \quad f(\widetilde{w}) := \begin{bmatrix} -P(\nu)^{-1}\nabla L(w) \\ G(\nu, g(w)) \end{bmatrix} \quad \text{and} \quad B(\widetilde{w})[\Sigma] := \begin{bmatrix} -\frac{1}{2}P(\nu)^{-1}\nabla H(w)^\top[\Sigma] \\ 2\nabla_w^2 G(\nu, g(w))\,[P(\nu)\,\Sigma\,P(\nu)] \end{bmatrix}.$$

where the complementarity relation is $0 \preceq \Sigma \perp A(\widetilde{w}) \succeq 0$ with ∇A given by:

$$\nabla A(\widetilde{w}) = [-\nabla H(w), 2\nabla P(\nu)].$$

Therefore, Lemma 6 implies that:

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w,\nu)}\left(\alpha(w,\nu), \beta(w,\nu)\right),$$

where the matrix $\alpha(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)$ and tensor $\beta(w,\nu) \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}$ are defined by:

$$\begin{aligned}
\alpha(\widetilde{w}) &:= \nabla A(\widetilde{w})[f(\widetilde{w})] \\
&= \begin{bmatrix} -\nabla H(w) & 2\nabla P(\nu) \end{bmatrix} \begin{bmatrix} -P(\nu)^{-1}\nabla L(w) \\ G(\nu, g(w)) \end{bmatrix} \\
&= \nabla H(w)[P(\nu)^{-1}\nabla L(w)] + 2\nabla P(\nu)[G(\nu, g(w))]
\end{aligned}$$

and

$$\begin{aligned}
\beta(\widetilde{w})[\Sigma] &:= \nabla A(\widetilde{w}) B(\widetilde{w})[\Sigma] \\
&= \begin{bmatrix} -\nabla H(w) & 2\nabla P(\nu) \end{bmatrix} \begin{bmatrix} -\frac{1}{2}P(\nu)^{-1}\nabla H(w)^\top[\Sigma] \\ 2\nabla_w^2 G(\nu, g(w))[P(\nu)\Sigma P(\nu)] \end{bmatrix} \\
&= \tfrac{1}{2}\nabla H(w) P(\nu)^{-1} \nabla H(w)^\top[\Sigma] + 4\nabla P(\nu)\ \nabla_w^2 G(\nu, g(w))\,[P(\nu)\,\Sigma\,P(\nu)].
\end{aligned}$$

Remark 4. For gradient descent, Scalar RMSProp, and RMSProp, the β we derived was always symmetric PSD. However, at the current level of generality this is not always true:

$$\beta(w,\nu)[\Sigma, \Sigma'] = \tfrac{1}{2}\left\langle \nabla H(w)^\top[\Sigma], \nabla H(w)^\top[\Sigma'] \right\rangle_{P(\nu)^{-1}} + \left\langle \nabla P(\nu)^\top[\Sigma], \nabla_w^2 G(\nu, g(w))[P(\nu)\Sigma' P(\nu)] \right\rangle.$$

While the first term is symmetric in Σ, Σ', the second is not necessarily symmetric, so results about existence and uniqueness for solutions to the SDCP may no longer hold.

Predicting time-averages The central flow's predictions for time-averages can be computed as follows. Recall that for a function $f(w)$, we use $\bar{f}(t)$ to denote the central flow's prediction for the time-average $\mathbb{E}[f(w_t)]$ at step t. In what follows, we abbreviate $P(t) := P(\nu(t))$.

The prediction for the time-averaged loss at time t is:

$$\begin{aligned}
\mathbb{E}[L(w_t)] &\approx \mathbb{E}\left[L(w(t)) + \nabla L(w(t))^\top \delta_t + \tfrac{1}{2}\delta_t^\top H(w(t))\delta_t\right] && \text{(Taylor expansion)} \\
&= L(w(t)) + \tfrac{1}{2}\left\langle H(w(t)), \Sigma(t)\right\rangle && (\mathbb{E}[\delta_t] = 0) \\
&= L(w(t)) + \tfrac{1}{2}\,\mathrm{tr}\left[2P(t)\Sigma\right] && (H\Sigma = 2P\Sigma) \\
&= L(w(t)) + \mathrm{tr}\left[P(t)\,\Sigma(t)\right] \\
&=: \bar{L}(t). && (153)
\end{aligned}$$

Similarly, the prediction for the time-averaged squared gradient norm at time t is:

$$\begin{aligned}\mathbb{E}[\|\nabla L(w_t)\|^2] &\approx \mathbb{E}\left[\|\nabla L(w(t)) + H(w(t))\delta_t\|^2\right] \\ &= \|\nabla L(w(t))\|^2 + \left\langle H^2(w(t)), \Sigma(t)\right\rangle \\ &= \|\nabla L(w(t))\|^2 + 4\,\mathrm{tr}\left[P(t)\,\Sigma(t)\,P(t)^\top\right] \\ &=: \overline{\|\nabla L(t)\|^2}. \end{aligned} \tag{154}$$

The central flow can also predict the covariance of the oscillations. The natural basis to examine the oscillations is the one in which the dynamics of each coordinate are decoupled under preconditioned gradient descent on the local quadratic Taylor approximation. Thus, we define the P-whitened displacement $\hat{\delta}_t := P(t)^{1/2}\delta_t = P(t)^{1/2}(w_t - w(t))$, and the P-whitened covariance matrix of the oscillations $\mathbb{E}[\hat{\delta}_t\hat{\delta}_t^\top] = P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}$. Let $V(t)\,\Lambda(t)\,V^\top(t)$ be the eigenvalue decomposition of $P(t)^{1/2}\,\Sigma\,P(t)^{1/2}$, and define $x_t := V(t)^\top\hat{\delta}_t = V(t)^\top P(t)^{1/2}(w_t - w(t))$ as the P-whitened displacement between the discrete optimizer and the central flow along the top eigenvectors $V(t)$. Then the central flow predicts that:

$$\mathbb{E}[x_t x_t^\top] = V(t)^\top P(t)^{1/2}\,\mathbb{E}[\delta_t\delta_t^\top]P(t)^{1/2}V(t) = V(t)^\top\,P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}\,V(t) = \Lambda(t).$$

In particular, the P-whitened variance of oscillations along the i-th eigenvector of $P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}$ is predicted to be the i-th eigenvalue of that matrix:

$$\mathbb{E}\left[\left(v_i(t)^\top P(t)^{1/2}(w_t - w(t))\right)^2\right] = \lambda_i(t). \tag{155}$$

Basis-dependent version We can use the general recipe given in Appendix A.6 to obtain a basis-dependent version of the central flow ODE that can be computed in time linear in d, when the preconditioner is diagonal.

Fix a time t, and we will often abbreviate $w(t)$, $\nu(t)$, and $P(\nu(t))$ as w, ν and P. Let $U \in \mathbb{R}^{d\times k}$ be a basis for the critical subspace $\mathcal{U}(w,\nu)$. Define $H_U(w) := U^\top H(w)U \in \mathrm{Sym}(\mathbb{R}^k)$ and $P_U(\nu) := U^\top P(\nu)U \in \mathrm{Sym}(\mathbb{R}^k)$, as well as their gradients $\nabla H_U(w) \in \mathrm{Sym}(\mathbb{R}^k)\otimes\mathbb{R}^d$ and $\nabla P_U(\nu) \in \mathrm{Sym}(\mathbb{R}^k)\otimes\mathbb{R}^{d_\nu}$. Explicitly:

$$\nabla H_U(w)_{ij} = \nabla_w[u_i^\top H(w)u_j] \quad \text{and} \quad \nabla P_U(\nu)_{ij} = \nabla_\nu[u_i^\top P(\nu)u_j]. \tag{156}$$

Similarly, let $\nabla^2 G_{PU}(\nu, g(w)) \in \mathbb{R}^{d_\nu}\otimes\mathrm{Sym}(\mathbb{R}^k)$ be the tensor defined as:

$$\nabla^2 G_{PU}(\nu, g(w))_{q,ij} = (Pu_i)^\top\,\nabla_g^2 G(\nu, g(w))_q\,(Pu_j). \tag{157}$$

Then the central flow takes the form:

$$\frac{dw}{dt} = -P(\nu)^{-1}\Big[\nabla L(w) + \tfrac{1}{2}\nabla H_U(w)^\top[X]\Big] \tag{158}$$

$$\frac{d\nu}{dt} = G(\nu, g(w)) + 2\nabla^2 G_{PU}(\nu, g(w))[X] \tag{159}$$

$$X \in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U(w,\nu), \beta_U(w,\nu)) \tag{160}$$

$$\alpha_U(w,\nu) = \nabla H_U(w)\big[P(\nu)^{-1}\nabla L(w)\big] + 2\nabla P_U(\nu)[G(\nu, g(w))] \tag{161}$$

$$\beta_U(w,\nu) = \tfrac{1}{2}\nabla H_U(w)P(\nu)^{-1}\nabla H_U(w)^\top + 4\nabla P_U(\nu)\,\nabla^2 G_{PU}(\nu, g(w)). \tag{162}$$

Suppose that we pick the basis U to be orthonormal w.r.t the preconditioner P, i.e. $U^\top PU = I$. Then the central flow's prediction eq. (153) for the time-averaged train loss can be efficiently computed as:

$$\begin{aligned}\bar{L}(t) &:= L(w(t)) + \mathrm{tr}\left[PUXU^\top\right] \\ &= L(w(t)) + \mathrm{tr}\left[XU^\top PU\right] \\ &= L(w(t)) + \mathrm{tr}\,[X]. \end{aligned} \tag{163}$$

Similarly, the prediction eq. (154) for the time-averaged squared gradient norm can be computed as:

$$\overline{\|\nabla L\|^2}(t) = \|\nabla L(w(t))\|^2 + 4\,\mathrm{tr}\left[(PU)X(PU)^\top\right]. \tag{164}$$

As for the oscillation covariance, we can compute both sides of eq. (155) without materializing $\Sigma(t)$ in full. If $X = U_X\,\Lambda(t)\,U_X^\top$ is the eigenvalue decomposition of X, and $V(t) := P^{1/2}UU_X$, then $V(t)\,\Lambda(t)\,V(t)^\top$ is the eigenvalue decomposition of $P^{1/2}\,\Sigma(t)\,P^{1/2}$. Note that X and U_X are dependent on the basis U, while $V(t)$ and $\Lambda(t)$ are independent of U.

Warning: variation in notation Although gradient descent can be cast as an instance of an adaptive preconditioned method (with $P = \eta^{-1}I$), the U defined here is different from the U in Appendix A.2.4, as the U there was orthonormal, i.e. $U^\top U = I$, whereas the U here is orthonormal w.r.t the preconditioner P, i.e. $U^\top PU = I$ or $U^\top U = \eta I$. As a result, the X defined here differs from the X defined in Appendix A.2.4 by a factor of η.

A.5.1 Discretizing the central flow for a generic adaptive preconditioned method

We describe our general procedure for discretizing DCPs in Appendix A.6.1. Let us now describe how this general procedure specializes to the case of our generic adaptive preconditioned method.

We use $w^{(t)}$, $\nu^{(t)}$, and $\Sigma^{(t)}$ to denote our estimate for the central flow's $w(t)$, $\nu(t)$, and $\Sigma(t)$. Let $\epsilon > 0$ be the discretization step size, e.g. $\epsilon = 0.25$. For some tolerance $\tau > 0$, e.g. $\tau = 0.05$, we will regard eigenvalues of the effective Hessian $P(\nu)^{-1}H(w)$ greater than $2 - \tau$ as those which might become unstable in the next discretization time step.

At each discretization step, we do the following. Abbreviate $w = w^{(t)}$ and $P = P(\nu^{(t)})$. First, we compute all eigenvalues of the effective Hessian $P^{-1}H(w)$ that are greater than $2 - \tau$, as well as the corresponding eigenvectors. Let k be the number of such eigenvalues, let $D \in \mathrm{diag}(\mathbb{R}^k)$ be a diagonal matrix containing such eigenvalues on the diagonal, and let $U \in \mathbb{R}^{d\times k}$ be the corresponding eigenvectors, normalized so that they are orthonormal w.r.t P, that is, $U^\top PU = I$. For example, one could set $U = P^{-1/2}\tilde{U}$, where the columns of $\tilde{U} \in \mathbb{R}^{d\times k}$ are orthonormal eigenvectors of $P^{-1/2}H(w)P^{-1/2}$.

Then, we compute the tensors ∇H_U, ∇P_U, $\nabla^2 G_{PU}$ as in eqs. (156) and (157), though note that U now refers to a basis of eigenvectors whose eigenvalues are *almost* 2 rather than exactly equal to 2. Then, we compute α_U and β_U as in eqs. (161) and (162). Then, we solve the following k-dimensional SDCP:

$$X^{(t)} = \mathrm{SDCP}_{\mathbb{R}^k}(2I - D + \epsilon\,\alpha_U, \epsilon\,\beta_U), \tag{165}$$

so that $\Sigma^{(t)} = UX^{(t)}U^\top$. Then, we update the weights and optimizer state via:

$$w^{(t+\epsilon)} = w^{(t)} - \epsilon\,P(\nu^{(t)})^{-1}\left[\nabla L(w^{(t)}) + \tfrac{1}{2}\nabla H_U^\top[X^{(t)}]\right] \tag{166}$$

$$\nu^{(t+\epsilon)} = \nu^{(t)} + \epsilon\left[G(\nu^{(t)}, g(w^{(t)})) + 2\nabla^2 G_{PU}(\nu^{(t)}, g(w^{(t)}))[X^{(t)}]\right]. \tag{167}$$

To predict the time-average of the train loss, the squared gradient, and the covariance of the oscillations, we use eq. (163), eq. (164), and eq. (155), respectively.

A.6 Differential Complementarity Problems

In this appendix, we give some brief background on differential complementarity problems (Stewart, 2011), and we describe how to turn these into ordinary differential equations.

A *differential complementarity problem* (DCP) (Stewart, 2011) is a dynamical system that is defined in terms of a complementarity relation. In this paper, we will consider DCPs of the form:

$$\tfrac{d}{dt}w(t) = f(w(t)) + B(w(t))[\Sigma(t)] \quad \text{where} \quad 0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0. \tag{DCP}$$

Here $f : \mathbb{R}^d \to \mathbb{R}^d$, $B : \mathbb{R}^d \to \mathbb{R}^d \otimes \mathrm{Sym}(\mathbb{R}^d)$ and $A : \mathbb{R}^d \to \mathrm{Sym}(\mathbb{R}^d)$ are given, and $w(t) \in \mathbb{R}^d$ and $\Sigma(t) \in \mathrm{Sym}(\mathbb{R}^d)$ are dynamical variables that must respect eq. (DCP) for almost all times t.

Example 1. The following 1-dimensional DCP models a particle moving to the right which hits a wall at $w = 1$:

$$\frac{dw(t)}{dt} = 1 - \Sigma(t) \quad \text{where} \quad 0 \le \Sigma(t) \perp 1 - w(t) \ge 0.$$

When $w(t) < 1$, complementarity forces $\Sigma(t) = 0$ so that $\frac{dw(t)}{dt} = 1$, i.e. the particle moves to the right. Once the particle has made contact with the wall at $w = 1$, $\Sigma(t)$ must jump to 1 so that $\frac{dw(t)}{dt^+} = 0$ in order to prevent the particle from violating the condition $w(t) \le 1$. Thus, if $t^\star$ denotes the t when $w(t)$ hits the wall, then we have $\Sigma(t) = 0$ for $t < t^\star$ and $\Sigma(t) = 1$ for $t > t^\star$. The choice of $\Sigma(t)$ at $t^\star$ itself is arbitrary and does not affect the DCP.

Throughout our derivations, we will search for a $\Sigma(t)$ that is right-continuous, e.g. in the above setting we would define $\Sigma(t^\star) = 1$, so that the leftwards force is applied the instant that $w(t)$ reaches 1.

To turn eq. (DCP) into an ordinary differential equation with an explicit right-hand side, we will now prove some additional constraints that the system must satisfy.

First, we prove that if $A(w(t)) \succeq 0$ for all times t, then the right derivative $\frac{d}{dt^+}A(w(t))$ must be PSD over the subspace $\ker A(w(t))$. We abbreviate $A(w(t))$ as $A(t)$.

Lemma 4 (Semidefinite Tangent Cone)**.** Let $A : \mathbb{R} \to \mathrm{Sym}(\mathbb{R}^d)$ be a matrix-valued function such that $A(t) \succeq 0$ for all t. If A is right-differentiable at t, $\frac{d}{dt_+}A(t) \succeq_{\mathcal{U}} 0$ where $\mathcal{U} = \ker A(t)$. If A is differentiable at t, $\frac{d}{dt}A(t) =_{\mathcal{U}} 0$.

Proof. Let $u \in \ker A(t)$. Then for any $\epsilon > 0$,

$$u^\top \left[\frac{A(t+\epsilon) - A(t)}{\epsilon}\right] u = u^\top \left[\frac{A(t+\epsilon)}{\epsilon}\right] u \ge 0.$$

Taking $\epsilon \to 0$ proves that $\frac{d}{dt_+}A(t) \succeq_{\mathcal{U}} 0$. By reversing time if A is left differentiable at t then $\frac{d}{dt_-}A(t) \preceq_{\mathcal{U}} 0$. Combining these inequalities shows that if A is differentiable at t then $\frac{d}{dt}A(t) \big|_{\mathcal{U}} = 0$. □

Next, we show that if the complementarity relation $0 \preceq \Sigma(t) \perp A(w(t)) \succeq 0$ holds for all times t, then the right derivative $\frac{d}{dt^+}A(w(t))$ must satisfy its own complementarity relation: $0 \preceq \Sigma(t) \perp \frac{d}{dt}A(w(t)) \succeq_{\ker A(w(t))} 0$.

Lemma 5 (Differentiating the complementarity relation)**.** If $A : \mathbb{R} \to \mathrm{Sym}(\mathbb{R}^d)$ is right-differentiable at t, $0 \preceq \Sigma(t) \perp A(t) \succeq 0$ for all t, and $\Sigma(\cdot)$ is right-continuous at t, then $0 \preceq \Sigma(t) \perp \frac{d}{dt^+}A(t) \succeq_{\mathcal{U}} 0$ where $\mathcal{U} = \ker A(t)$.

Proof. The only condition that needs to be checked is $\Sigma(t) \perp \frac{d}{dt^+}A(t)$, since $\Sigma(t) \succeq 0$ is given and $\frac{d}{dt^+}A(t) \succeq_{\mathcal{U}} 0$ is implied by Lemma 4. By assumption, for any $\epsilon > 0$, we have $\Sigma(t+\epsilon) \perp A(t+\epsilon)$, which implies:

$$\left\langle \Sigma(t+\epsilon), \frac{A(t+\epsilon) - A(t)}{\epsilon} \right\rangle = -\left\langle \Sigma(t+\epsilon), \frac{A(t)}{\epsilon} \right\rangle \le 0.$$

Taking $\epsilon \to 0$ and using right-continuity of $\Sigma(\cdot)$ implies $\left\langle \Sigma(t), \frac{d}{dt^+}A(t) \right\rangle \le 0$. But also, $\left\langle \Sigma(t), \frac{d}{dt^+}A(t) \right\rangle \ge 0$ as both are PSD matrices by Lemma 4. This implies that $\left\langle \Sigma(t), \frac{d}{dt^+}A(t) \right\rangle = 0$. □

Note that as discussed in Appendix A.1.6, throughout the rest of this work, when we write $\frac{dw}{dt}$ we either mean $\frac{dw}{dt^+}$ or we mean that the statement holds for almost all t.

Now we have enough information to solve for $\Sigma(t)$. Explicitly, if $\mathcal{U}(w) := \ker A(w)$, we can define

$$\alpha(w) := \nabla A(w)[f(w)] \in \mathrm{Sym}(\mathbb{R}^d), \quad \beta(w) := \nabla A(w) B(w) \in \mathrm{Sym}(\mathbb{R}^d)^{\otimes 2}.$$

Lemma 6. If $w(t), \Sigma(t)$ satisfy eq. (DCP) for almost all t, and $\Sigma(\cdot)$ is right-continuous, then

$$\Sigma(t) \in \mathrm{SDCP}\left(\alpha(w(t)), \beta(w(t))\right)$$

for all t, where $\mathcal{U}(w) = \ker A(w)$. Furthermore if $\beta(w(t))$ is symmetric positive definite as an operator on $\mathrm{Sym}(\mathcal{U}(w))$ for all t, then $\Sigma(t) = \Sigma(w(t))$ is unique.

Proof. First, Lemma 5 implies that

$$0 \preceq \Sigma(t) \perp \tfrac{d}{dt}A(w(t)) \succeq_{\mathcal{U}(w(t))} 0$$

where $\mathcal{U}(w) = \ker A(w(t))$. By the chain rule,

$$\frac{d}{dt}A(w(t)) = \nabla A(w(t))\left[\frac{dw}{dt}\right] = \nabla A(w(t))[f(w(t)) + B(w(t))[\Sigma(t)]].$$

Therefore, using that $\Sigma \in \mathrm{Sym}(\mathcal{U}(w))$, we can expand $\frac{d}{dt}A(w(t))$ as:

$$\frac{d}{dt}A(w(t)) = \alpha(w(t)) + \beta(w(t))[\Sigma(t)],$$

which implies that:

$$\Sigma(t) \in \mathrm{SDCP}_{\mathcal{U}(w(t))}\left(\alpha(w(t)), \beta(w(t))\right).$$

□

This lets us turn eq. (DCP) into an ODE:

$$\frac{dw}{dt} = f(w) + B(w)[\Sigma] \quad \text{where} \quad \Sigma \in \mathrm{SDCP}_{\mathcal{U}(w)}\left(\alpha(w), \beta(w)\right)$$
$$\alpha(w) := \nabla A(w)[f(w)], \quad \beta(w) := \nabla A(w)B(w), \quad \mathcal{U}(w) := \ker A(w)$$

(DCP to ODE)

We can also define an equivalent basis-dependent version of this ODE, which makes clear that simulating the ODE only requires time and space that scale linearly in the dimension d. For each t, let $U \in \mathbb{R}^{d\times k}$ denote a basis for the critical subspace $\mathcal{U}(w(t))$, where $k = \dim \mathcal{U}(w(t))$. Then we can write eq. (DCP) in terms of U as follows:

$$\begin{aligned}
\frac{dw}{dt} &= f(w) + B(w)[UXU^\top] \\
X &\in \mathrm{SDCP}_{\mathbb{R}^k}(\alpha_U(w), \beta_U(w)) \\
\alpha_U(w) &:= U^\top \alpha(w) U \in \mathrm{Sym}(\mathbb{R}^k) \\
\beta_U(w)[X] &:= U^\top \Big(\beta(w)[UXU^\top]\Big) U \in \mathrm{Sym}(\mathbb{R}^k) \quad \forall X \in \mathrm{Sym}(\mathbb{R}^k).
\end{aligned}$$

(DCP to ODE, basis dependent)

Thus, to compute $\frac{dw}{dt}$, one need only compute $\alpha_U \in \mathrm{Sym}(\mathbb{R}^k)$ and $\beta_U \in \mathrm{Sym}(\mathbb{R}^k)^{\otimes 2}$, then solve a k-dimensional SDCP to obtain $X \in \mathrm{Sym}(\mathbb{R}^k)$, then form $\frac{dw}{dt}$ in terms of X.

In practice, due to nonsmoothness of the DCP, we do not discretize it by computing $\frac{dw}{dt}$ and then taking Euler steps. Instead, we discretize the DCP directly, as described below in Appendix A.6.1.

Relation to the broader DCP literature In the literature on differential complementarity problems, it is a standard practice to turn a DCP into an ODE by differentiating the constraints. In particular, eq. (DCP) is known as a *pure index-one DCP* (Stewart, 2011, Section 5.2.1), because it needs to be differentiated exactly once in order to be turned into an explicit ODE.

Existence and uniqueness of the ODE Existence and uniqueness for projected gradient flows (including the gradient descent central flow) is guaranteed by Cornet (1983), under the assumption that $\beta(w)$ is positive definite over $\mathrm{Sym}(\mathcal{U}(w))$ for all w, where $\mathcal{U}(w) = \ker A(w)$ denotes the critical subspace. We view this assumption as mild. Existence and uniqueness can be argued for DCPs more generally (including the other central flows) under the assumption that β is positive definite over all $\mathrm{Sym}(\mathbb{R}^d)$ (Stewart, 2011). However, this assumption is too strong for our setting; indeed, for gradient descent, β has rank at most d and hence cannot span the $\frac{d(d+1)}{2}$ dimensional space of $\mathrm{Sym}(\mathbb{R}^d)$ whenever $d > 1$. However, we suspect that under additional reasonable regularity conditions, this could be relaxed to a condition that β need only be positive definite over $\mathrm{Sym}(\ker A(w))$.

Smoothness of the ODE The ODE is non-smooth at *breakpoints* where the dimension of $\mathcal{U}(w) = \ker A(w)$ changes. In between the breakpoints, $\Sigma(t)$ is continuous and $w(t)$ is differentiable. Moreover, the solution to the SDCP is given by the linear inverse $\Sigma(t) = U\beta_U^{-1}(w)[\alpha_U(w)]U^\top$ and we have $\frac{d}{dt}A(t)\,\big|_{\mathcal{U}(w)} = 0$. At the breakpoints, however, $\Sigma(t)$ is discontinuous and $w(t)$ is not differentiable (although they are right-continuous and right-differentiable, respectively)

A.6.1 Discretizing the DCP

We now describe how we discretize eq. (DCP) in practice. Let $\epsilon > 0$ denote the discretization step size, and consider a grid of time steps $\mathcal{T} = \{t_0, t_1, \ldots, t_N\}$ that are spaced apart by ϵ. For any $t \in \mathcal{T}$, let $w^{(t)}$ denote our estimate for the central flow's $w(t)$, and let $\Sigma^{(t)}$ denote our estimate for the central flow's $\Sigma(t)$. In what follows, we will use $t \in \mathcal{T}$ to denote the current time step and $t + \epsilon \in \mathcal{T}$ to denote the next one.

It is not immediately trivial to discretize eq. (DCP) using time-stepping. For example, it doesn't make sense to search for a pair $w^{(t+\epsilon)}$, $\Sigma^{(t)}$ that satisfies

$$w^{(t+\epsilon)} \;=\; w^{(t)} + \epsilon\left(f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[\Sigma^{(t)}\Big]\right) \quad \text{and} \quad 0 \preceq \Sigma^{(t)} \perp A\Big(w^{(t)}\Big) \succeq 0,$$

as this would always be satisfied by $\Sigma^{(t)} = 0$. We could instead search for a pair $w^{(t+\epsilon)}$, $\Sigma^{(t)}$ which satisfies

$$w^{(t+\epsilon)} \;=\; w^{(t)} + \epsilon\left(f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[\Sigma^{(t)}\Big]\right) \quad \text{and} \quad 0 \preceq \Sigma^{(t)} \perp A\Big(w^{(t+\epsilon)}\Big) \succeq 0,$$

where the complementarity constraint is enforced at time $t + \epsilon$, rather than at time t. However, it is still difficult to handle this constraint due to the nonlinearity of A. We will therefore *linearize* A around $w^{(t)}$. Define $A^{\text{lin}}(w)$ as the linearization of A around the current point $w^{(t)}$:

$$A^{\text{lin}}(w) \;:=\; A\Big(w^{(t)}\Big) + \nabla A\Big(w^{(t)}\Big)\Big[\,w - w^{(t)}\,\Big]. \tag{168}$$

We can therefore look for a choice of $w^{(t+\epsilon)}$ and $\Sigma^{(t)}$ that together satisfy

$$w^{(t+\epsilon)} \;=\; w^{(t)} + \epsilon\left(f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[\Sigma^{(t)}\Big]\right) \quad \text{and} \quad 0 \preceq \Sigma^{(t)} \perp A^{\text{lin}}\Big(w^{(t+\epsilon)}\Big) \succeq 0. \tag{169}$$

This implies the following set of conditions on $\Sigma^{(t)}$ alone:

$$0 \preceq \Sigma^{(t)} \perp A\Big(w^{(t)}\Big) + \epsilon\,\nabla A\Big(w^{(t)}\Big)\Big[f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[\Sigma^{(t)}\Big]\Big], \tag{170}$$

which we recognize as precisely an SDCP:

$$\Sigma^{(t)} \;\in\; \mathrm{SDCP}_{\mathbb{R}^d}\left(A\Big(w^{(t)}\Big) + \epsilon\,\nabla A\Big(w^{(t)}\Big)\Big[f\Big(w^{(t)}\Big)\Big],\; \epsilon\,\nabla A\Big(w^{(t)}\Big)\,B\Big(w^{(t)}\Big)\right). \tag{171}$$

Thus one could solve for $\Sigma^{(t)}$ above, and then take an Euler step on w:

$$w^{(t+\epsilon)} \;\leftarrow\; w^{(t)} + \epsilon\left(f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[\Sigma^{(t)}\Big]\right).$$

Unfortunately, it is not possible to directly run this "idealized" time-stepping scheme, as it is impractical to formulate or solve an SDCP over $\mathbb{R}^d$. Therefore, we instead approximate it by projecting it onto the bottom eigendirections of A which are "close" to the stability threshold at 0. In particular, if $U \in \mathbb{R}^{d \times k}$ is a basis of these directions and $\mathcal{U} = \operatorname{span} U$, then we require $\Sigma^{(t)} \in \operatorname{Sym}(\mathcal{U})$, and we enforce:

$$0 \preceq \Sigma^{(t)} \perp A^{\text{lin}}\Big(w^{(t+\epsilon)}\Big) \succeq_{\mathcal{U}} 0. \tag{172}$$

This boils down to an SDCP over the low-dimensional subspace $\mathcal{U}$:

$$\Sigma^{(t)} \in \text{SDCP}_{\mathcal{U}}\left(A\Big(w^{(t)}\Big) + \epsilon\, \nabla A\Big(w^{(t)}\Big)\Big[f\Big(w^{(t)}\Big)\Big],\ \epsilon\, \nabla A\Big(w^{(t)}\Big)\, B\Big(w^{(t)}\Big)\right). \tag{173}$$

As described in Appendix A.1.4, solving this SDCP only requires solving a k-dimensional SDCP:

$$\Sigma^{(t)} \ = \ U\, X^{(t)} U^\top, \quad X^{(t)} \ = \ \text{SDCP}\left(A_U\Big(w^{(t)}\Big) + \epsilon\, \alpha_U\Big(w^{(t)}\Big),\ \epsilon\, \beta_U\Big(w^{(t)}\Big)\right), \tag{174}$$

where $A_U(w) \in \operatorname{Sym}(\mathbb{R}^k)$, $\alpha_U(w) \in \operatorname{Sym}(\mathbb{R}^k)$, and $\beta_U(w) \in \operatorname{Sym}(\mathbb{R}^k)^{\otimes 2}$ are defined as

$$A_U(w) := U^\top A(w)\, U, \tag{175}$$

$$\alpha_U(w) := U^\top [\nabla A(w)[f(w)]] U, \tag{176}$$

$$\beta_U(w)[X] := U^\top \Big[\nabla A(w)\Big[B(w)\Big[U X U^\top\Big]\Big]\Big] U. \tag{177}$$

Then, we update the weights using:

$$w^{(t+\epsilon)} \ = \ w^{(t)} + \epsilon\, \left(f\Big(w^{(t)}\Big) + B\Big(w^{(t)}\Big)\Big[U X^{(t)} U^\top\Big]\right).$$

For the central flows in this work, A and B are such that $\alpha_U\ \beta_U$, and $\frac{dw}{dt}$ can be computed efficiently.

Correctness of this discretization scheme Under suitable conditions on f, A, B, this time-stepping scheme will converge to the solution of the DCP as $\epsilon \to 0$ (Stewart, 2011). For technical reasons involving the rank of ∇AB, our paper will not rigorously prove the convergence of this time-stepping scheme for our central flows. However, we do empirically observe that the dynamics converge to a limiting curve as $\epsilon \to 0$.

A.7 Continuous-time approximation to an EMA

In this appendix, we justify our choice for the continuous-time approximation to an EMA.

Consider a discrete-time exponential moving average (EMA) of a process $f(t)$:

$$\nu_t = \beta_2 \nu_{t-1} + (1 - \beta_2) f(t). \tag{178}$$

What is a good continuous-time approximation $\nu(t)$ to ν_t? This question arises when we derive stable and central flows for Scalar RMSProp and RMSProp (where f is the squared gradient norm or the elementwise squared gradient, respectively).

Subtracting ν_{t-1} from both sides of eq. (178) and rearranging yields:

$$\nu_t - \nu_{t-1} = (1 - \beta_2)(f_t - \nu_{t-1}). \tag{179}$$

This suggests the following continuous-time approximation to eq. (178):

$$\nu'(t) = (1 - \beta_2)(f(t) - \nu_t). \tag{180}$$

However, this approximation breaks down for small β_2. Indeed, as $\beta_2 \to 0$, the discrete-time EMA eq. (178) adapts "infinitely fast" so that $\nu_t \approx f(t)$, yet the naive continuous-time approximation eq. (180) does not have this property.

Therefore, to obtain a continuous-time approximation that works well even for small β_2, we use the following alternative approximation:

$$\nu'(t) = \left(\frac{1-\beta_2}{\beta_2}\right)(f(t) - \nu(t)). \tag{181}$$

The β_2 in the denominator ensures that when $\beta_2 \approx 0$, $\nu(t)$ adapts "infinitely" fast to $f(t)$.

To give intuition for our approximation eq. (181), suppose that we are using an EMA to track a one-dimensional *linear* process $f(t)$ (i.e. $f'(t)$ is constant). Then both the discrete and continuous time EMA's have closed forms. The closed-form solution to the discrete time EMA eq. (178) can be written as:

$$\nu_t = f(\underbrace{t-\tau}_{\text{time-lag}}) + \underbrace{\beta_2^t[\nu_0 - f(-\tau)]}_{\text{burn in}} \quad \text{where} \quad \tau := \frac{1-\beta_2}{\beta_2}. \tag{182}$$

Since the burn-in term vanishes exponentially with time, the steady state is that ν_t will track $f(t)$ with a "time delay" of $\tau := \frac{\beta_2}{1-\beta_2}$.

Similarly, for a continuous-time EMA of the form $\nu'(t) = \gamma[f(t) - \nu(t)]$ for some γ, the general solution is:

$$\nu(t) = f(\underbrace{t-\tau}_{\text{time-lag}}) + \underbrace{e^{-\gamma t}[\nu(0) - f(-\tau)]}_{\text{burn in}} \quad \text{where} \quad \tau := 1/\gamma. \tag{183}$$

Thus, the steady state is that $v(t)$ will track $f(t)$ with a "time delay" of $\tau = 1/\gamma$.

To ensure that the continuous-time EMA asymptotically matches the discrete-time EMA, we need to set γ so that the time delays match:

$$\frac{\beta_2}{1-\beta_2} = \frac{1}{\gamma} \implies \gamma = \frac{1-\beta_2}{\beta_2},$$

This motivate our choice of scaling factor in eq. (181).

Note that if we instead wanted to match the *burn in*, we would set $\gamma = \log(1/\beta_2)$. However, we believe it is more important to match the time-lag than the burn in.

A.8 Miscellaneous math

Here, we state some miscellaneous mathematical facts that are used elsewhere.

Fact 1. Let $L(w)$ be three-times differentiable, and let $S(w) := \lambda_1(H(w)))$ denote the top Hessian eigenvalue. Suppose that the top Hessian eigenvalue at $\overline{w}$ has multiplicity 1, and let u be the top Hessian eigenvector at $\overline{w}$. Then, for $w = \overline{w} + xu$, we have:

$$\nabla L(w) = \nabla L(\overline{w}) + S(\overline{w})xu + \tfrac{x^2}{2}\nabla S(\overline{w}) + o(x^2). \tag{184}$$

Proof. We begin by writing the Taylor expansion of $\nabla L(w)$ around $\overline{w}$:

$$\nabla L(w) = \nabla L(\overline{w}) + H(\overline{w})xu + \tfrac{x^2}{2}\nabla^3 L(\overline{w})[u,u] + o(x^2). \tag{185}$$

Because u is an eigenvector of $H(\overline{w})$ with eigenvalue $S(\overline{w})$, the second term can be simplified to $S(\overline{w})xu$. Finally, by Danskin's theorem (or equivalently the standard formula for the derivative of an eigenvalue):

$$\nabla_{\overline{w}} S(\overline{w}) = \nabla_{\overline{w}}\left[\max_{\|v\|=1} v^T H(\overline{w})v\right] = \nabla_{\overline{w}}\left[u^T H(\overline{w})u\right] = \nabla^3 L(\overline{w})[u,u] \tag{186}$$

where u is the argmax of the second expression, i.e. the top eigenvector of the Hessian at $\overline{w}$. □

Fact 2. For PSD matrices $X, Y \succeq 0$, it holds that $\operatorname{tr}(XY) = 0$ if and only if $\operatorname{span} X \perp \operatorname{span} Y$.

Proof. First, if $\operatorname{span} X \perp \operatorname{span} Y$, then $\operatorname{span} Y \subseteq \ker X$. Thus, it must hold for every u that $XYu = 0$, implying that $XY = 0$ and hence $\operatorname{tr}(XY) = 0$.

For the other direction, assume that $\operatorname{tr}(XY) = 0$. Then:

$$0 = \operatorname{tr}(XY) = \operatorname{tr}\left(Y^{1/2}X^{1/2}X^{1/2}Y^{1/2}\right) = \left\|X^{1/2}Y^{1/2}\right\|_F^2.$$

Since X, Y are PSD, so is $X^{1/2}Y^{1/2}$, and hence its norm can only be zero if $X^{1/2}Y^{1/2} = 0$. Multiplying on the right by $Y^{1/2}$ and the left by $X^{1/2}$ gives $XY = 0$. This implies that $\operatorname{span} X \perp \operatorname{span} Y$, as for any $x \in \operatorname{span} X$ and $y \in \operatorname{span} Y$, we have $x = Xu$ and $y = Yv$ for some u, v, and so

$$x^\top y = (Xu)^\top (Yv) = u^\top XYv = 0.$$

□

Note that since the span of a symmetric matrix is orthogonal to its kernel, $\operatorname{span} X \perp \operatorname{span} Y$ is equivalent to $\operatorname{span} X \subseteq \ker Y$ and to $\operatorname{span} Y \subseteq \ker X$. Thus the following corollary is immediate:

Corollary 1. For PSD matrices $X, Y \succeq 0$, it holds that $\operatorname{tr}(XY) = 0$ if and only if $\operatorname{span} X \subseteq \ker Y$.

B Experimental Details

B.1 Implementation details

Our code can be found at: `http://github.com/centralflows/centralflows`.

In order to reuse code between the central flows for gradient descent, Scalar RMSProp, and RMSProp, we cast all three optimizers as instances of the generic adaptive preconditioned method that is described in Appendix A.5. This template assumes that the weights are updated via a preconditioned gradient step of the form $w_{t+1} = w_t - P(\nu_t)^{-1}\nabla L(w_t)$, where $P(\nu_t)$ is a preconditioner that is derived from some optimizer state ν_t that is in turn updated based on the gradients. For example, for gradient descent with learning rate η, the preconditioner is simply $P = \eta^{-1}I$. The effective Hessian is defined as $P(\nu_t)^{-1}H(w_t)$, and the EOS condition is that the largest eigenvalue of this matrix (the effective sharpness S^{eff}) is 2. See Appendix A.5 for more information.

Eigenvalue computation To regularly recompute the top eigenvalues and eigenvectors of the effective Hessian, we use the LOBPCG algorithm (Knyazev, 2001), which only requires access to the Hessian via Hessian-vector products, and which allows us to warm-start using the previously computed eigenvectors. We were originally inspired by the LOBPCG implementation in Jax's `jax.experimental.sparse.linalg` (Bradbury et al., 2018).

How many eigenvalues to track? For all processes (i.e. discrete optimizers, central flows, stable flows), we track all eigenvalues of the effective Hessian that are above the threshold 1.5. We then track the same number of eigenvalues of the Hessian. Note that for gradient descent and Scalar RMSProp the Hessian eigenvalues are trivially related to the effective Hessian eigenvalues, whereas for RMSProp we need to do an extra eigenvalue solve to obtain the Hessian eigenvalues.

Discretizing the stable flow To discretize the stable flows (e.g. gradient flow), we use Euler's method. To discretize for one unit of time, we pick some integer `nsubsteps`, we set $\epsilon = 1/\texttt{nsubsteps}$, and we repeat $w \leftarrow w + \epsilon \frac{dw}{dt}$ for `nsubsteps` times. We dynamically adapt `nsubsteps` based on the current effective sharpness S^{eff}. The basic criterion of update stability requires that $\epsilon < 2/S^{\text{eff}}$. To be on the safe side, and to guard against any implicit discretization effects, we enforce the stronger condition that $\epsilon < 0.5/S^{\text{eff}}$, or equivalently that $\texttt{nsubsteps} \geq \lceil 2S^{\text{eff}} \rceil$. We also enforce a floor of $\texttt{nsubsteps} \geq 4$. Thus, we set $\texttt{nsubsteps} = \max(4, \lceil 2S^{\text{eff}} \rceil)$.

Since discretizing the stable flow would take a prohibitively long time in regions of weight space where the effective sharpness is too high, we automatically terminate the stable flow if the effective sharpness exceeds a certain threshold (we used `100`).

Discretizing the central flow To discretize the central flows, we use the scheme described in Appendix A.5.1. This is in turn an instance of the general scheme described in Appendix A.6.1 for discretizing differential complementarity problems. At each discretization time step, we do the following:

1. We compute the top eigenvalues and eigenvectors of the effective Hessian $P(\nu)^{-1}H(w)$. In particular, we compute all eigenvalues greater than $2 - \tau$ for some small tolerance $\tau > 0$ (we use `0.05`), as well as the corresponding eigenvectors. Let k be the number of such eigenvalues. To compute eigenvalues of the non-symmetric matrix $P(\nu)^{-1}H(w)$, we first use warm-started LOBPCG to compute the eigenvalues $D \in \text{diag}(\mathbb{R}^k)$ and orthonormal eigenvectors $\tilde{U} \in \mathbb{R}^{d\times k}$ of the symmetric matrix $P(\nu)^{-1/2}H(w)P(\nu)^{-1/2}$. The eigenvalues of $P(\nu)^{-1}H(w)$ are then D, and the eigenvectors are $U = P(\nu)^{-1/2}\tilde{U} \in \mathbb{R}^{d\times k}$. Note that these eigenvectors are orthonormal w.r.t the preconditioner $P(\nu)$, i.e. $U^\top P(\nu)U = I$.

2. We compute the third-derivative tensor $\nabla_U H(w) \in \text{Sym}(\mathbb{R}^k) \otimes \mathbb{R}^d$ defined in eq. (156) by looping over all pairs (u_i, u_j) of columns of $U \in \mathbb{R}^{d\times k}$ and computing the third derivative $\nabla_w[u_i^\top H(w)u_j] \in \mathbb{R}^d$, which can be done using automatic differentiation. Note that due to the symmetry, it is only necessary to compute and store the $\binom{k+1}{2}$ "upper triangular" entries of this tensor, rather than the full k^2.

3. We compute the tensor $\nabla P_U(\nu) \in \text{Sym}(\mathbb{R}^k) \otimes \mathbb{R}^{d_\nu}$ defined in eq. (156), which measures the gradient of the preconditioner w.r.t the optimizer state $\nu \in \mathbb{R}^{d_\nu}$. Our implementation uses automatic differentiation to do this,

though one could also simply hard-code the derivatives for the various optimizers of interest. We similarly compute the tensor $\nabla^2 G_{PU}(\nu, g(w)) \in \mathbb{R}^{d_\nu} \otimes \mathrm{Sym}(\mathbb{R}^k)$ defined in eq. (157), where $g(w) = \nabla L(w)$ and d_ν is the dimension of the optimizer state ν. This tensor measures the Hessian of the optimizer state w.r.t the gradient of the weights. For these tensors, we also only need to compute and store the $\binom{k+1}{2}$ upper triangular entries.

4. Using $\nabla H_U(w)$, $\nabla P_U(\nu)$, and $\nabla^2 G_{PU}(\nu, g(w))$, we compute the tensors $\alpha_U(w, \nu) \in \mathrm{Sym}(\mathbb{R}^k)$ and $\beta_U(w, \nu) \in \mathrm{Sym}(\mathbb{R}^k)^{\otimes 2}$ defined in eqs. (161) and (162). Here we simply materialize the full tensors, as k is small.

5. We solve the semidefinite complementarity problem:

$$X = \mathrm{SDCP}_{\mathbb{R}^k}(2I - \Lambda + \epsilon\, \alpha_U, \epsilon\, \beta_U).$$

   We do so by formulating the SDCP as a semidefinite-constrained quadratic program eq. (54), as described in Appendix A.1.4, and solving this using the convex programming library `cvxpy`.

6. We take an Euler step of size ϵ on the weights w and optimizer state ν, as given in eqs. (166) and (167).

To discretize the flow for one unit of time, we pick some integer `nsubsteps`, set the discretization step size as $\epsilon = 1/\texttt{nsubsteps}$, and repeat the above process for `nsubsteps` times. Note that because the central flows keep the effective sharpness controlled at 2, it is not necessary to dynamically adapt `nsubsteps` throughout training, as $\texttt{nsubsteps} \geq 2$ always sufffices to ensure stability. We therefore used the fixed values $\texttt{nsubsteps} = 4$ and $\epsilon = 0.25$.

The computational cost of each discretization step is dominated by the cost of computing the top eigenvalues and eigenvectors in step 1 above, as well as the cost of computing the third derivatives in step 2 above. The computational cost of the rest of the steps (including solving the SDCP) is negligible. The time complexity of each discretization step scales quadratically with the number of eigenvalues that are at the edge of stability, k, as $\binom{k+1}{2} = \Theta(k^2)$ third derivatives need to be computed.

Verifying the central flow To assess whether the central flow accurately models the discrete optimizer, we run both processes simultaneously. That is, we repeatedly both (a) take a step on the discrete optimizer; and (b) discretize the central flow for one unit of time. Let w_t denote the discrete optimizer's iterate at step t, and let $w^{(t)}$ denote our estimate for the central flow solution at time t.

To verify that the central flow approximates the long-term weight-space trajectory of the discrete optimizer, we record $\|w^{(t)} - w_t\|$, the Euclidean distance between the discrete optimizer and the central flow.

We use eqs. (163) and (164) to compute the central flow's predictions for the time-average of the train loss and squared gradient norm. For any quantity f (e.g. loss or squared gradient norm), let $\bar{f}(w^{(t)})$ denote the central flow's prediction for the time-average of f at step/time t. To assess the accuracy of this prediction, we compare $\{\bar{f}(w^{(t)})\}$ against a Gaussian smoothing of the empirical time series $\{f(w_t)\}$. That is, for each t, we compare:

$$\bar{f}(w^{(t)}) \quad \text{vs.} \quad \left(\frac{1}{\sum_j c_j}\right) \sum_j c_j f(w_{t+j}) \quad \text{where} \quad c_j = \exp\left(-\frac{j^2}{2\sigma^2}\right),$$

where σ^2 is the bandwidth of the Gaussian kernel. We describe below momentarily how we determine σ^2.

When predicting the covariance of the oscillations, we compare both sides of eq. (155). Let $(\lambda_i^{(t)}, v_i^{(t)})$ denote the i-th eigenvalue and eigenvector of the matrix $P^{1/2}(\nu^{(t)})\, \Sigma^{(t)}\, P^{1/2}(\nu^{(t)})$, and define $x_i^{(t)} := \left\langle v_i^{(t)}, P^{1/2}(\nu^{(t)})(w(t) - w_t) \right\rangle^2$. Then for each eigenvalue index i, we compare the predicted time series $\{\lambda_i^{(t)}\}$ against a Gaussian smoothing of the empirical time series $\{x_i^{(t)}\}$. For gradient descent and Scalar RMSProp, where P is a scalar, we post-hoc rescaled both quantities by P so as to report the oscillations in terms of Σ rather than $P^{1/2}\Sigma P^{1/2}$.

For some plots (e.g. those in the main paper), we picked the Gaussian kernel's bandwidth by visual inspection (we emphasize that it is not possible to turn a bad prediction into a good prediction by adjusting the bandwidth). In fact,

for some figures (e.g. Figures 15 and 18), we found it best to re-tune the bandwidth *within* an experiment, whenever the number of unstable eigenvalues underwent a change. On the other hand, for the plots on the "bulk" experiments section (Appendix E), we picked a single bandwidth somewhat arbitrarily and used this for all experiments.

Warm-start In our experiments, we first run the discrete optimizer for 10-15 steps and then use this as an initialization for the discrete optimizer, central flow, and stable flow. The first reason why we do this is that the effective sharpness is sometimes much larger than 2 at the original initialization (particularly for the adaptive optimizers), yet comes down below 2 within the very first few steps of training. (The central flow is not currently defined when the effective sharpness is greater than 2.) Another reason is that even when the effective sharpness is less than 2 at initialization, we observed that the quality of the central flow approximation is sometimes enhanced by this warm start. This is potentially due to the size of the gradients during the first few steps, and the central flow could possibly be improved during this phase by incorporating the implicit gradient norm penalty from Barrett and Dherin (2021). However, we think it is likely that this source of deviation between the discrete optimizer and the stable / central flows is negligible in the long run (see Appendix C.1), and thus we hypothesize that the warm-starting could be removed in cases where the effective sharpness is less than 2 at initialization.

Second-order midpoints When reporting metrics from the discrete optimizers (gradient descent, Scalar RMSProp, and RMSProp, as opposed to their central flows), we usually report the top Hessian eigenvalues measured not at the optimizer iterates $\{w_t\}$ themselves, but at the (second order) midpoints $\{\hat{w}_t\}$, defined as $\hat{w}_t := \frac{1}{4}\left[2w_t - w_{t-1} - w_{t+1}\right]$ (so named because it is the midpoint of the midpoints $\frac{1}{2}[w_t - w_{t-1}]$ and $\frac{1}{2}[w_{t+1} - w_t]$). This empirically makes the Hessian trajectories slightly crisper (less "noisy"), while not altering the fundamental patterns.

Numerical issues in the central flow In some runs (e.g. Figure 55.17), the central flow's $\Sigma(t)$ predictions are erratic for stretches. This phenomenon seems to be a numerical issue that arises when discretizing the flow. We are optimistic that the issue is fixable.

B.2 Architecture details

Here we describe our architectures. Note that our code for all architectures can be found at: `http://github.com/centralflows/centralflows`.

Both our derivations and the analytic formulas for the central flows rely on higher-order information about the loss function (e.g. Hessians and third derivatives). As a result, we require that all of the architectures are smooth. This rules out the commonly used ReLU activation (Nair and Hinton, 2010).

CNN Our CNN has four layers, an initial channel width of 32, and 3x3 convolutional kernels. It uses the GeLU activation function, average pooling, and a linear readout layer.

ResNet We use a ResNet (He et al., 2016) with 20 layers and GeLU activations. We use GroupNorm (Wu and He, 2018) in place of BatchNorm (Ioffe and Szegedy, 2015), as we empirically find that BatchNorm often leads to sub/super-quadraticity (see the discussion in Appendix C.2).

Vision Transformer We use the Vision Transformer (ViT) (Dosovitskiy et al., 2021) implementation from LucidRains (2024). Our ViT has depth 3, embedding dimension 64, number of heads 8, MLP dimension 256, and patch size 4. We use the ViT modifications proposed in Beyer et al. (2022). We initialize the weights and biases of the final linear layer to be zero, as this makes the curvature low at initialization. For unknown reasons, we found that the core PyTorch LayerNorm implementation (written in C++) leads to third derivatives being computed incorrectly; thus, we substituted in an alternative implementation written in vanilla PyTorch, which empirically fixed the issue.

LSTM Our LSTM (Hochreiter and Schmidhuber, 1997) has 2 layers, an embedding dimension of 48, and a hidden dimension of 48.

(Sequence) Transformer Our sequence transformer has 4 layers, an embedding dimension of 32, an MLP dimension of 128, and 4 attention heads. We disabled dropout, to make the network deterministic. As with the ViT

(see above), we initialize the weights and biases of the final linear layer to be zero, and we substitute a vanilla PyTorch LayerNorm implementation in place of the default C++ LayerNorm implementation.

Mamba We use the Mamba (Gu and Dao, 2024) implementation from Torres-Leguet (2024). Our Mamba has 2 layers and a model dimension of 64.

B.3 Dataset details

Here we describe our datasets. The code can be found at:
`http://github.com/centralflows/centralflows`.

CIFAR-10 We test the vision architectures on a subset of CIFAR-10 that contains 1000 training examples, all from the first 4 CIFAR-10 classes. We use the standard preprocessing of subtracting the dataset-wide channel-wise mean, and dividing by the dataset-wide channel-wise standard deviation. When training using MSE loss, we encode the ground truth class as 1 and the others as 0.

Sorting We test the sequence architectures on the synthetic sorting task described in Karpathy (2020). The network is fed a sequence of numbers and is then tasked (via a language modeling loss) with returning these numbers in sorted order. We used numbers 1 through 4, and sequences of length 8. The size of the training datraset was usually 1,000 (except for Mamba, where it was 250).

C Miscellaneous

C.1 Implicit gradient regularization

Recall from Section 3.2 that when gradient descent is stable, we approximate its trajectory by the gradient flow:

$$\frac{dw}{dt} = -\eta \nabla L(w). \tag{187}$$

In particular, the central flow automatically reduces to eq. (187) whenever sharpness $S(w) < 2/\eta$.

On the other hand, Barrett and Dherin (2021) argued that gradient descent with step size η should instead be approximated by a *modified* gradient flow with a penalty on the squared gradient norm:

$$\frac{dw}{dt} = -\eta \nabla \left[L(w) + \frac{\eta}{4} \|\nabla L(w)\|^2 \right], \tag{188}$$

$$= -\eta \left[\nabla L(w) + \frac{\eta}{2} H(w) \nabla L(w) \right]. \tag{189}$$

Subsequently, Rosca et al. (2023) showed how to improve the approximation by incorporating higher-order penalties, and Cattaneo et al. (2024) extended the approach to adaptive optimizers.

Empirically, we find that in the stable regime, the modified gradient flow eq. (188) is indeed a better approximation to gradient descent than the vanilla gradient flow eq. (187). This observation is illustrated in Figure 25. However, all things considered, we observe that in the stable regime, the vanilla gradient flow is *already* a good enough approximation to gradient descent (Figure 25). While gradient descent does differ from gradient flow in deep learning, the vast majority of this difference appears to be due to the curvature-reduction effect of oscillations in the edge of stability regime, not to discretization error that manifests even in the stable regime. This point is illustrated in Figure 26. Thus, in the interest of simplicity, we left out any implicit gradient regularizer from our central flows.

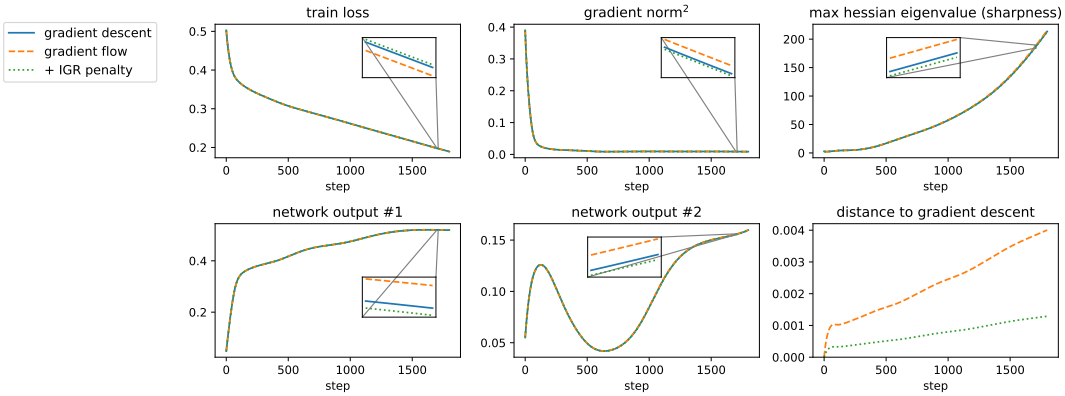


Figure 25: **In the stable regime, the IGR penalty marginally improves the accuracy of the gradient flow approximation, but this accuracy is already good.** We train a network using gradient descent (blue), vanilla gradient flow eq. (187) (orange, dashed) and gradient flow with the IGR penalty eq. (189) (green, dotted), all with $\eta = 0.01$. This figure shows the initial phase of training, when gradient descent is in the stable regime (i.e. sharpness is below $2/\eta$). Consistent with Barrett and Dherin (2021), notice that in the stable regime, gradient flow + IGR is a visibly better approximation to gradient descent than vanilla gradient flow. In particular, observe that the distance to the gradient descent trajectory is smaller for gradient flow + IGR than for vanilla gradient flow (bottom right). Similarly, note that the network outputs on two examples, the train loss, and the squared gradient norm agree better under gradient flow + IGR than under vanilla gradient flow. That said, notice that even the vanilla gradient flow is a good approximation to gradient descent in this regime. Please refer to Figure 26 for the continuation of this experiment into the EOS regime. *Details*: a CNN is trained on a subset of CIFAR-10 using MSE loss.

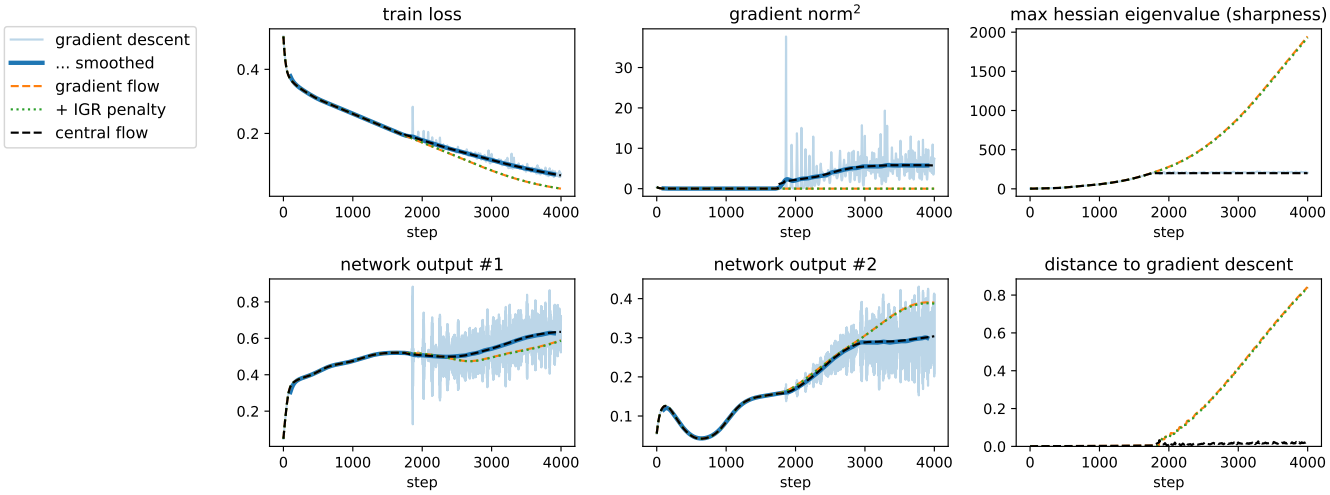


Figure 26: **In the EOS regime, the central flow accurately approximates the trajectory of gradient descent, whereas neither the original nor the IGR-penalized gradient flow does so.** We continue the experiment from Figure 25 for more iterations, into the EOS regime. Observe that neither the original gradient flow (orange, dashed) nor the IGR-penalized gradient flow (green, dotted) reasonably approximates the trajectory of gradient descent (blue) in this regime, whereas the central flow (black, dashed) does so. In particular, notice that the distance from gradient descent to the central flow stays small, whereas the distance to both the original and IGR-penalized gradient flows grows large over time (bottom right). Further, the central flow accurately predicts the time-averaged network outputs on two examples, as well as the train loss and squared gradient norm.

A subtle confounder Some works (e.g. Geiping et al. (2022)) have observed that adding an *explicit* squared gradient norm penalty can help full-batch training recover the superior generalization performance of minibatch training. This would seem to support the argument of Barrett and Dherin (2021) that the implicit regularization of discrete gradient descent can be captured by a flow with a squared gradient norm penalty. Yet, we believe that these results could be instead due to a subtle confounder: adding a squared gradient norm penalty changes the oscillatory EOS dynamics, and in particular, enhances the implicit curvature regularization.

Consider running gradient descent with step size η, while adding an implicit gradient regularizer corresponding to some step size τ. The update rule is:

$$w_{t+1} = w_t - \eta \left[\nabla L(w_t) + \tfrac{\tau}{2} H(w_t)\, \nabla L(w_t)\right]. \tag{190}$$

On the one-dimensional quadratic function $L(w) = \frac{1}{2} S w^2$, the iterates would evolve according to:

$$\begin{aligned} w_{t+1} &= w_t - \eta \left[S w_t + \tfrac{\tau}{2} S^2 w_t\right] \\ &= [1 - \eta S - \tfrac{1}{2}\eta\tau S^2] w_t. \end{aligned} \tag{191}$$

Whereas vanilla gradient descent is unstable if $S > 2/\eta$, this iteration is unstable if $\eta S + \frac{1}{2}\eta\tau S^2 > 2 \iff S > \frac{\sqrt{1+\frac{4\tau}{\eta}}-1}{\tau}$, which is $< \frac{2}{\eta}$. That is, it becomes unstable at *lower* values of the sharpness S. Accordingly, in line with the general EOS pattern, we find that on neural network objectives, while gradient descent implicitly constrains the sharpness to $2/\eta$, the update rule eq. (190) implicitly constrains the sharpness to this strictly smaller value (Figure 27). In other words, adding an explicit gradient norm penalty also results in stronger *implicit* curvature regularization. This acts as a subtle experimental confounder, which could explain the reports in the literature that explicitly penalizing the gradient norm substantially boosts generalization performance.

Thus, we hypothesize that if the the IGR-penalized gradient flow eq. (188) were properly discretized, it would not yield improved generalization (as it would not substantially affect the trajectory). Yet, if an IGR-penalized objective

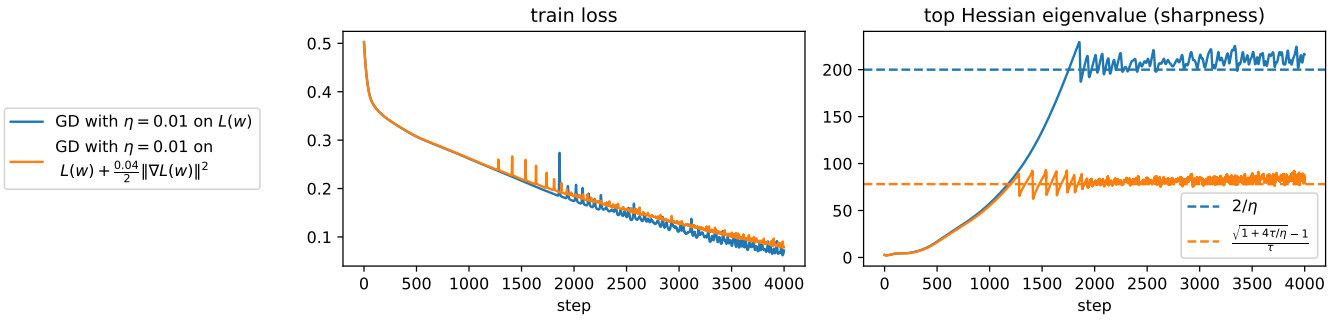


Figure 27: **A dangerous confounder: explicit *gradient* regularization induces stronger implicit *curvature* regularization.** We train a CNN on a subset of CIFAR-10 using MSE loss. In blue, we run gradient descent with step size $\eta = 0.01$ on the original objective $L(w)$. In orange, we run gradient descent with step size $\eta = 0.01$ on the implicitly regularized objective $L(w) + \frac{\tau}{4}\|L(w)\|^2$, with $\tau = 0.04$. This mimics an attempt to capture the $\eta = 0.04$ dynamics with an explicit gradient regularizer. Observe that the explicit *gradient* regularizer implicitly affects the *curvature* dynamics, causing the sharpness to saturate at $\frac{\sqrt{1+4\tau/\eta}-1}{\tau} \approx 78.08$ instead of at $2/\eta = 200$. We believe that similar effects may be responsible for reports in the literature (e.g. Geiping et al. (2022)) that explicit gradient norm regularization recovers the beneficial effects of large learning rates and small batch sizes.

is optimized using a standard optimization algorithm, this could induce stronger implicit curvature regularization which substantially affects the trajectory and the generalization performance.

Momentum Finally, we note that it is plausible that the IGR effect is negligible for vanilla gradient descent but relevant when momentum is used, as momentum amplifies the strength of the IGR effect (Ghosh et al., 2023).

C.1.1 Implementation details

To discretize the IGR-penalized gradient flow eq. (189), we used a forward Euler scheme:

$$w(t + dt) = w(t) - \eta\, dt\, \left[\nabla L(w(t)) + \tfrac{\eta}{2} H(w(t))\nabla L(w(t))\right]. \tag{192}$$

We dynamically adapt the discretization step size dt based on the current sharpness (which we are already measuring). On a quadratic function $L(w) = \frac{1}{2}Sw^2$ with sharpness w, the Euler method eq. (192) is convergent so long as:

$$dt \leq \frac{2}{\eta S + \frac{1}{2}\eta^2 S^2}.$$

To be on the safe side, and to try to avoid any implicit effects, we use a discretization step size of one-quarter that threshold. In particular, at every integer time t, we compute the sharpness $S(w)$, and set:

$$m = \lceil 2\eta S + \eta^2 S^2 \rceil \quad \text{and} \quad dt = 1/m,$$

and we take m Euler steps eq. (192) with discretization step size dt.

C.2 Failure mode: higher-order terms

Our theory models the objective using a local cubic Taylor approximation. Sometimes, however, a cubic Taylor expansion is inadequate to capture the dynamics within the critical subspace, and this gives rise to a failure mode for the central flow, which was previously discussed in Damian et al. (2023, Appendix F).

This failure mode is illustrated in Figure 28, which depicts a stretch of gradient descent where one eigenvalue is at the edge of stability and where the cyclic EOS dynamics have collapsed to a period-2 fixed point. (This makes for a simpler setting than the full cyclic dynamics, which helps us better illustrate the issue.) Observe that the sharpness measured at the (second-order) midpoints between the gradient descent iterates is noticeably *lower* than $2/\eta$, whereas the sharpness along the central flow is strictly *equal* to $2/\eta$. Further, observe that the actual squared displacement between gradient descent and the central flow is noticeably different from the central flow's prediction for this value, $\sigma^2(t)$. This means that the central flow is applying the wrong strength of implicit sharpness regularization, which will cause error to accumulate over the long run.

These issues arise because the loss function along the top Hessian eigenvector is not well-modeled by its cubic Taylor expansion. In Figure 29, at various points during this stretch of training, we plot the curvature quantity $u^\top H(w)u$ along the line between two successive iterates, i.e. for $w \in \{\alpha w_t + (1-\alpha)w_{t+1} : 0 \le \alpha \le 1\}$. Here, u is the top Hessian eigenvector measured at the midpoint between the two iterates $\overline{w} := \frac{1}{2}(w_t + w_{t+1})$. We also plot the first-order Taylor approximation of this curvature quantity, $S(\overline{w}) + \nabla S(\overline{w})^\top (w - \overline{w})$, which arises from the local cubic Taylor approximation, as well as the second-order Taylor approximation of this curvature quantity, which arises from the local quartic Taylor approximation. Observe that the curvature along this line segment is not well-modeled by its first-order Taylor approximation. This is an indicator that the cubic Taylor approximation which we employ in our analysis is failing to hold within the local region that is being traversed via the oscillations.

By contrast, Figures 30 and 31 depicts a different deep learning problem where the central flow approximation is more accurate, and where the local curvature *is* well-described by the cubic Taylor expansion.

Please see Damian et al. (2023, Appendix F) for an extended discussion of this issue, in the special case of one unstable eigenvalue. In this setting, the loss function can either be *super*-quadratic along the top Hessian eigenvector, in which case the real curvature lies *above* than its first-order Taylor approximation, and the curvature at the midpoint is *less* than $2/\eta$; or it can be *sub*-quadratic, in which case the real curvature lies *below* its first-order Taylor approximation, and the curvature at the midpoint is *greater* than $2/\eta$. When multiple eigenvalues are unstable, we expect that the loss function could conceivably be subquadratic along some directions and superquadratic along others.

In the special case of one unstable eigenvalue, Damian et al. (2023) derived a correction to their constrained trajectory (analogous to our central flow) which they empirically showed to match the real gradient descent trajectory even in the

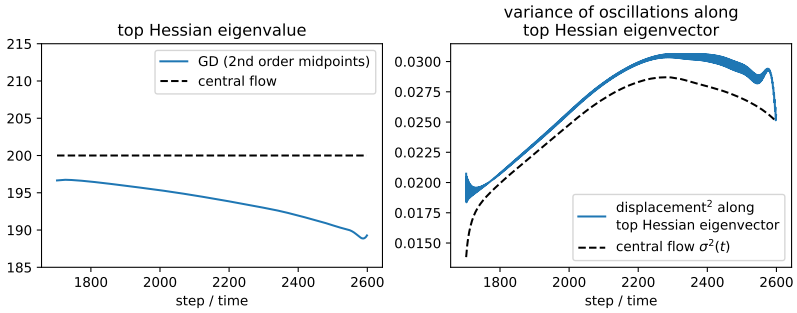


Figure 28: **Illustrating this failure mode for the central flow approximation.** This figure shows a segment of training which suffers from the failure mode discussed here. Observe that the sharpness along the gradient descent trajectory (measured at the second-order midpoints) is different from the sharpness of the central flow, which is locked at $2/\eta$. Further, the central flow poorly predicts the variance of the oscillations along the top Hessian eigenvector. *Details*: a CNN is trained on a two-class subset of CIFAR-10 with logistic loss.

sub/super-quadratic setting. Interestingly, with this correction, the implicit regularizer still takes the form $\sigma^2 \nabla S(w)$ for some σ^2; however, σ^2 cannot be determined solely from the local cubic Taylor approximation, and instead requires knowledge of the exact loss function along the top Hessian eigenvector direction. It would be interesting to re-derive this correction under the central flow framework, and to extend it to the setting of multiple unstable eigenvalues.

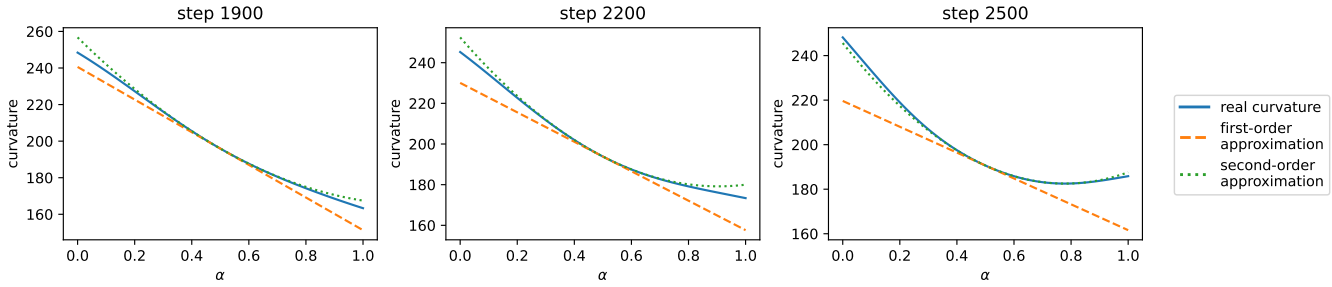


Figure 29: **When the central flow fails, the local cubic structure poorly predicts local curvature.** During the stretch of training depicted in Figure 28, at three different steps, we plot the curvature metric $u^\top H(w)u$ measured along the line segment between the current iterate and the next one: $\alpha w_t + (1-\alpha)w_{t+1},\ 0 \le \alpha \le 1$, where u denotes the top Hessian eigenvector measured at the midpoint between the two iterates. Observe that this curvature metric (blue) is poorly predicted by its linear approximation around the midpoint (orange), which is based on the local cubic structure of the loss.

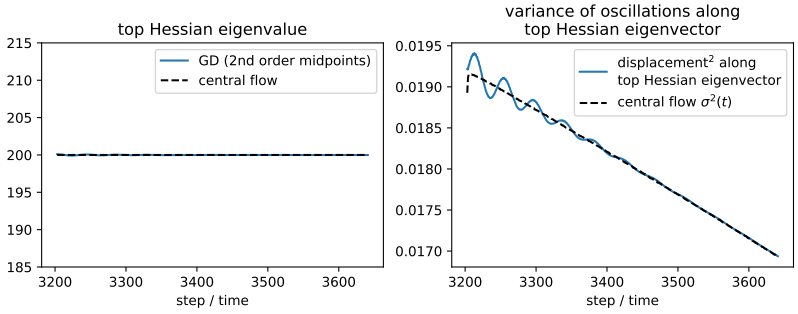


Figure 30: **A success case for the central flow.** This figure shows a segment of training which does *not* suffer from this failure mode. Observe that the sharpness along the gradient descent trajectory (measured at the second-order midpoints) is quite close to the sharpness along the central flow, which is locked at $2/\eta$. Further, observe that the central flow accurately predicts the variance of the oscillations along the top Hessian eigenvector. *Details*: a CNN is trained on a two-class subset of CIFAR-10 with MSE loss.

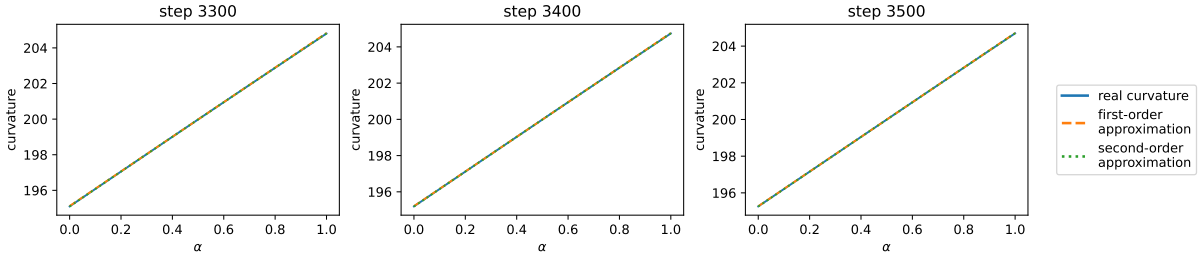


Figure 31: **When the central flow succeeds, the local cubic structure accurately predicts local curvature.** This figure shows the same curvature metric as Figure 29, but in the "success case" setting of Figure 30. Observe that here, the local curvature is well-predicted by its local linearization.

D Supplementary Figures

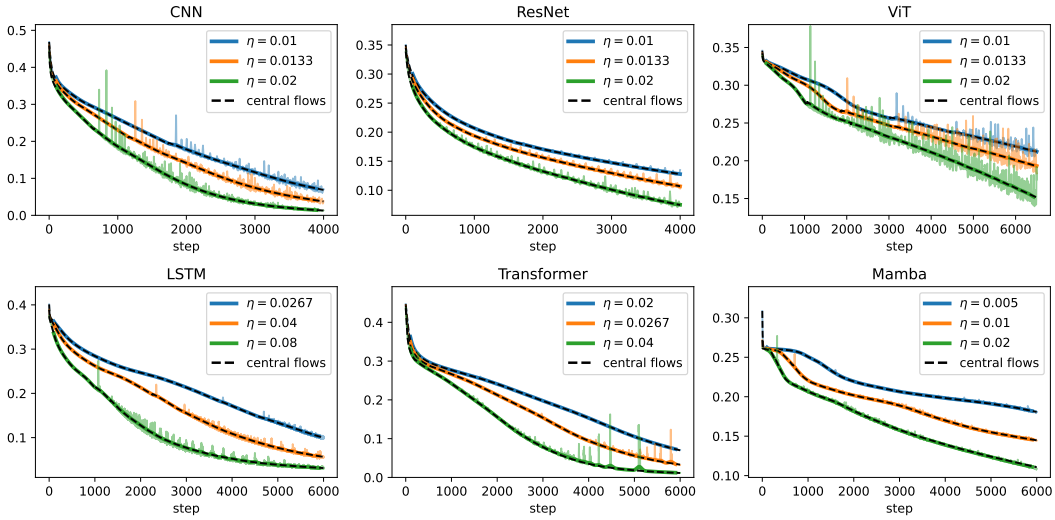


Figure 32(a): **Central flow predictions for gradient descent loss curves (MSE).** For six architectures, and three learning rates each, we compare the actual loss curve (colors) to the central flow's prediction for the time-averaged loss curve eq. (23) (black dashed). The faint colored lines are the raw loss curves; the thick colored lines are the time-averaged (smoothed) versions. Observe that the central flow accurately predicts the time-averaged loss curves.

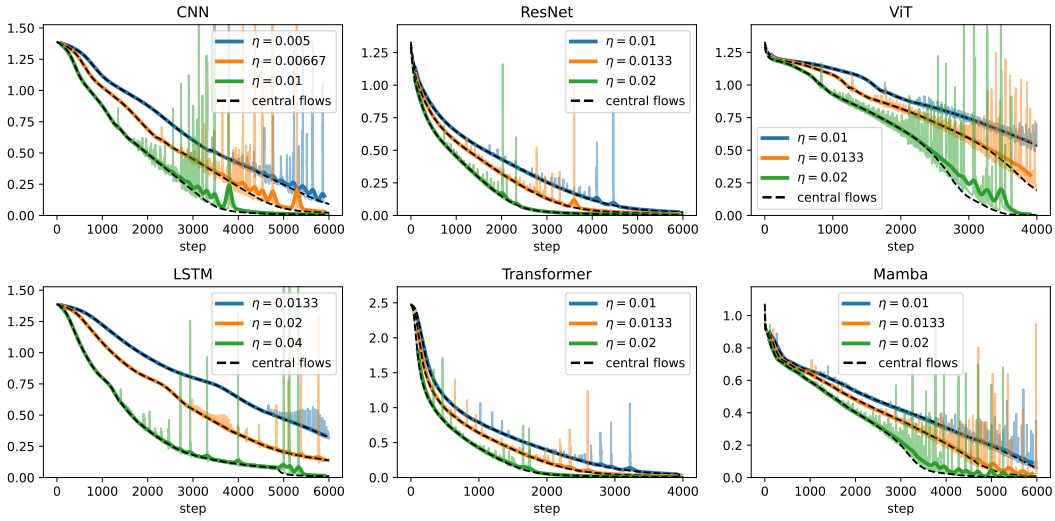


Figure 32(b): **Central flow predictions for gradient descent loss curves (CE).** Similar to Figure 32(a), but for cross-entropy loss. Observe that the prediction is sometimes a bit off, especially at the end, and especially with larger learning rates.

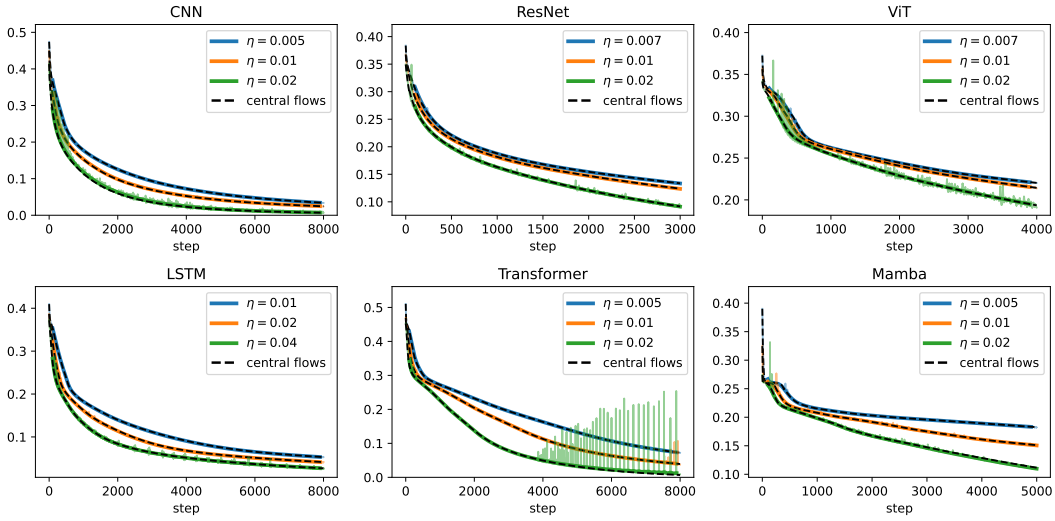


Figure 33(a): **Central flow predictions for Scalar RMSProp loss curves (MSE).** For six architectures, and three learning rates each, we compare the actual loss curve (colors) to the central flow's prediction for the time-averaged loss curve (black dashed). The faint colored lines are the raw loss curves; the thick colored lines are the time-averaged (smoothed) versions. Observe that the central flow accurately predicts the time-averaged loss curves.

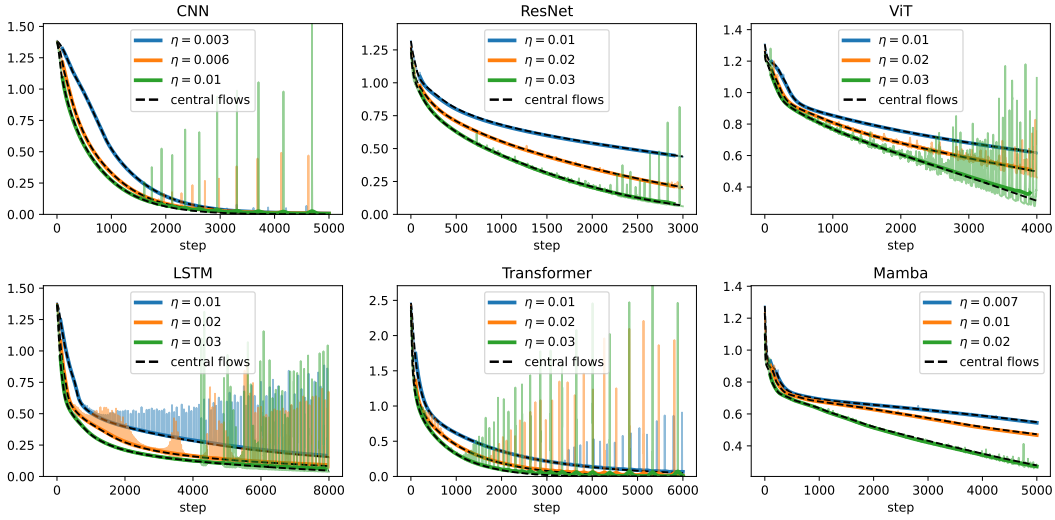


Figure 33(b): **Central flow predictions for Scalar RMSProp loss curves (CE).** Similar to Figure 33(a) but for cross entropy loss.

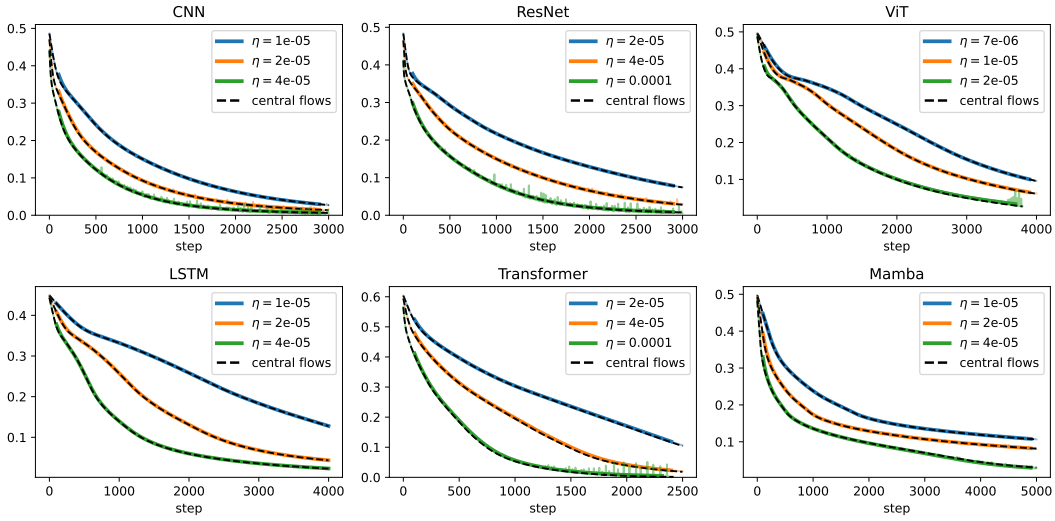


Figure 34(a): **Central flow predictions for RMSProp loss curves (MSE).** For six architectures, and three learning rates each, we compare the actual loss curve (colors) to the central flow's prediction for the time-averaged loss curve (black dashed). The faint colored lines are the raw loss curves; the thick colored lines are the time-averaged (smoothed) versions. Observe that the central flow accurately predicts the time-averaged loss curves.

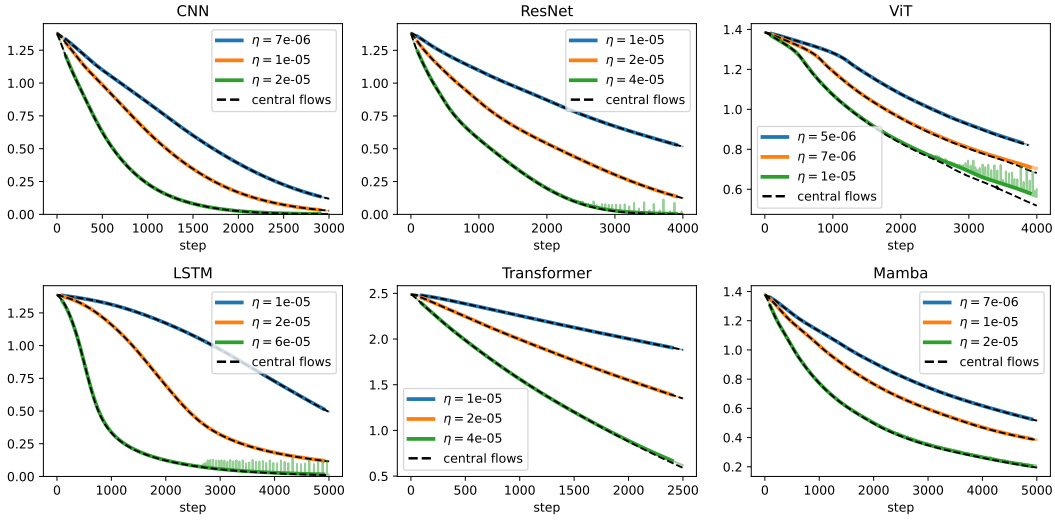


Figure 34(b): **Central flow predictions for RMSProp loss curves (CE).** Similar to Figure 34(a) but for cross-entropy loss.

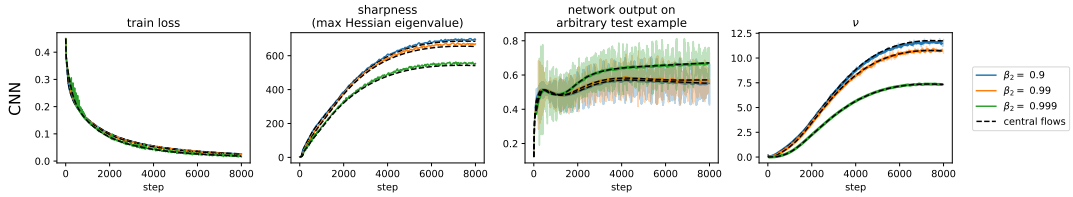


Figure 36: **Validating the Scalar RMSProp central flow across β_2 values**. We run both Scalar RMSProp and its central flow at multiple values of the β_2 hyperparameter (colors). Observe that the central flows (black) well-approximate the trajectory of Scalar RMSProp. *Details*: a CNN is trained on a subset of CIFAR-10 with MSE loss, $\eta = 0.01$, and bias correction.

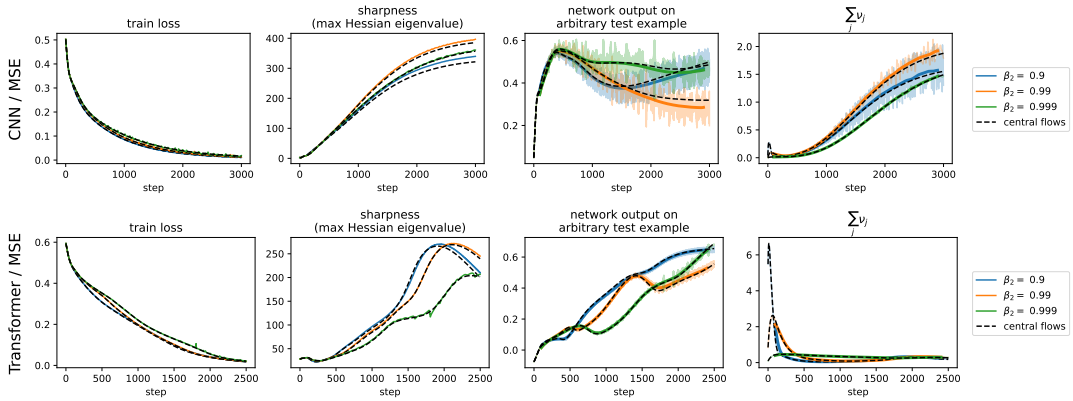


Figure 35: **Validating the RMSProp central flow across β_2 values**. We run both RMSProp and its central flow at multiple values of the β_2 hyperparameter (colors). Observe that the central flows (black) well-approximate the trajectory of RMSProp. *Details*: the top row is a CNN trained on a subset of CIFAR-10 with MSE loss, $\eta =$ 2e-5, $\epsilon =$ 1e-8, and bias correction. The bottom row is a Transformer trained on a synthetic sequence prediction task with MSE loss, $\eta =$ 4e-5, $\epsilon =$ 1e-8, and bias correction.

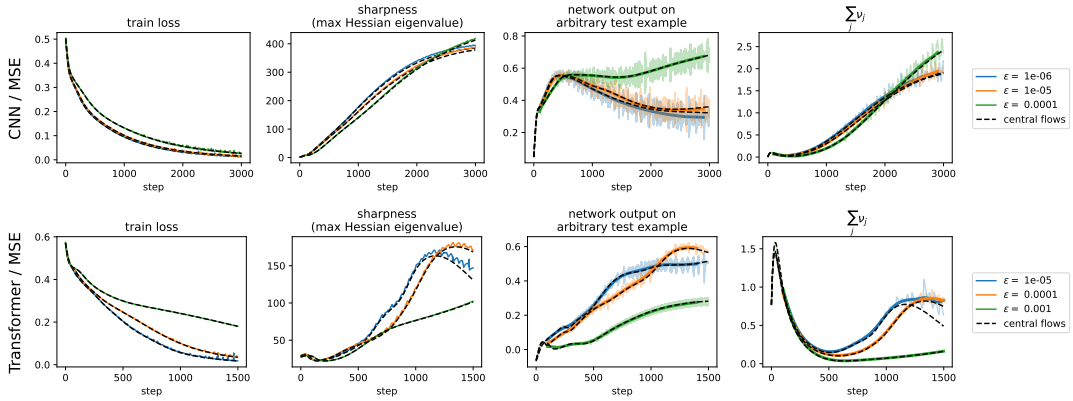


Figure 37: **Validating the RMSProp central flow across ϵ values**. We run both RMSProp and its central flow at multiple values of the ϵ hyperparameter (colors). Observe that the central flows (black) well-approximate the trajectory of RMSProp. *Details*: the top row is a CNN trained on a subset of CIFAR-10 with MSE loss, $\eta =$ 2e-5, $\epsilon =$ 1e-8, and bias correction. The bottom row is a Transformer trained on a synthetic sequence prediction task with MSE loss, $\eta =$ 4e-5, $\epsilon =$1e-8, and bias correction.

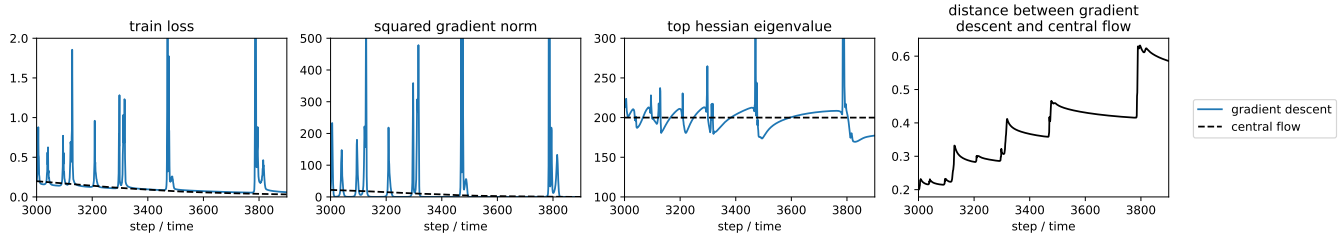


Figure 39: **Large spikes degrade the accuracy of the central flow approximation**. Every few hundred iterations, there is a large spike (visible in e.g. the loss and the gradient norm), which causes the distance between gradient descent and the central flow to jump. For reasons we do not understand, such large spikes are relatively common during full-batch training with cross-entropy loss, especially near the end of training. *Details*: a CNN is trained on a subset of CIFAR-10 with cross-entropy loss, using gradient descent with $\eta = 0.01$.

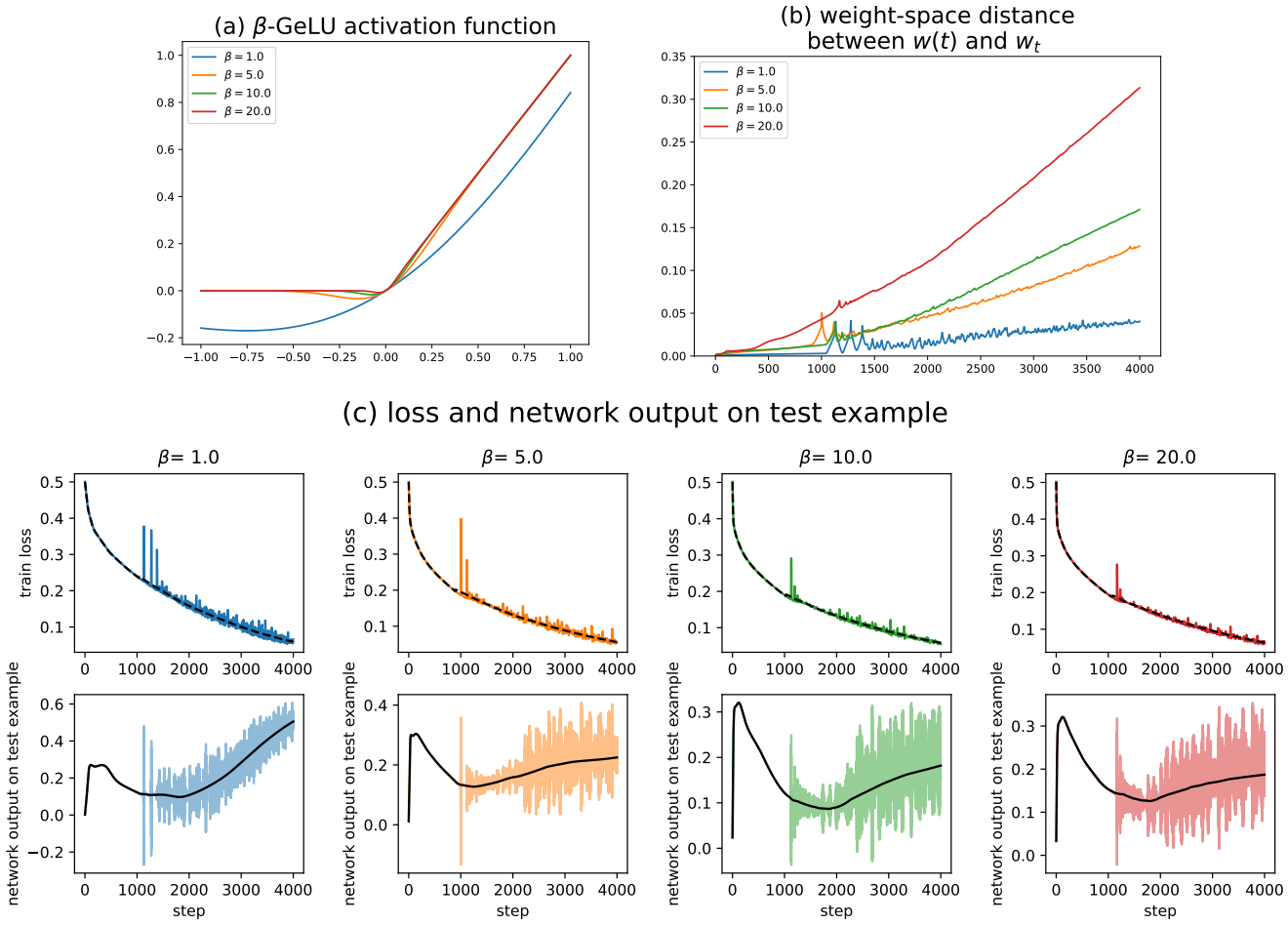


Figure 38: **Accuracy of central flow degrades as activation function becomes less smooth.** We consider networks with the β-GeLU activation function from Dauphin et al. (2024), defined as $x \mapsto x\Phi(\beta x)$ where Φ is the standard Gaussian CDF. This activation interpolates between (smooth) GeLU when $\beta = 1$ and (non-smooth) ReLU when $\beta = \infty$. Subfigure (a) plots this activation function with varying β. Subfigure (b) shows that when β is larger (i.e. when the activation is less smooth), the approximation error between the central flow $w(t)$ and the optimizer trajectory w_t grows faster. Subfigure (c) plots the loss curve, and the network's output on a test example, for both the optimizer trajectory and the central flow. Fortunately, even when $\beta = 20$, at which point β-GeLU is a very close approximation to ReLU, the central flow accurately predicts the overall training loss curve.

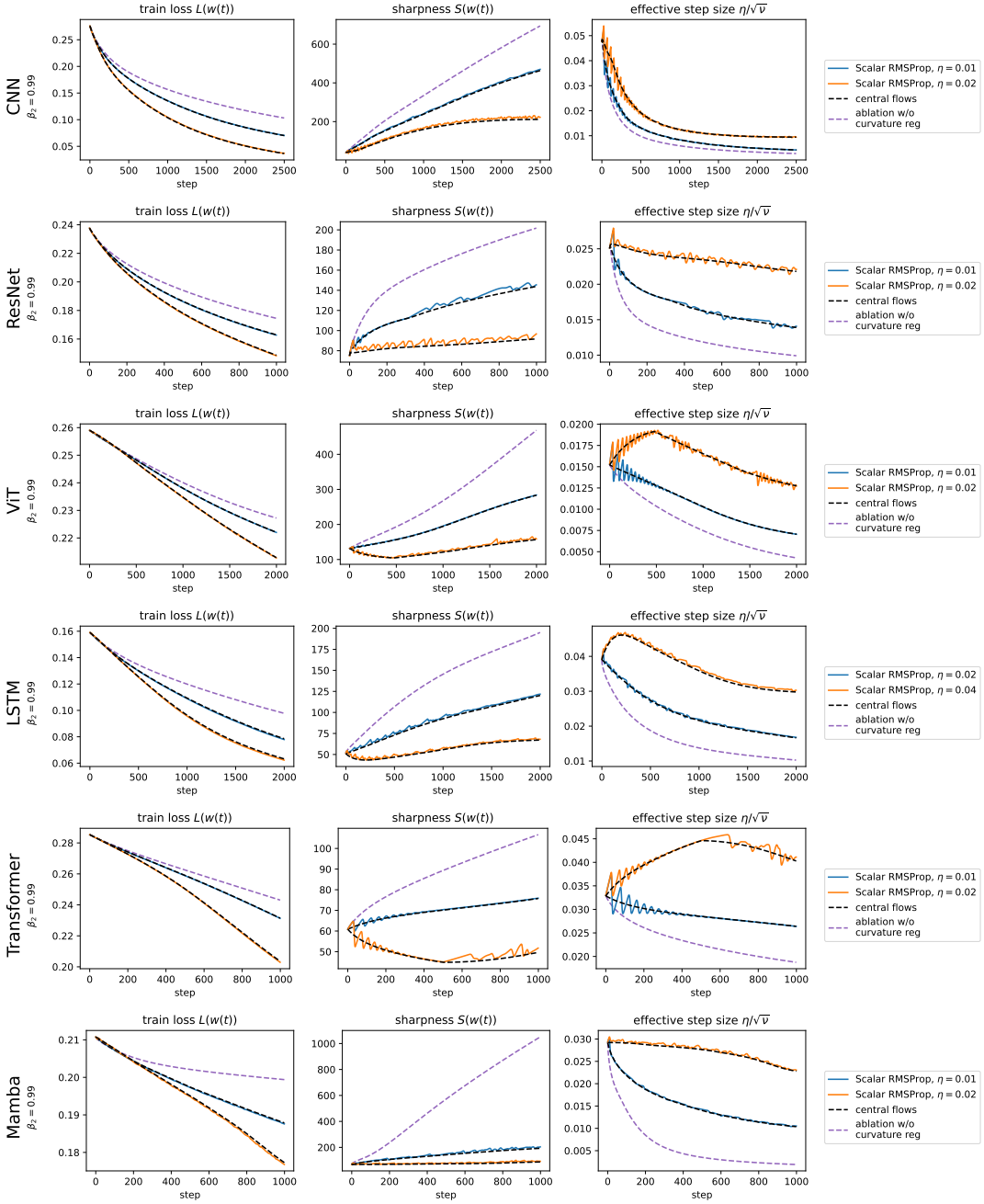


Figure 40: **Implicit curvature reduction accelerates optimization for Scalar RMSProp**. Starting from the same initialization, we run the Scalar RMSProp central flow at various learning rates, as well as an ablated flow $\frac{dw}{dt} = -\frac{2}{S(w)}\nabla L(w)$ with curvature regularization removed. These three flows all use the same step size strategy but differ in the strength of implicit curvature regularization. Initially, the flows with higher curvature regularization often optimize slower; however, over the longer run, they are able to take larger steps and optimize faster. Each row depicts a different deep learning setting. All experiments use MSE loss.[65]

[65]For discrete Scalar RMSProp, we show the train loss not at the iterates themselves but at the second-order midpoints between iterates $\hat{w}_t := \frac{1}{4}[2w_t + w_{t-1} + w_{t+1}]$, which removes most of the oscillations along the top eigenvectors. This makes the train loss directly comparable to the flow losses.

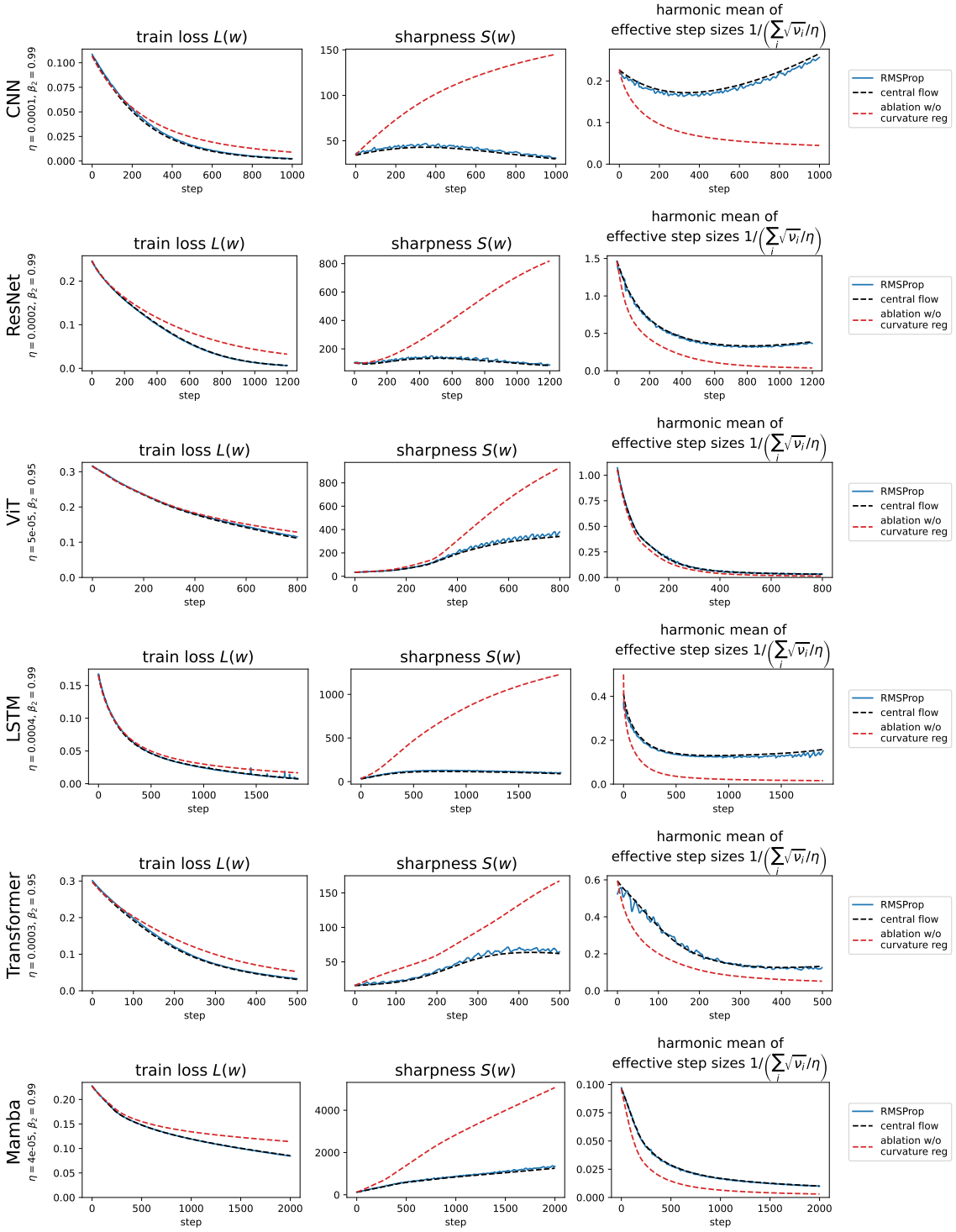


Figure 41: **Implicit curvature reduction accelerates optimization for RMSProp**. We compare RMSProp (blue) and its central flow (black) to an ablated flow (red) which leaves out the implicit curvature regularization, and maintains stability purely by the effect of oscillations on ν. Over time, RMSProp and the central flow navigate to lower-curvature regions (center), where they take larger steps (right), and optimize faster (left) than the ablated flow. Each row is a different DL setting. The left column plots the train loss,[66] the middle column plots the sharpness, and the right column plots the harmonic mean of the effective learning rates. These experiments all use MSE loss.

[66] For discrete RMSProp, we show the train loss not at the iterates themselves but at the second-order midpoints between iterates $\hat{w}_t := \frac{1}{4}[2w_t + w_{t-1} + w_{t+1}]$, which removes most of the oscillations along the top eigenvectors. This makes the train loss directly comparable to the flow losses.

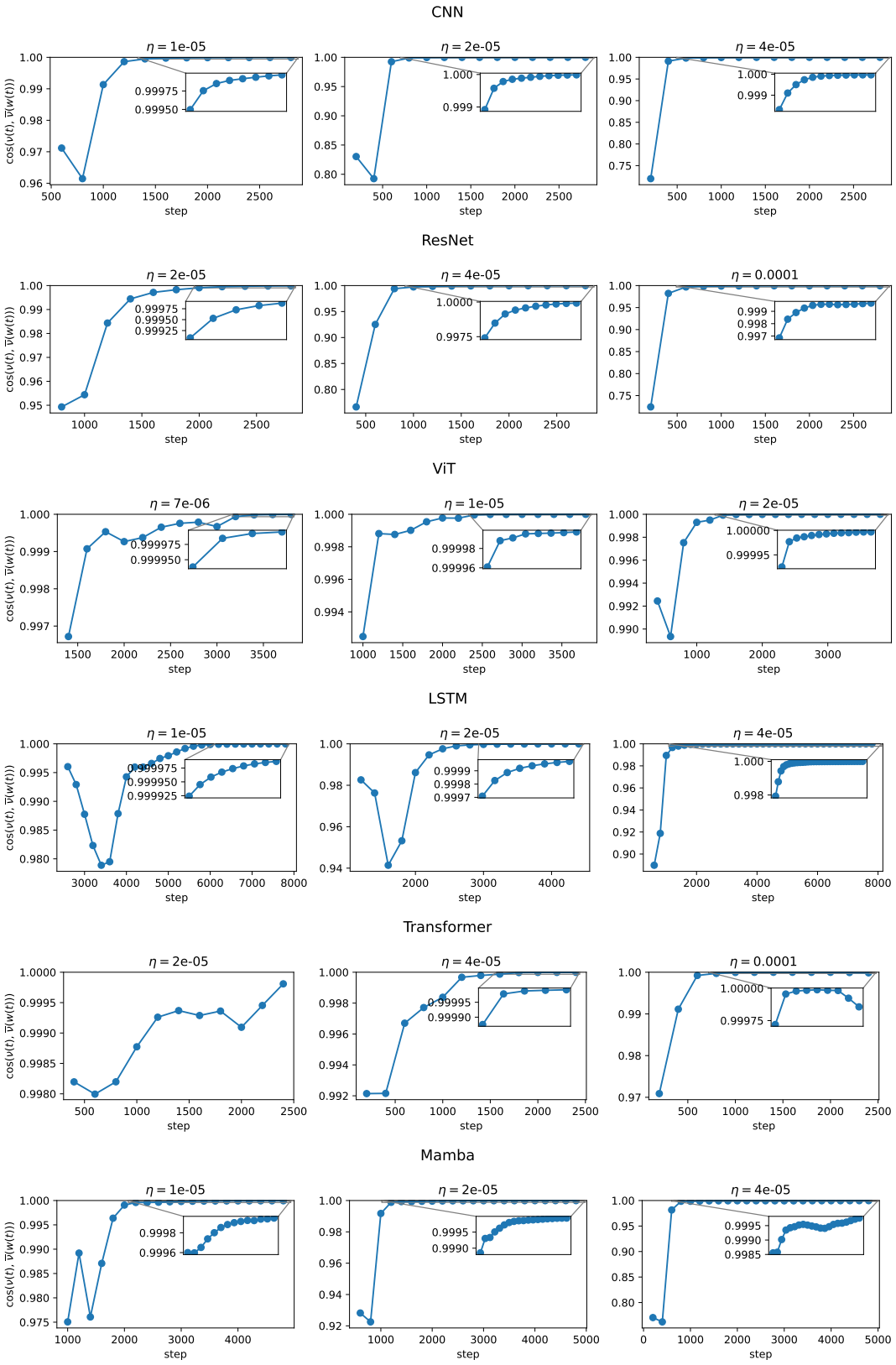


Figure 42(a): **The EMA ν reaches stationarity during training (MSE).** While running the RMSProp central flow, beginning at the time when training enters EOS, we monitor the cosine similarity between the EMA $\nu(t)$ and the stationary EMA $\bar{\nu}(w(t))$. This cosine similarity rises to high values (nearly 1) during training, implying that $\nu(t)$ reaches stationarity.

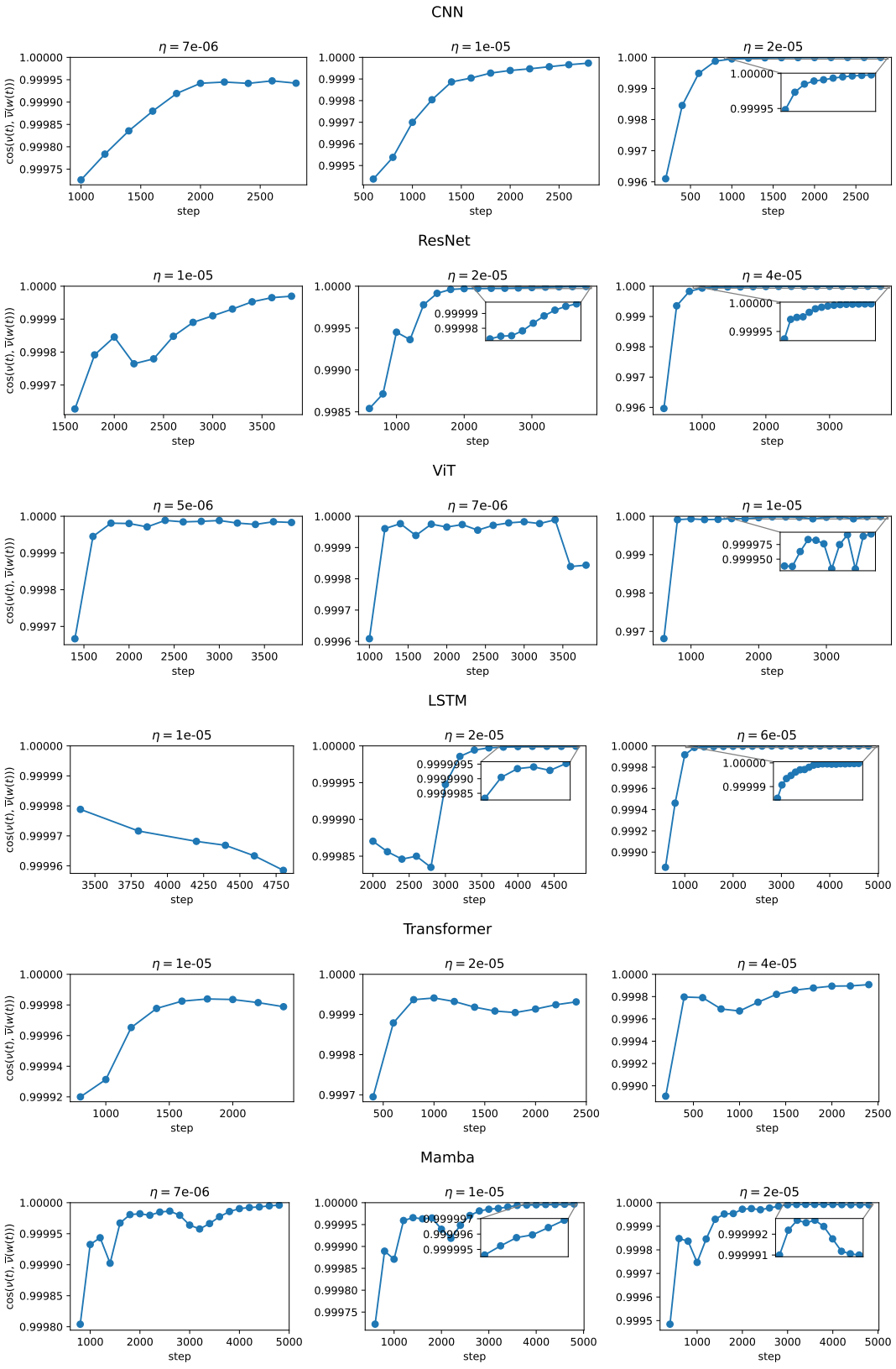


Figure 42(b): **The EMA ν reaches stationarity during training (cross-entropy).** This figure is analogous to Figure 42(a) but for cross-entropy loss.

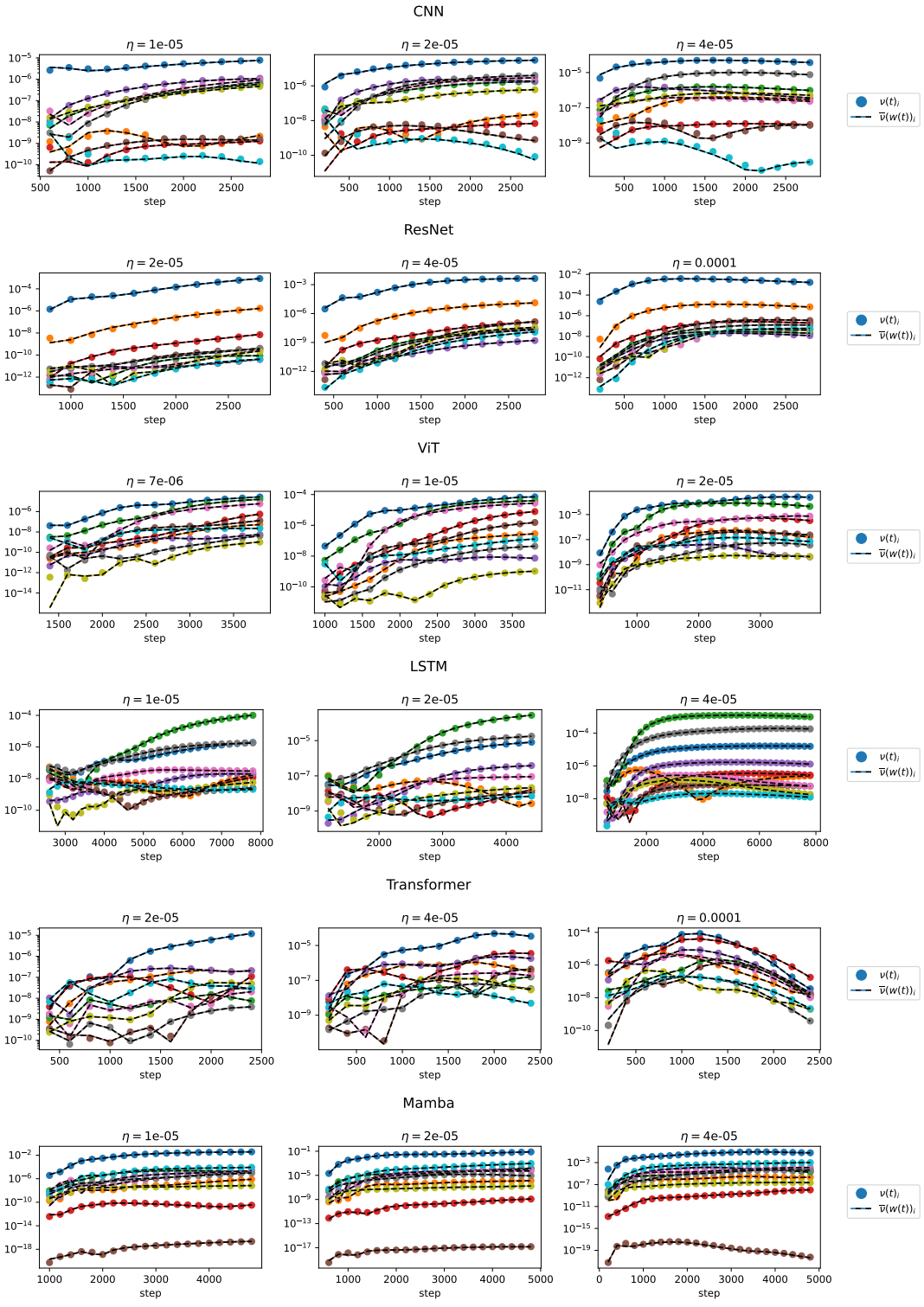


Figure 43(a): **Stationary EMA is accurate at a coordinate-wise level (MSE)**. While running the RMSProp central flow, starting at the time when training enters EOS, we plot the evolution of ten coordinates of the actual EMA $\nu(t)$ (dots) and the stationary EMA $\overline{\nu}(w(t))$ (half-black dashed lines). Each color is a different coordinate, and the ten coordinates are uniformly spaced throughout the network. We can see that, starting soon after training reaches EOS, the stationary EMA $\overline{\nu}(w(t))$ becomes an excellent approximation to the real EMA $\nu(t)$, on a coordinatewise level. We can also see that both the real EMA and the stationary EMA evolve significantly (in tandem) during this time.

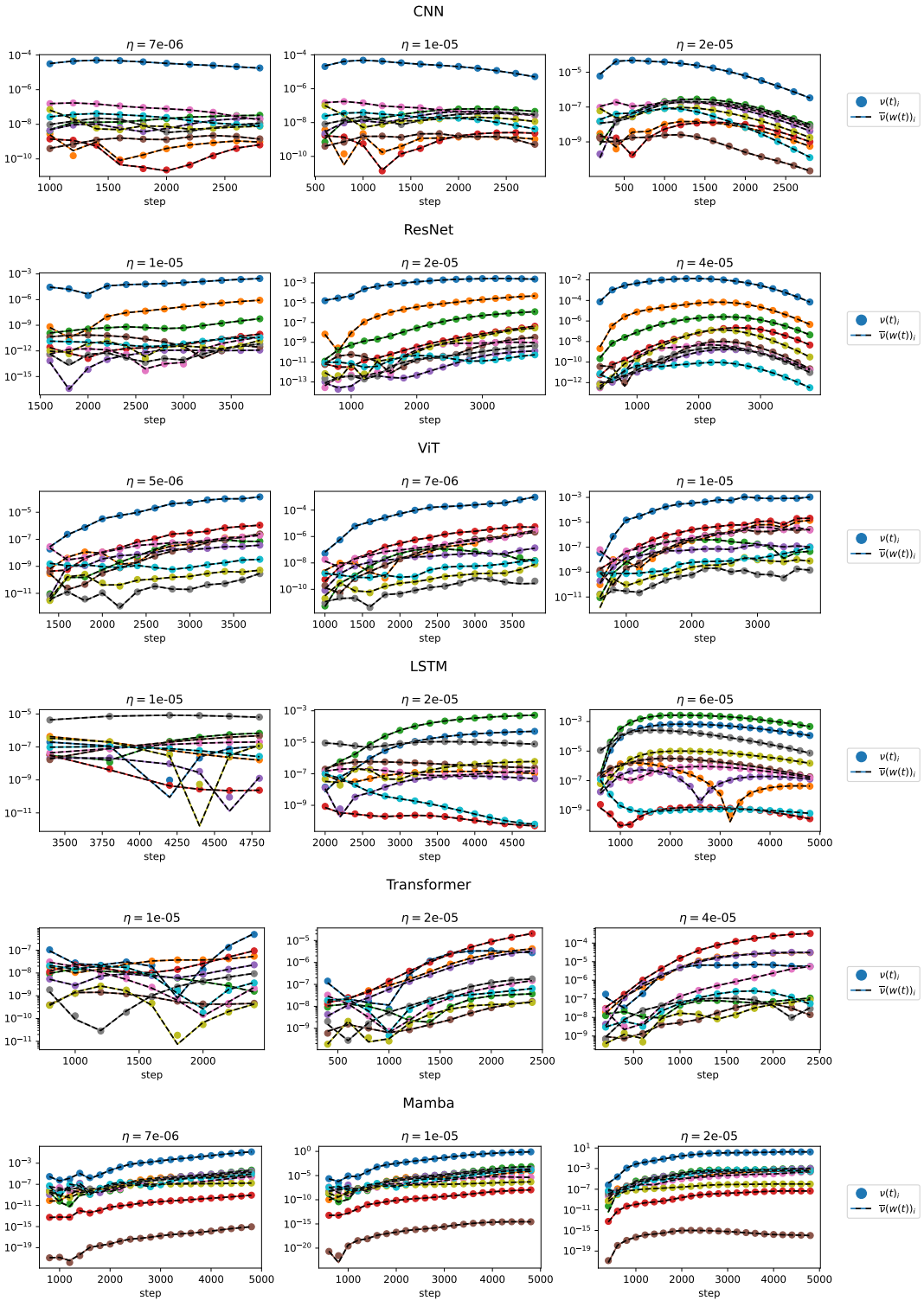


Figure 43(b): **Stationary EMA is accurate at a coordinate-wise level (CE)**. Analogous to Figure 43(a), but for cross-entropy loss.

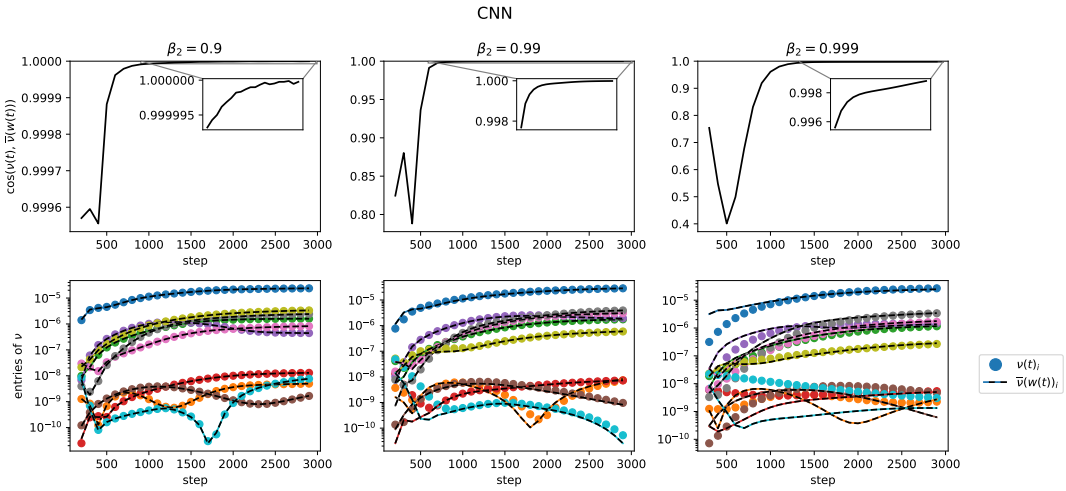


Figure 44: **Assessing how β_2 impacts the convergence of ν to stationarity.** For multiple values of β_2 (columns), we monitor the closeness between $\nu(t)$ and the stationary value $\overline{\nu}(w(t))$ over time. In particular, the top row reports the cosine similarity between these two vectors, and the bottom row compares ten individual coordinates (colors). As one might expect, we see that when β_2 is smaller, $\nu(t)$ converges faster to $\overline{\nu}(w(t))$ and the ultimate similarity is higher. Conversely, when β_2 is at the highest value of 0.999, some of the coordinates are noticeably off (e.g. the teal, brown, and orange coordinates.) *Details:* a CNN is trained on a subset of CIFAR-10 using MSE loss at $\eta =$ 2e-5, $\beta_2 \in \{0.9, 0.99, 0.999\}$, $\epsilon =$1e-8, and bias correction.

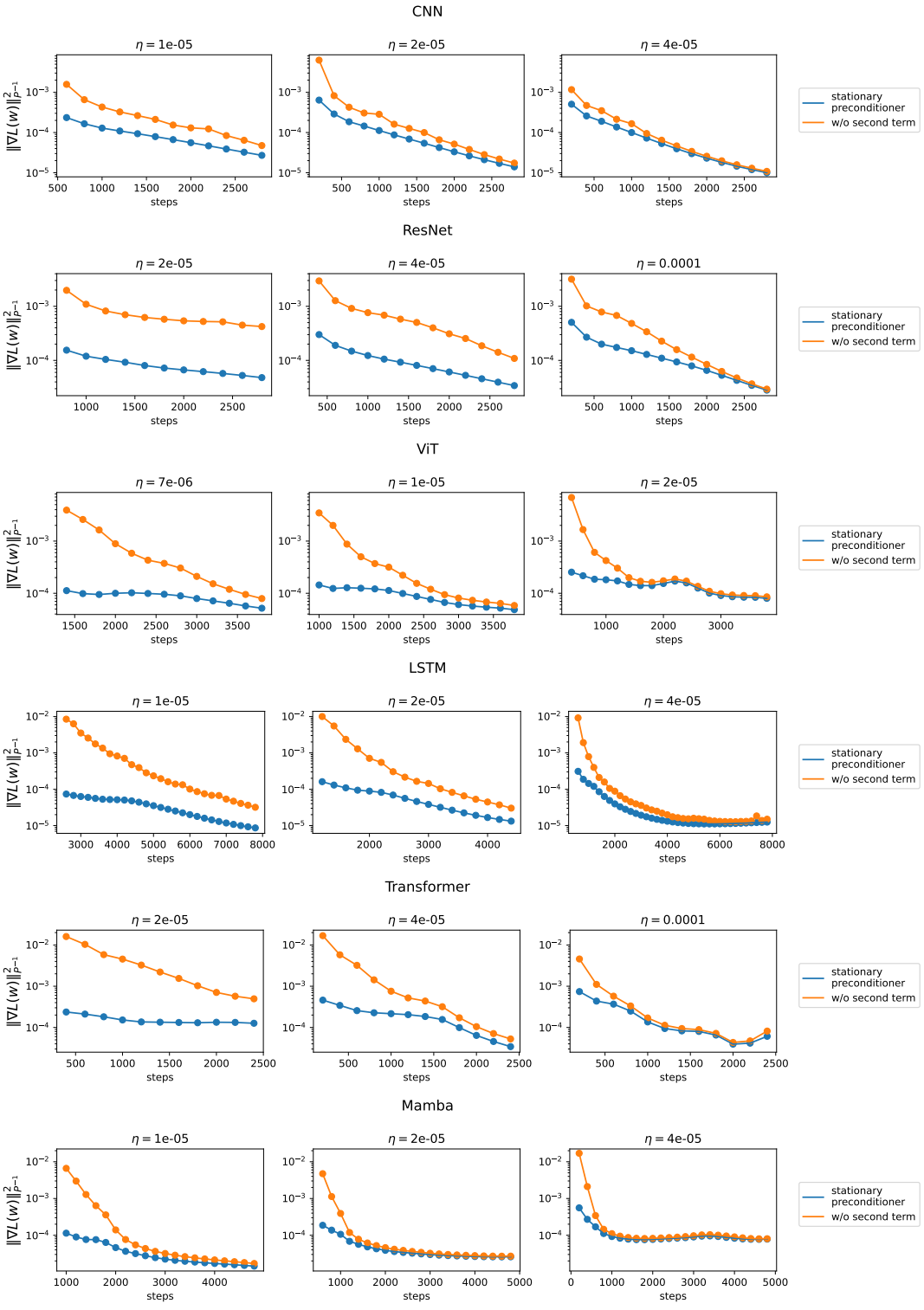


Figure 45(a): **RMSProp stationary preconditioner is suboptimal** (MSE). We compare the RMSProp stationary preconditioner, defined as the solution to the optimization problem eq. (35), to an alternative preconditioner defined as the solution to eq. (36), a similar optimization problem but without the second term in the objective. We assess each preconditioner P by reporting $\|\nabla L(w)\|^2_{P^{-1}}$, the instantaneous rate of decrease in the loss under the preconditioned gradient flow with preconditioner P. Observe that this value is higher under the alternative preconditioner (orange) than under the RMSProp stationary preconditioner (blue), meaning that the alternative preconditioner would decrease the loss faster. The gap between the two preconditioners tends to be smaller when η is larger, which is reasonable because the second term in eq. (35) is proportional to $\frac{1}{\eta^2}$.

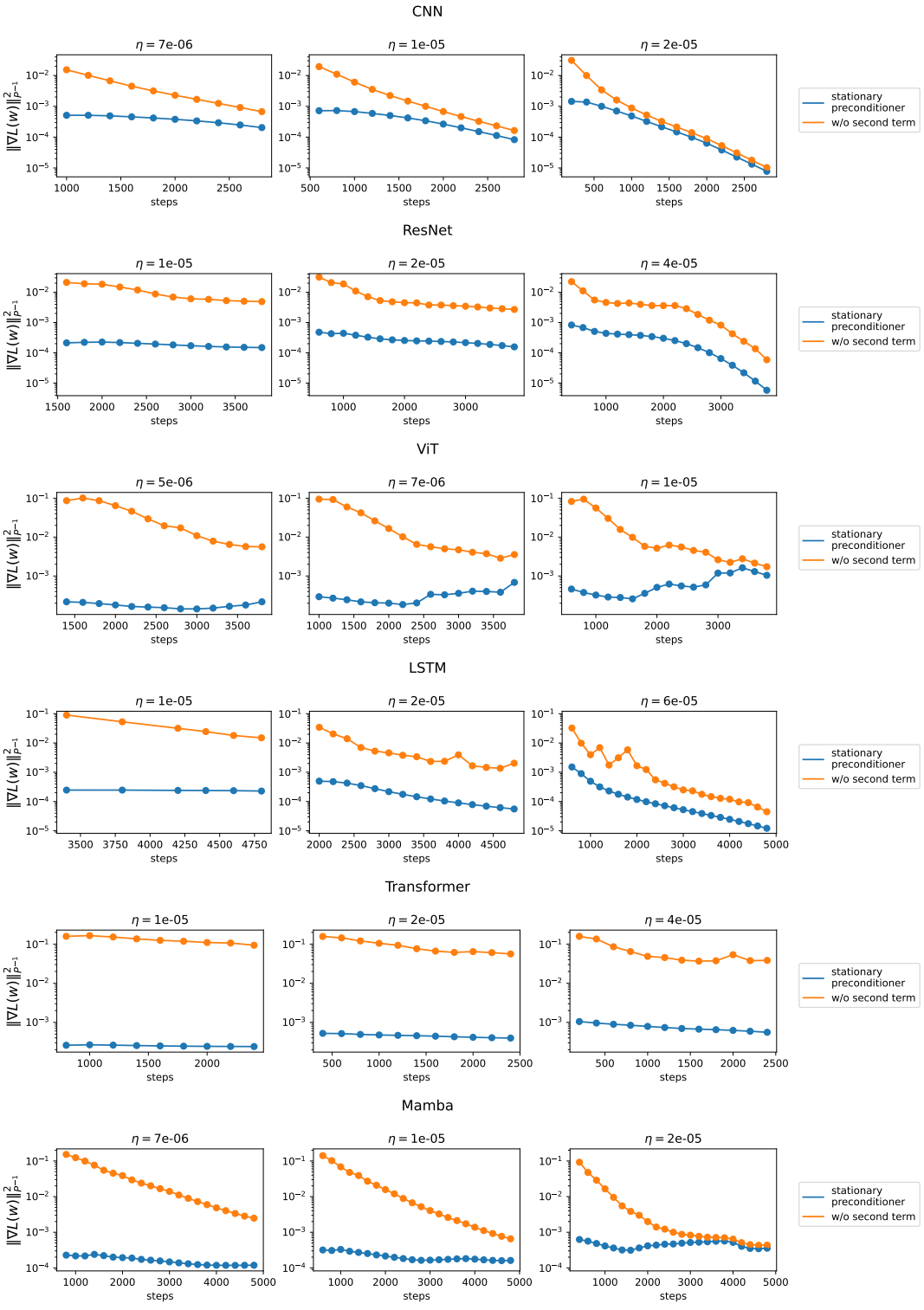


Figure 45(b): **RMSProp stationary preconditioner is suboptimal** (CE). This figure is analogous to Figure 45(a), but for cross-entropy loss.

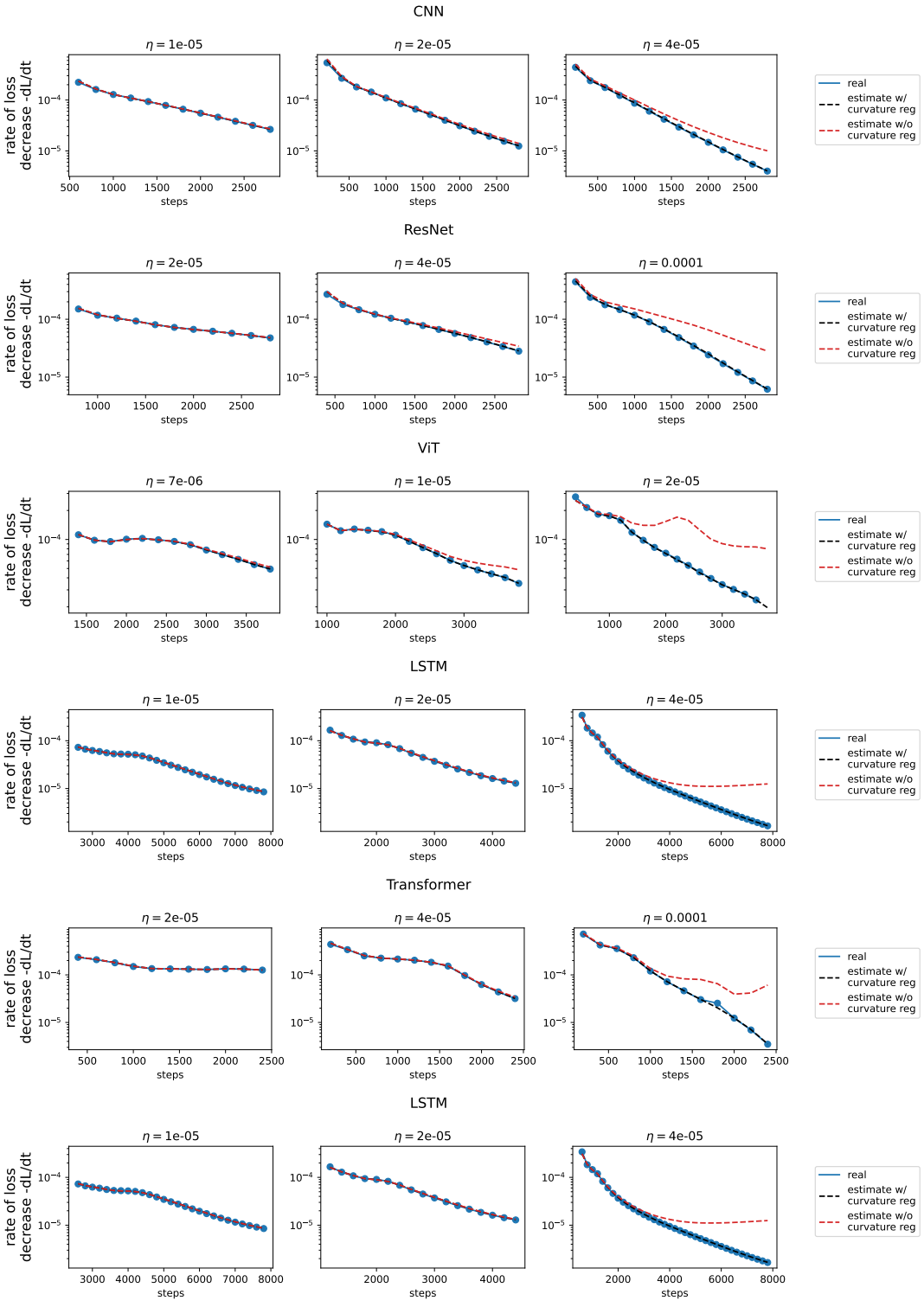


Figure 46(a): **Stationary flow accurately predicts the instantaneous speed of optimization (MSE)**. The stationary flow eq. (37), which incorporates an implicit curvature regularizer, predicts (black) the rate of loss decrease $-\frac{dL}{dt}$ (blue) more accurately than a naive estimate $\|\nabla L(w)\|^2_{P^{-1}(w)}$ (in red) which uses the stationary preconditioner but does not incorporate curvature regularization. Observe that the gap between the two estimates is larger when η is larger, suggesting that, like Scalar RMSProp, the implicit regularization of RMSProp increases in strength with η.

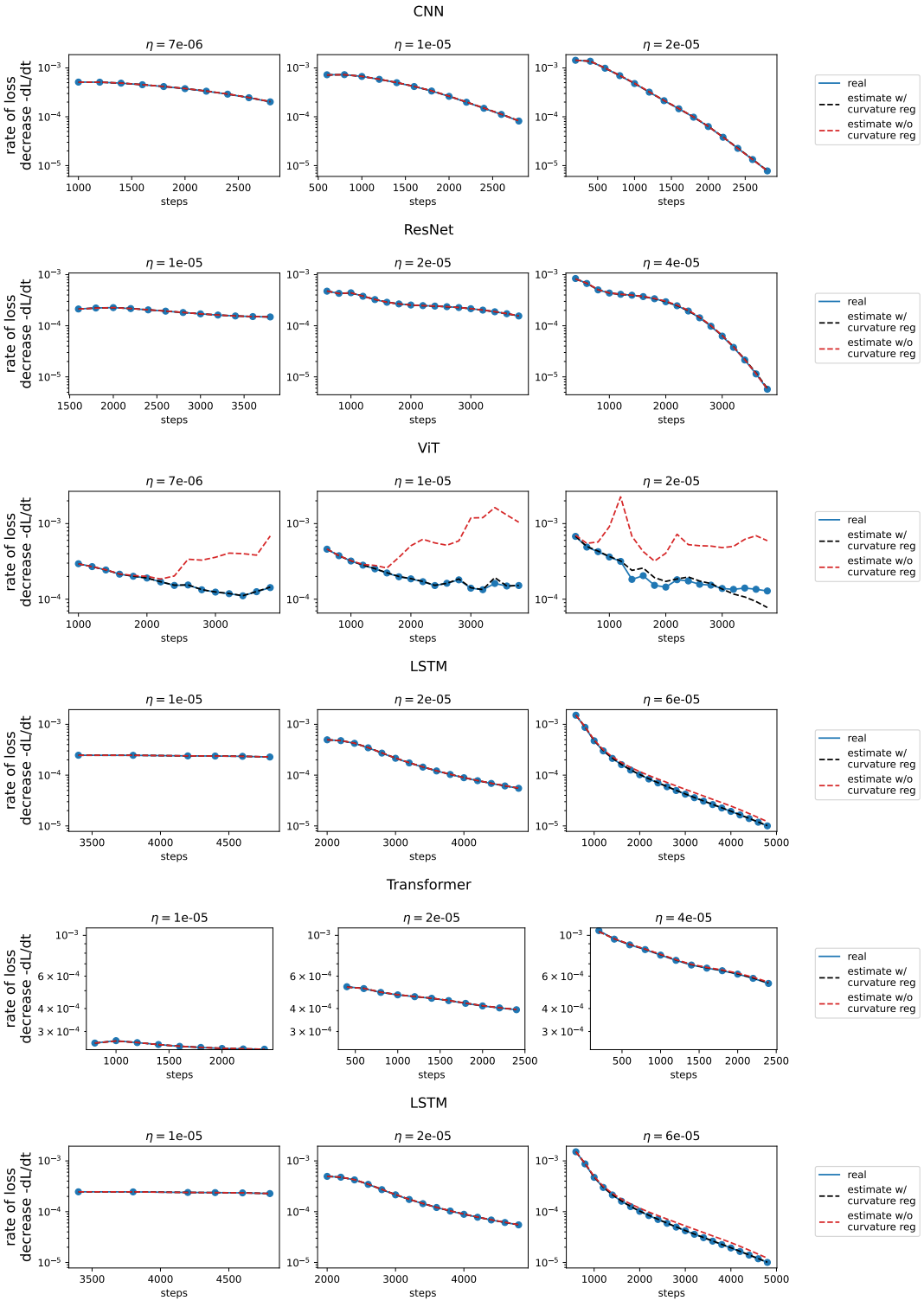


Figure 46(b): **Stationary flow accurately predicts the instantaneous speed of optimization (CE)**. Same as Figure 46(a), but with cross-entropy loss.

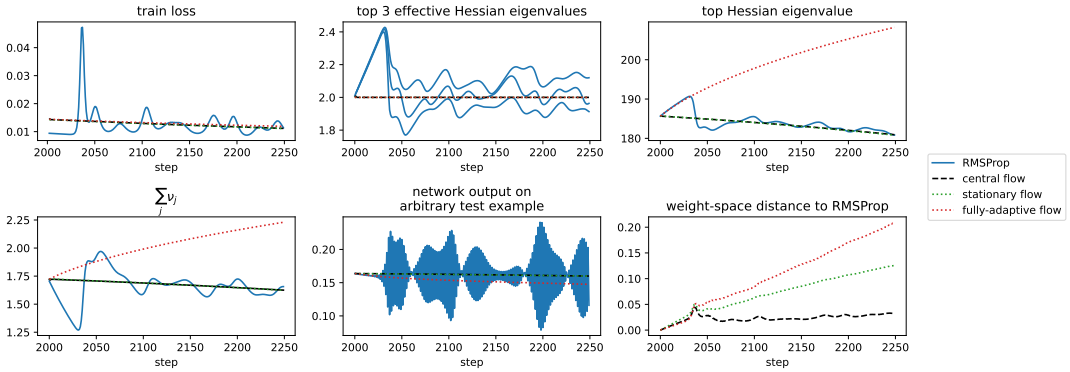


Figure 47(a): **Stationary flow can be accurate over moderate timescales**. Starting at a point during training when ν has reached stationarity, we run the stationary flow eq. (37) (in green) alongside both RMSProp (in blue) and the central flow (in black). As a baseline, we also run an ablated version of the stationary flow (in red) which adapts using the stationary ν but does not implicitly regularize curvature. Observe that the stationary flow accurately tracks the central flow (and, in turn, RMSProp), whereas the baseline is a worse approximation. This experiment uses a CNN trained on a subset of CIFAR-10 using MSE loss with hyperparameters $\eta =$ 4e-05, $\beta_2 = 0.99$, and $\epsilon =$1e-8.

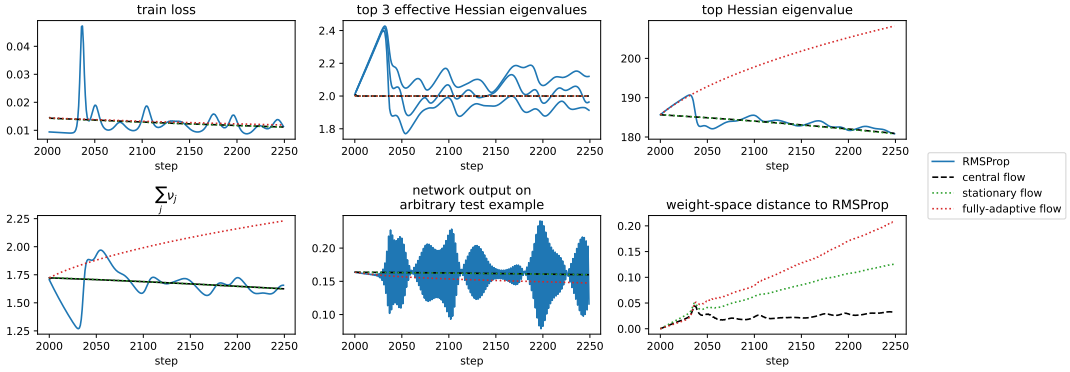


Figure 47(b): **Stationary flow can be accurate over moderate timescales**. Same as Figure 47(a), but using a Transformer trained on a synthetic sequence task using MSE loss with hyperparameters $\eta =$ 1e-4, $\beta_2 = 0.95$, and $\epsilon =$1e-8.

E Bulk Experimental Data

This section contains the bulk experimental data from our central flow experiments:

- Appendix E.1 contains gradient descent experiments. See Figure 48 for a fully annotated example of a gradient descent trajectory.
- Appendix E.2 contains Scalar RMSProp experiments. See Figure 51 for a fully annotated example of a Scalar RMSProp trajectory.
- Appendix E.3 contains RMSProp experiments. See Figure 54 for a fully annotated example of a RMSProp trajectory.

E.1 Gradient Descent

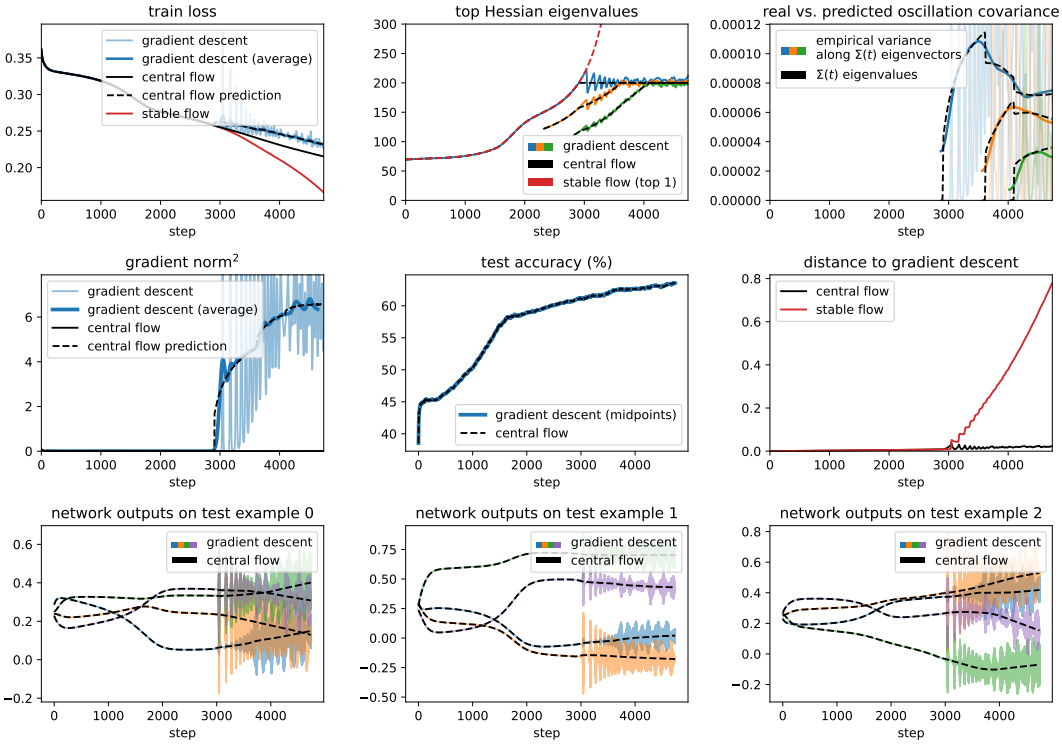


Figure 48: **Annotated example of a gradient descent experiment.** Using gradient descent with $\eta = 2/200$, we train a ViT on a subset of CIFAR-10. The central flow (black) accurately models the trajectory of gradient descent (blue), whereas gradient flow (red) takes a different path. As described in Appendix B.1, we terminate gradient flow once the sharpness gets too high.
Top left: The loss along the central flow (solid black) decreases monotonically, whereas the loss along the gradient descent trajectory (light blue) behaves non-monotonically once the dynamics enter EOS. While the gradient descent loss is higher than the central flow loss, the central flow can accurately predict the *time-averaged* loss along the gradient descent trajectory, using eq. (73) (dashed black); this can be seen to match the empirical time average of the gradient descent loss curve (dark blue). Finally, the train loss along gradient flow (in red) decreases faster, because it follows a different, unregularized path.
Top center: We plot the top three Hessian eigenvalues under gradient descent (colors) and under the central flow (black). Under GD, the top Hessian eigenvalues equilibrate around $2/\eta$; under the central flow they are fixed exactly at $2/\eta$. In red, we plot the top Hessian eigenvalue under the gradient flow, which rises beyond $2/\eta$. Note that for GD, we report the Hessian eigenvalues at the second-order midpoints (see Appendix B.1), rather than at the iterates themselves, as this makes for clearer plots.
Top right: We show that the central flow's $\Sigma(t)$ accurately predicts the covariance of the oscillations. In black, we plot the nonzero eigenvalues of $\Sigma(t)$; the number is always the same as the number of Hessian eigenvalues at $2/\eta$. In faint colors, we plot the squared magnitude of the displacement between gradient descent and the central flow along each eigenvector of $\Sigma(t)$. In thick colors, we plot the time-averages of these displacements, i.e. the empirical variance of the oscillations along each eigenvector of $\Sigma(t)$. Observe that the eigenvalues of $\Sigma(t)$ accurately predict the instantaneous variance of the oscillations along the corresponding eigenvectors, as we expect from eq. (76).
Middle left: We plot the squared gradient norm along the gradient descent trajectory (light blue) and its empirical time-average (dark blue). In dashed black, we plot the central flow's prediction eq. (74) for the time-averaged squared gradient norm along the trajectory; this prediction is quite accurate. In solid black, we plot the squared gradient norm along the central flow, which is much smaller, indicating that that most of the gradient norm comes from the oscillations.
Middle center: We plot the test accuracy under gradient descent (blue) and the central flow (black). For gradient descent, we report the test accuracy at second-order midpoints, as this removes much of the oscillations. Because the central flow matches the gradient descent trajectory, the test accuracy is nearly the same across both trajectories.
Middle right: The Euclidean distance in weight space between gradient descent and the central flow (black) stays small over time, indicating that these two trajectories stay close. By contrast, the distance between gradient descent and the *gradient* flow (red) grows rapidly once the dynamics enter EOS.
Bottom row: We show the network's final-layer predictions on three arbitrary examples. Under gradient descent (colors) these predictions oscillate due to the oscillations in weight space. Under the central flow (black), the predictions evolve smoothly while still following the same macroscopic path.

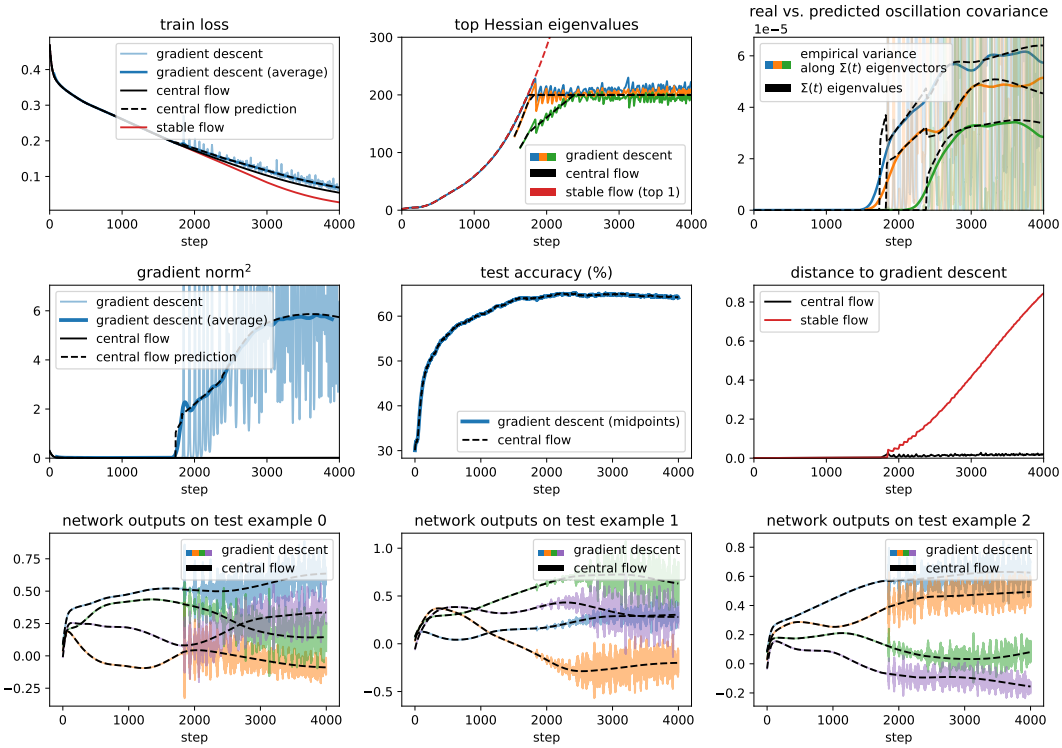


Figure 49.1: Gradient descent central flow for a CNN with MSE loss, $\eta = 0.005$.

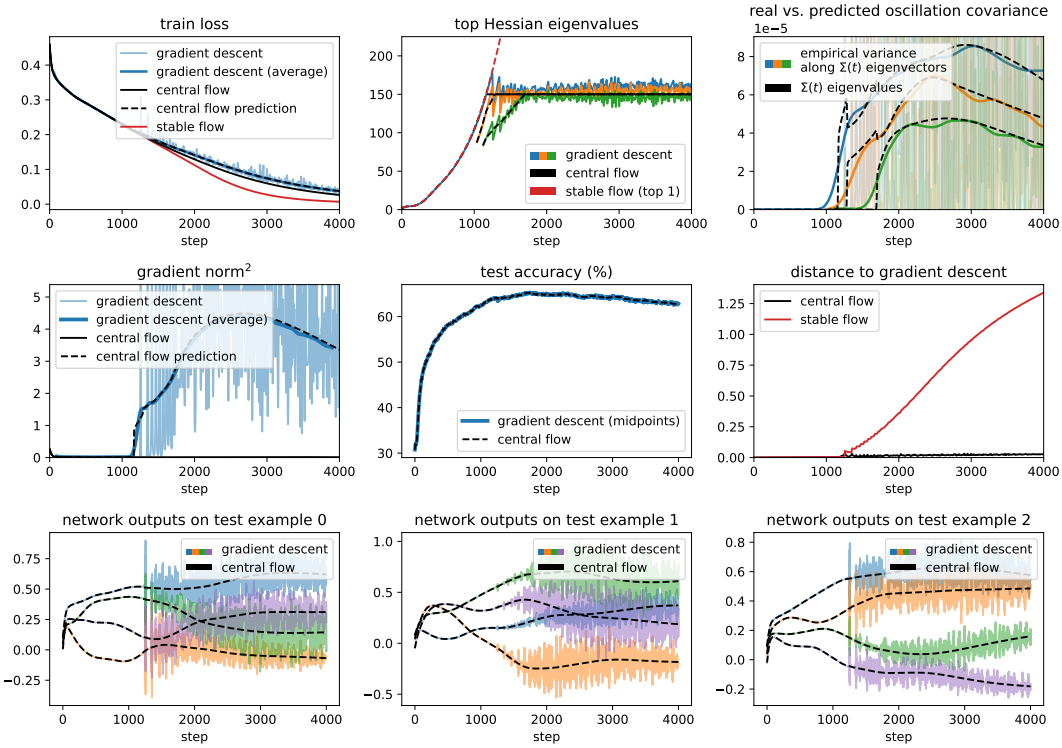


Figure 49.2: Gradient descent central flow for a CNN with MSE loss, $\eta = 0.006666$.

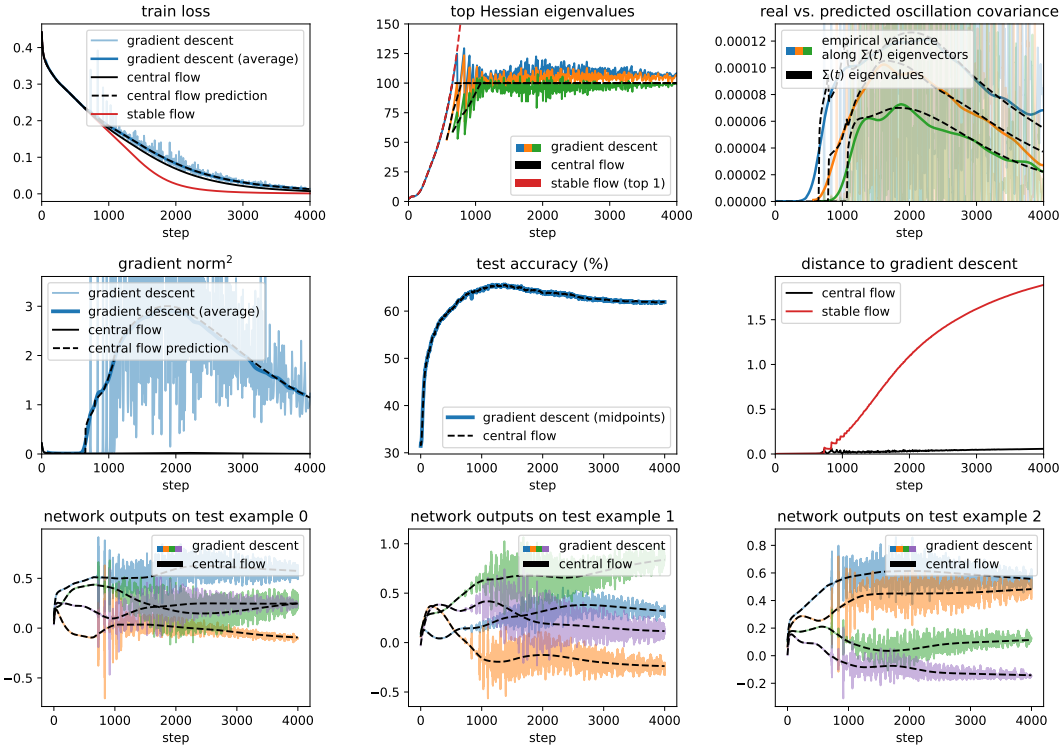


Figure 49.3: Gradient descent central flow for a CNN with MSE loss, $\eta = 0.01$.

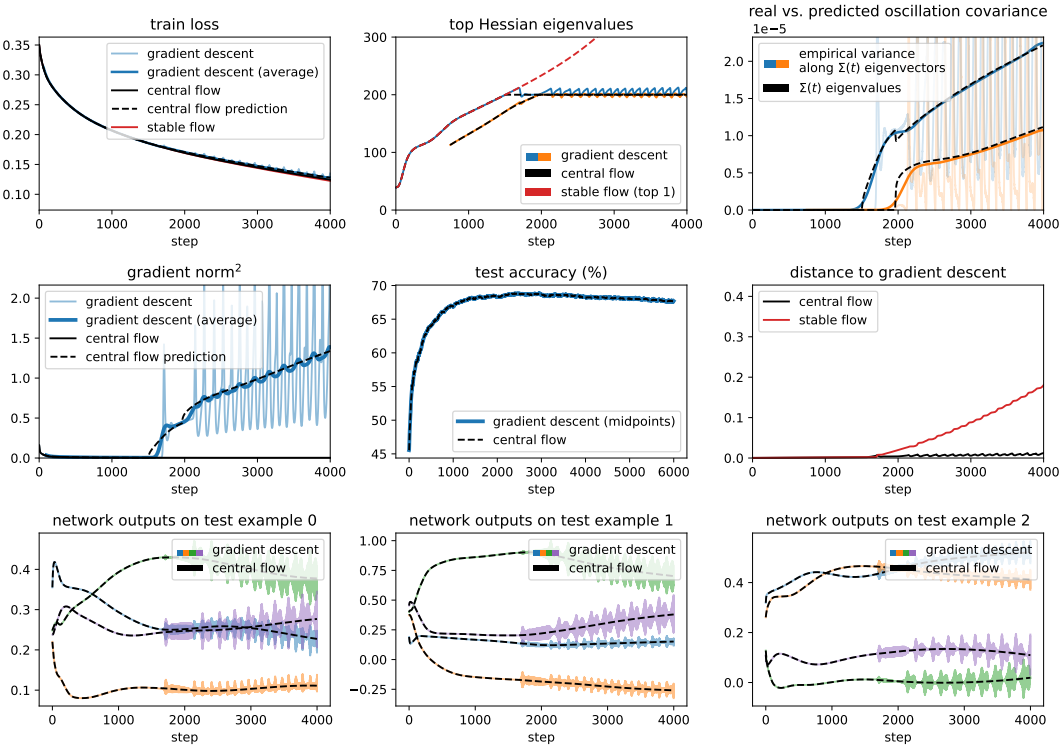


Figure 49.4: Gradient descent central flow for a ResNet with MSE loss, $\eta = 0.01$.

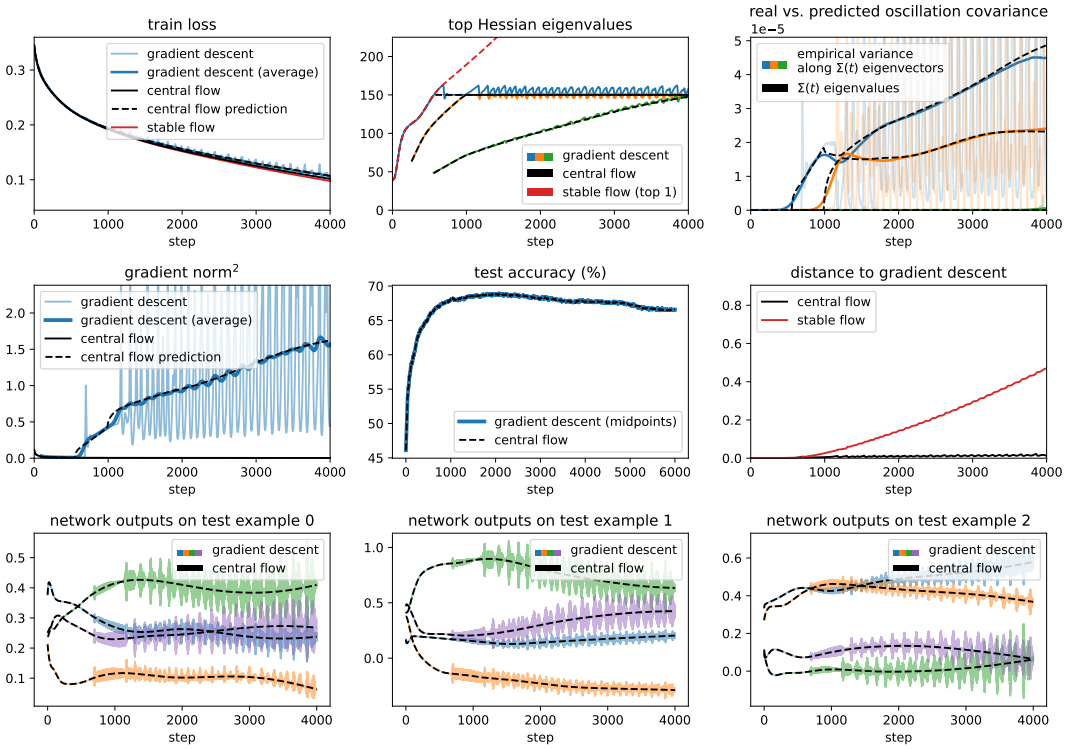


Figure 49.5: Gradient descent central flow for a ResNet with MSE loss, $\eta = 0.013333$.

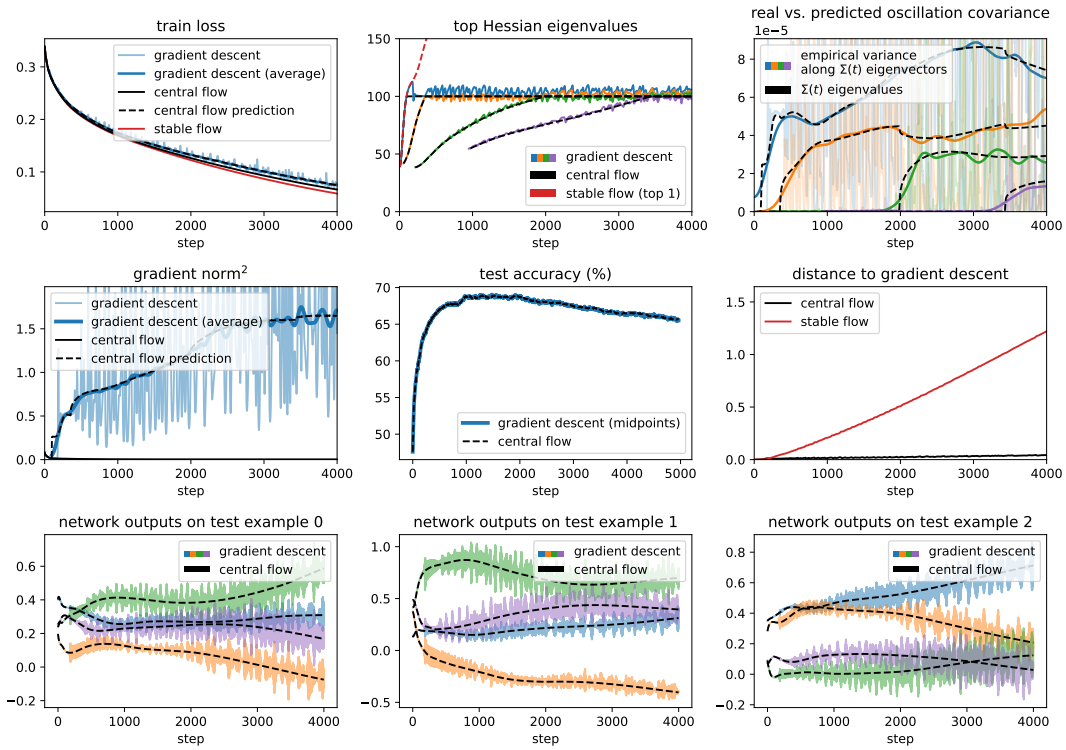


Figure 49.6: Gradient descent central flow for a ResNet with MSE loss, $\eta = 0.02$.

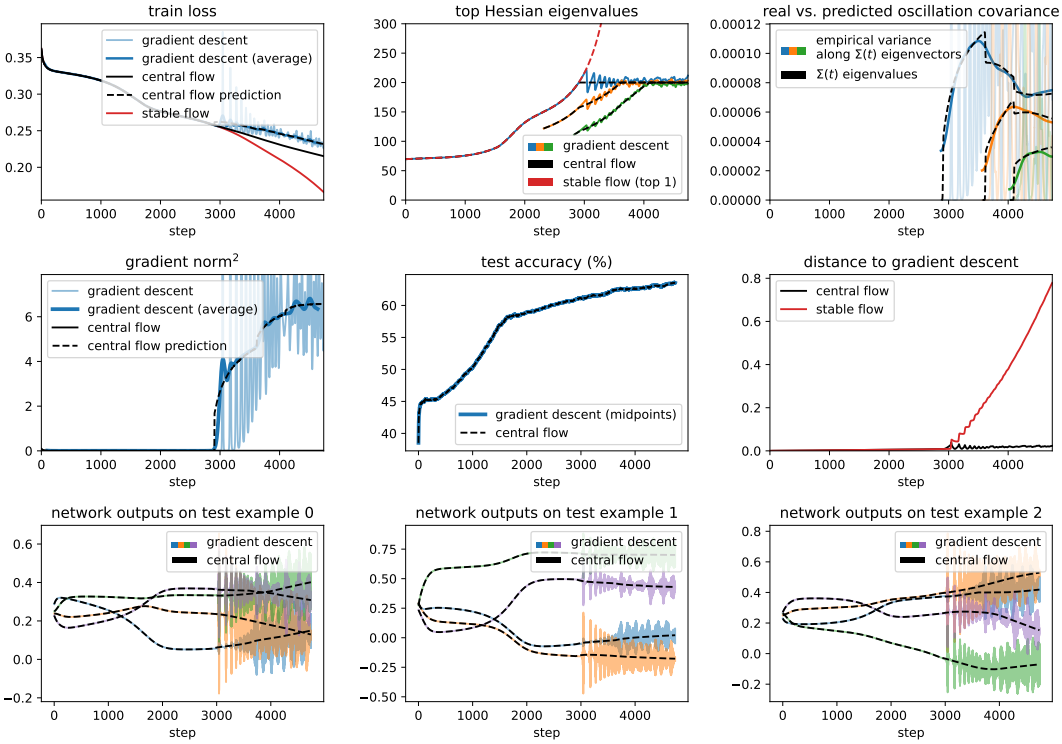


Figure 49.7: Gradient descent central flow for a ViT with MSE loss, $\eta = 0.01$.

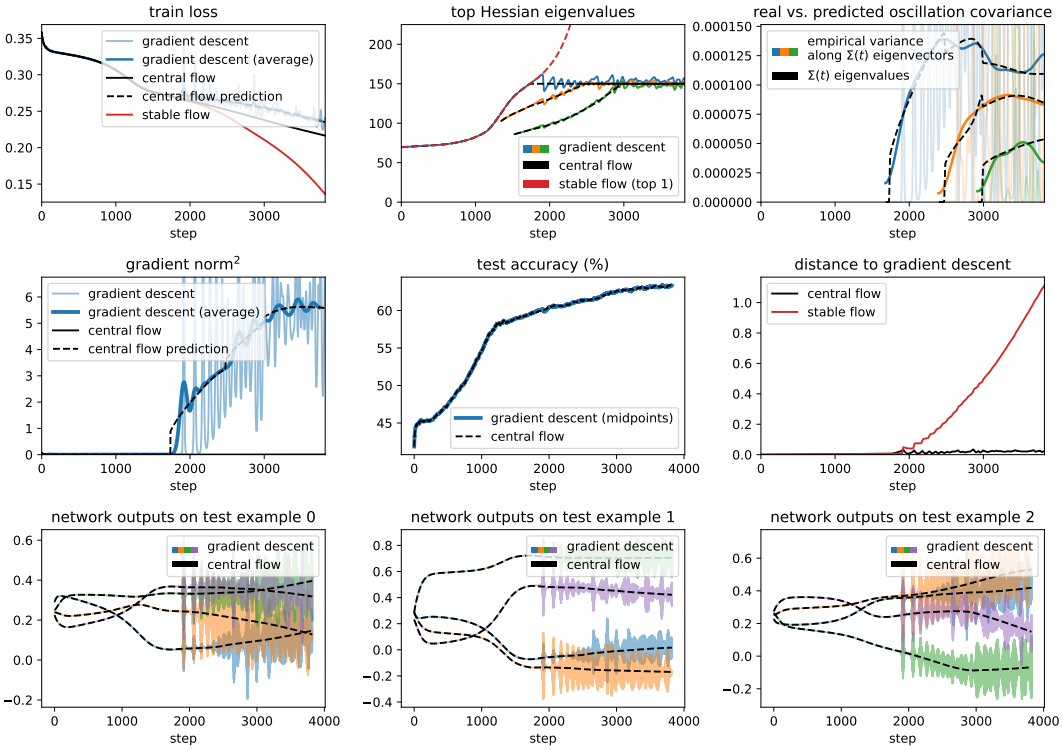


Figure 49.8: Gradient descent central flow for a ViT with MSE loss, $\eta = 0.013333$.

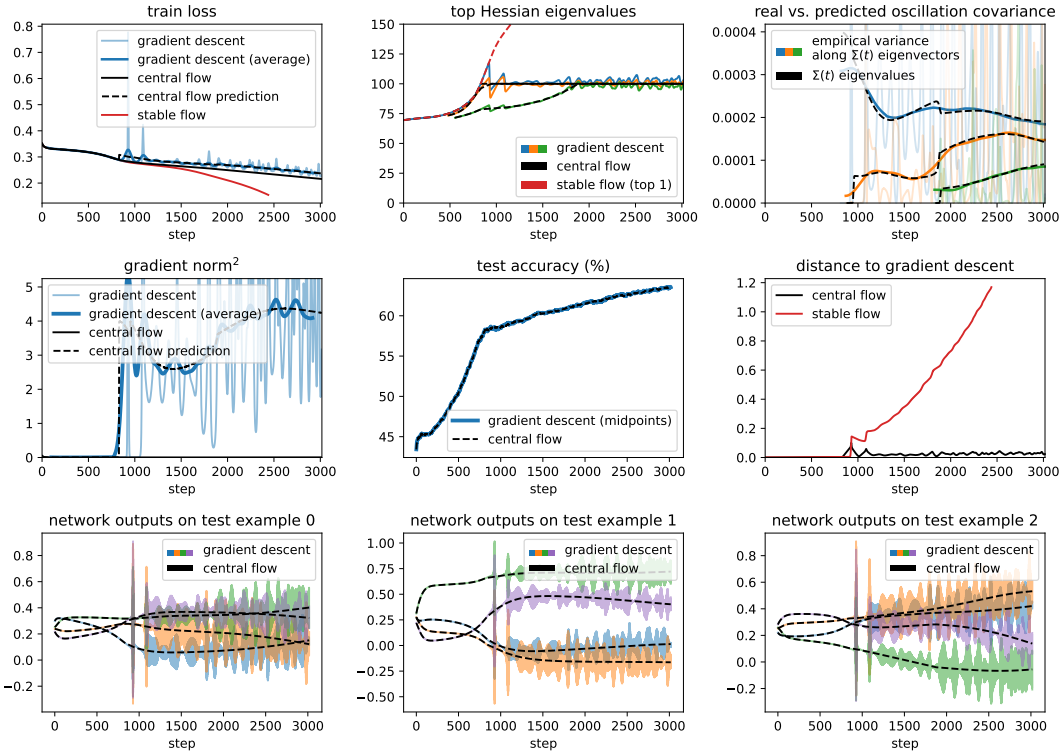


Figure 49.9: Gradient descent central flow for a ViT with MSE loss, $\eta = 0.02$.

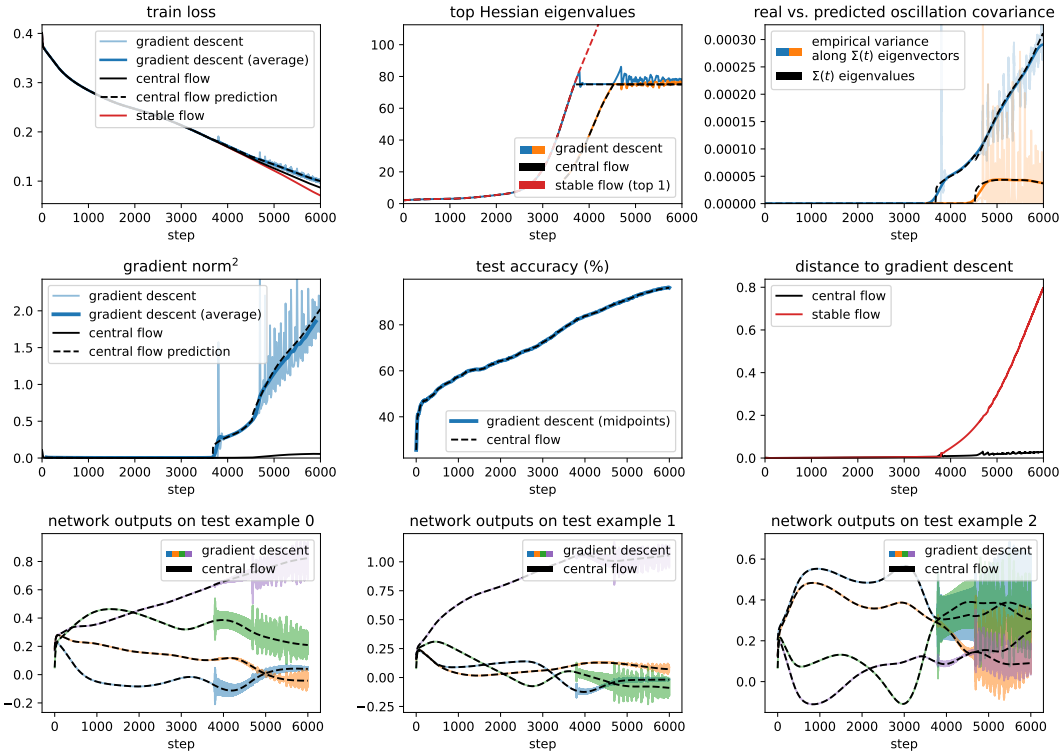


Figure 49.10: Gradient descent central flow for a LSTM with MSE loss, $\eta = 0.01333$.

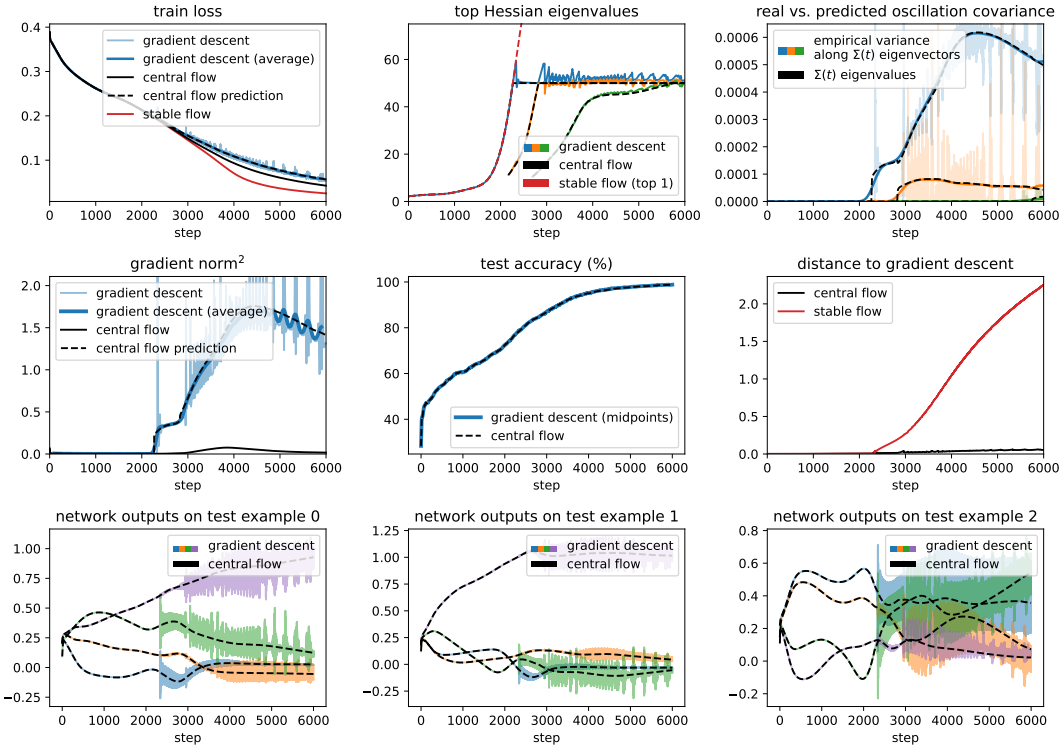


Figure 49.11: Gradient descent central flow for a LSTM with MSE loss, $\eta = 0.02$.

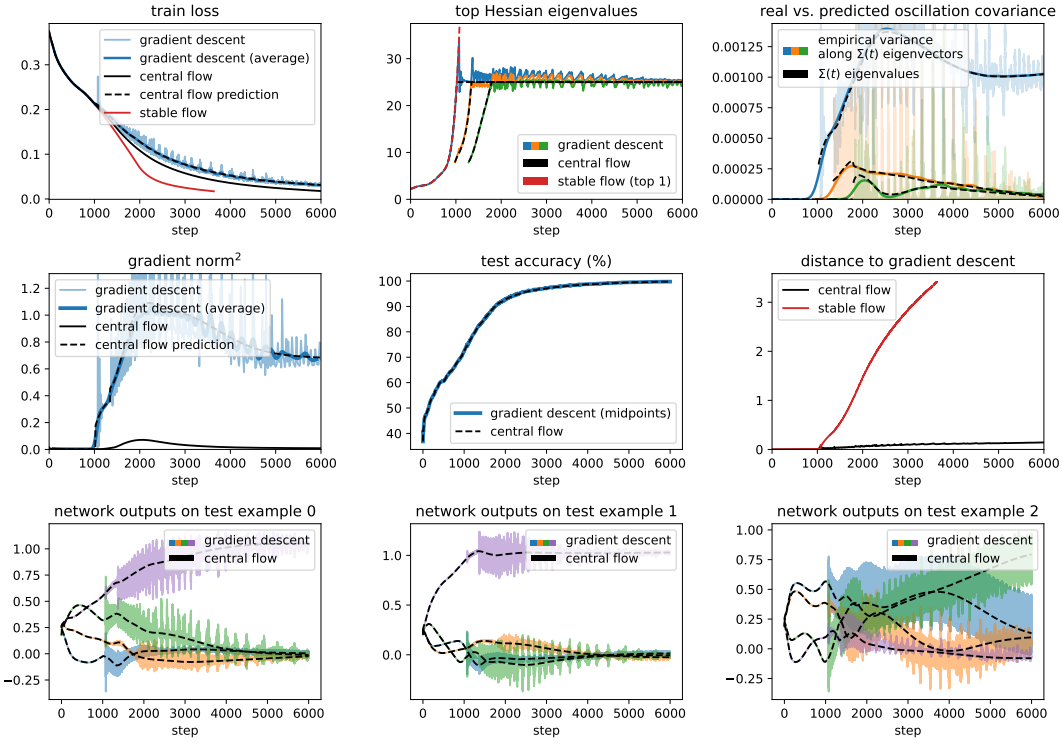


Figure 49.12: Gradient descent central flow for a LSTM with MSE loss, $\eta = 0.04$.

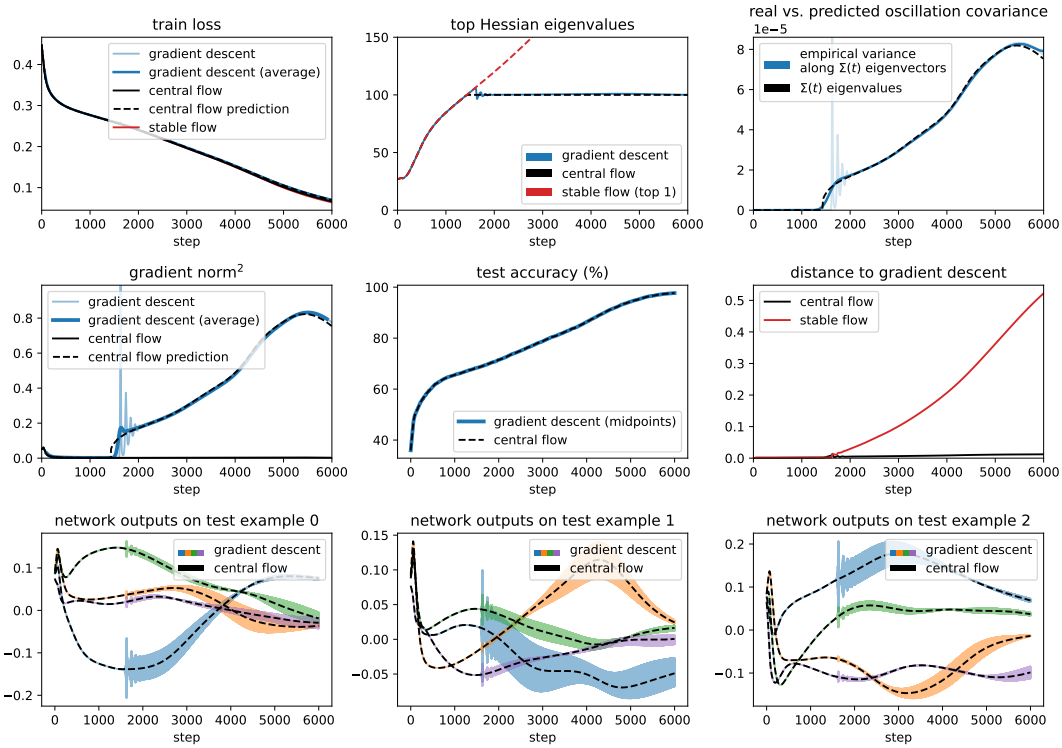


Figure 49.13: Gradient descent central flow for a Transformer with MSE loss, $\eta = 0.01$.

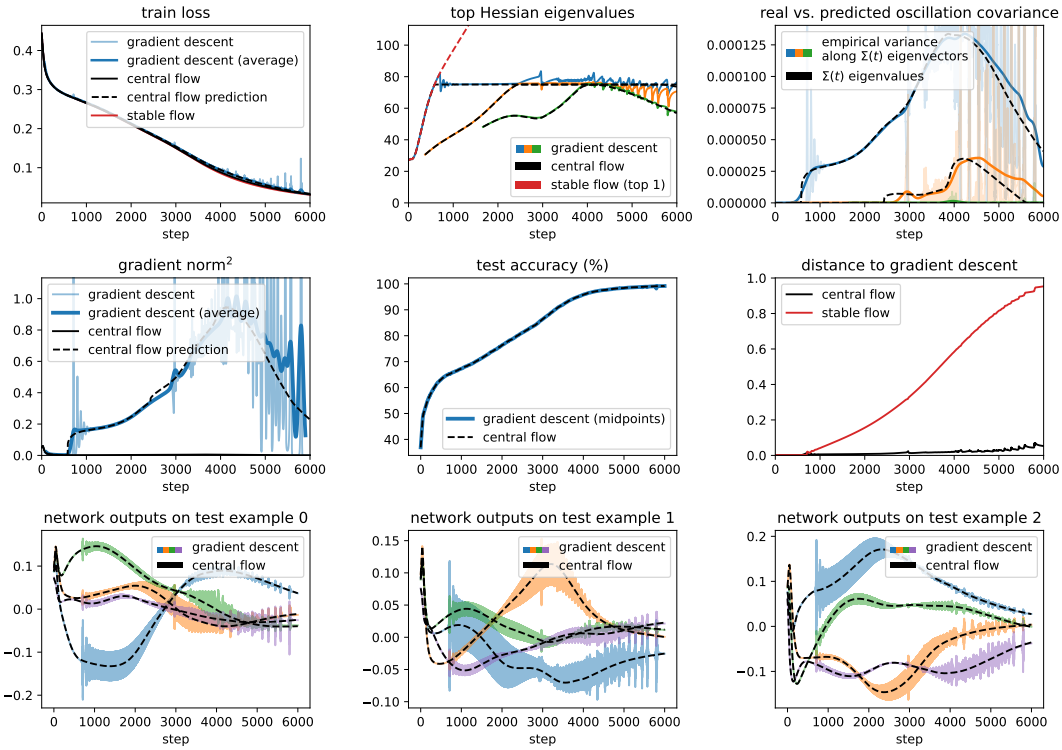


Figure 49.14: Gradient descent central flow for a Transformer with MSE loss, $\eta = 0.013333$.

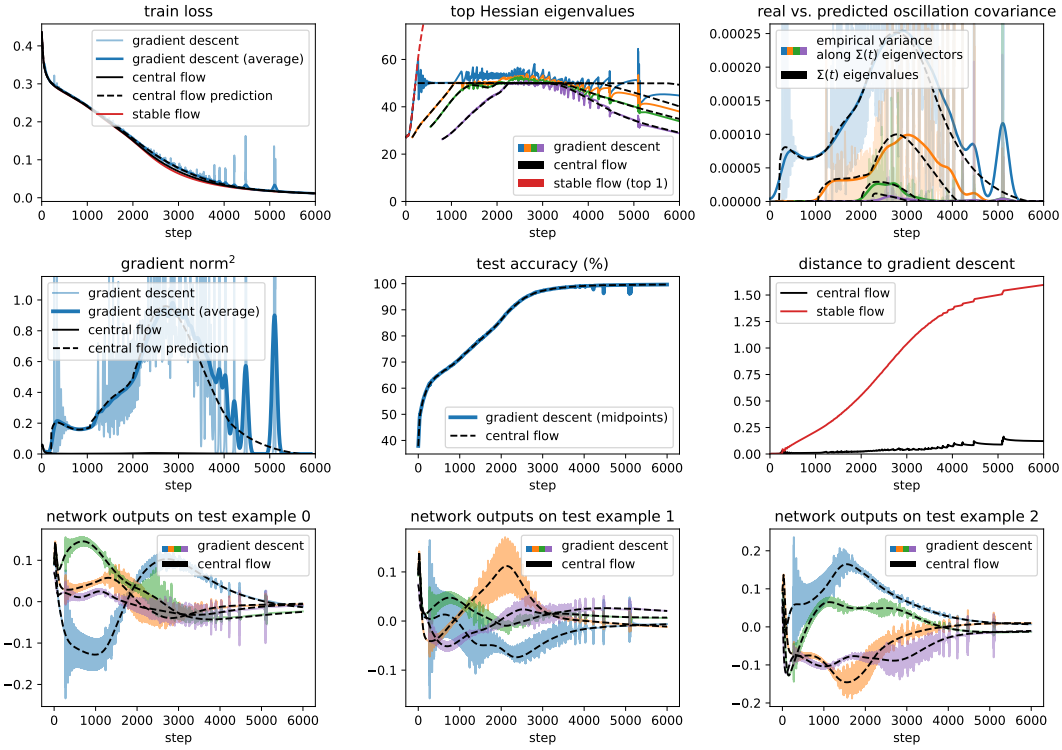


Figure 49.15: Gradient descent central flow for a Transformer with MSE loss, $\eta = 0.02$.

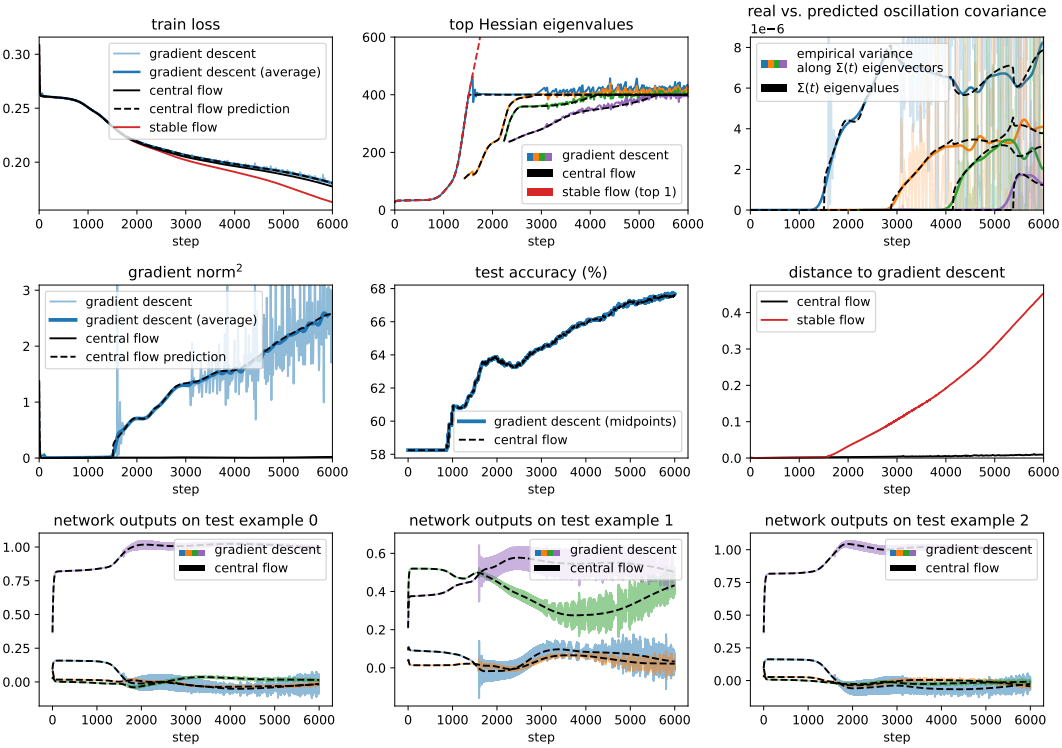


Figure 49.16: Gradient descent central flow for a Mamba with MSE loss, $\eta = 0.01$.

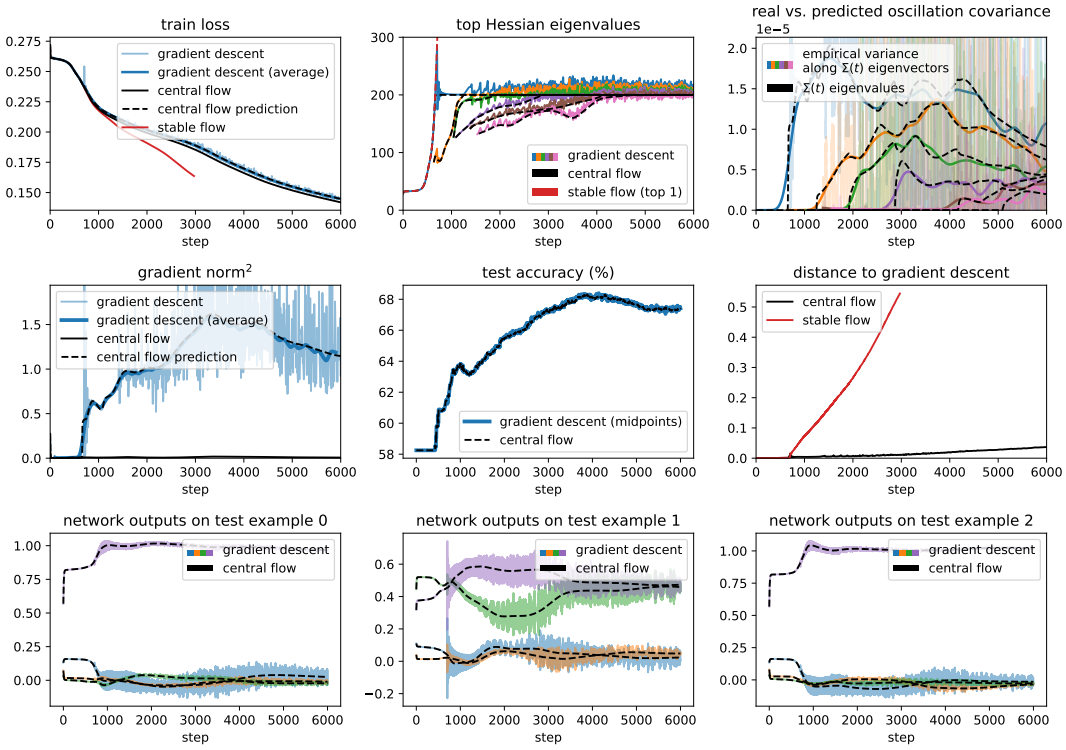


Figure 49.17: Gradient descent central flow for a Mamba with MSE loss, $\eta = 0.013333$.

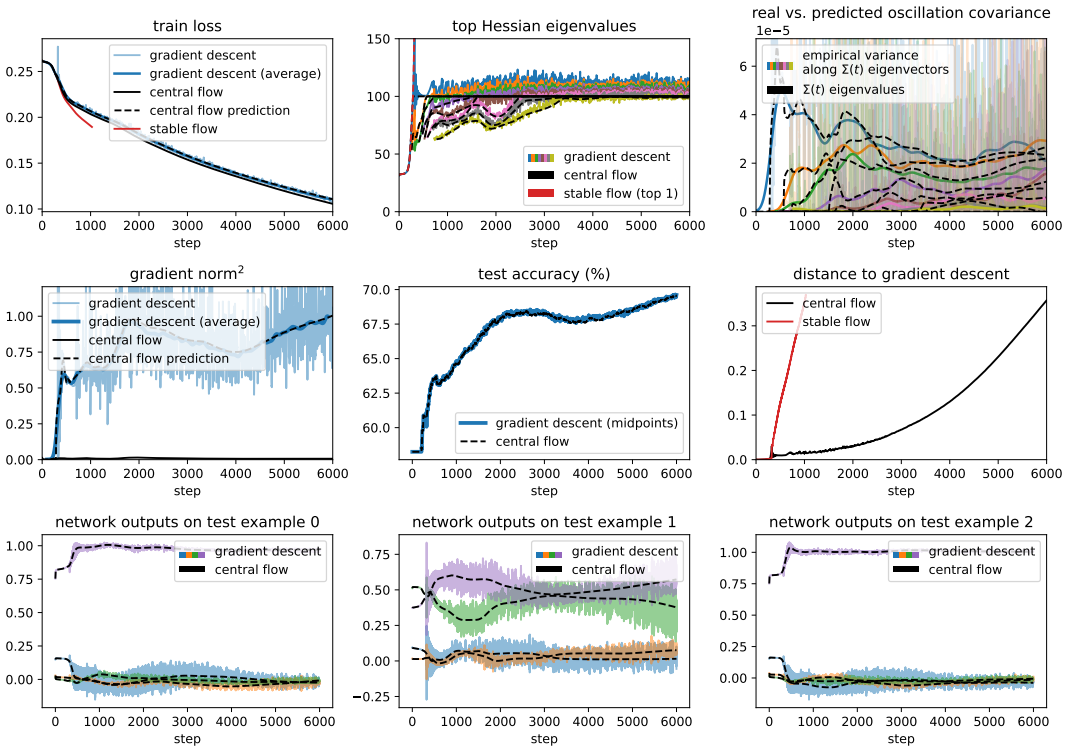


Figure 49.18: Gradient descent central flow for a Mamba with MSE loss, $\eta = 0.02$.

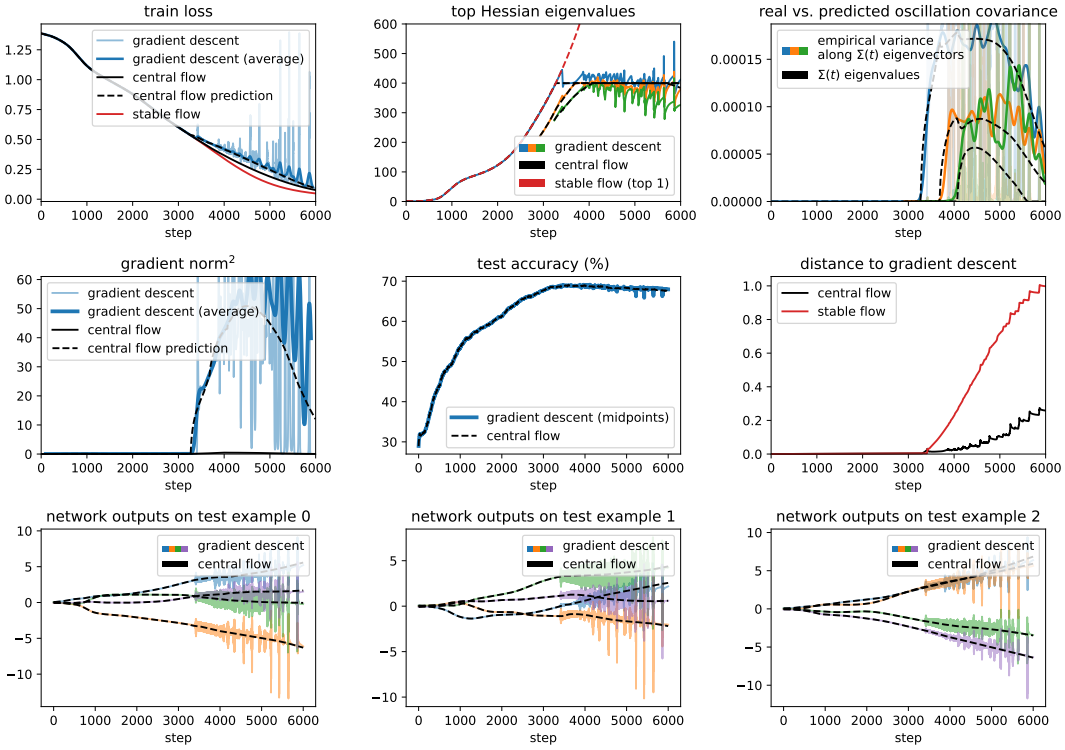


Figure 50.1: Gradient descent central flow for a CNN with CE loss, $\eta = 0.005$.

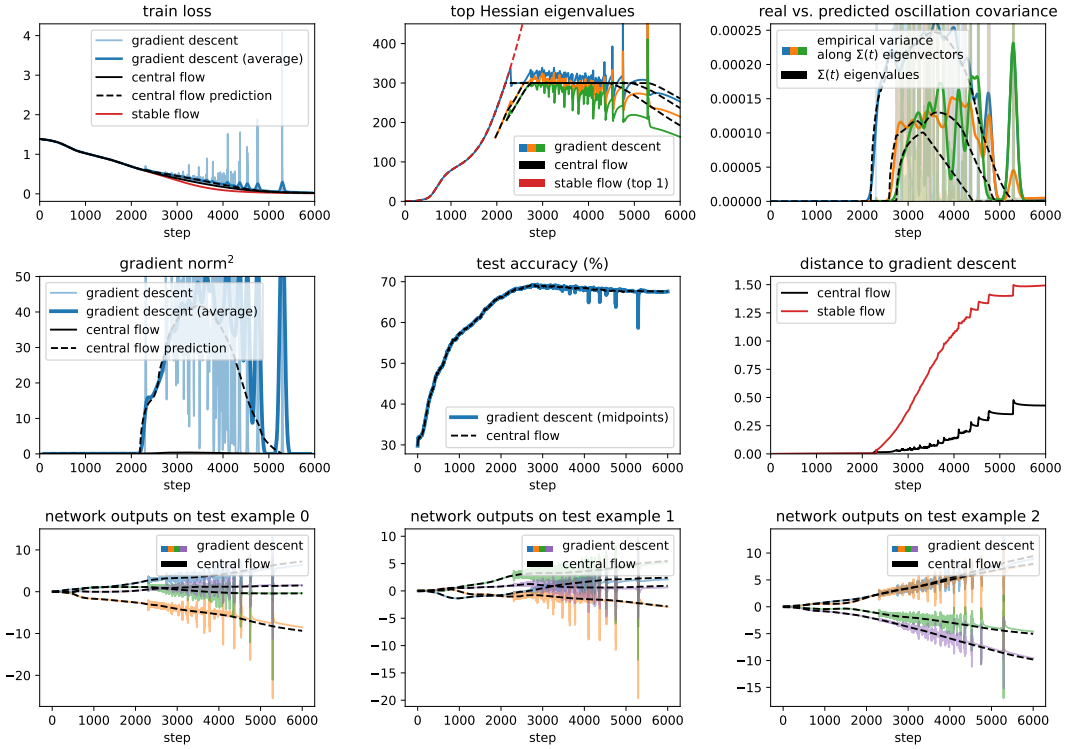


Figure 50.2: Gradient descent central flow for a CNN with CE loss, $\eta = 0.006666$.

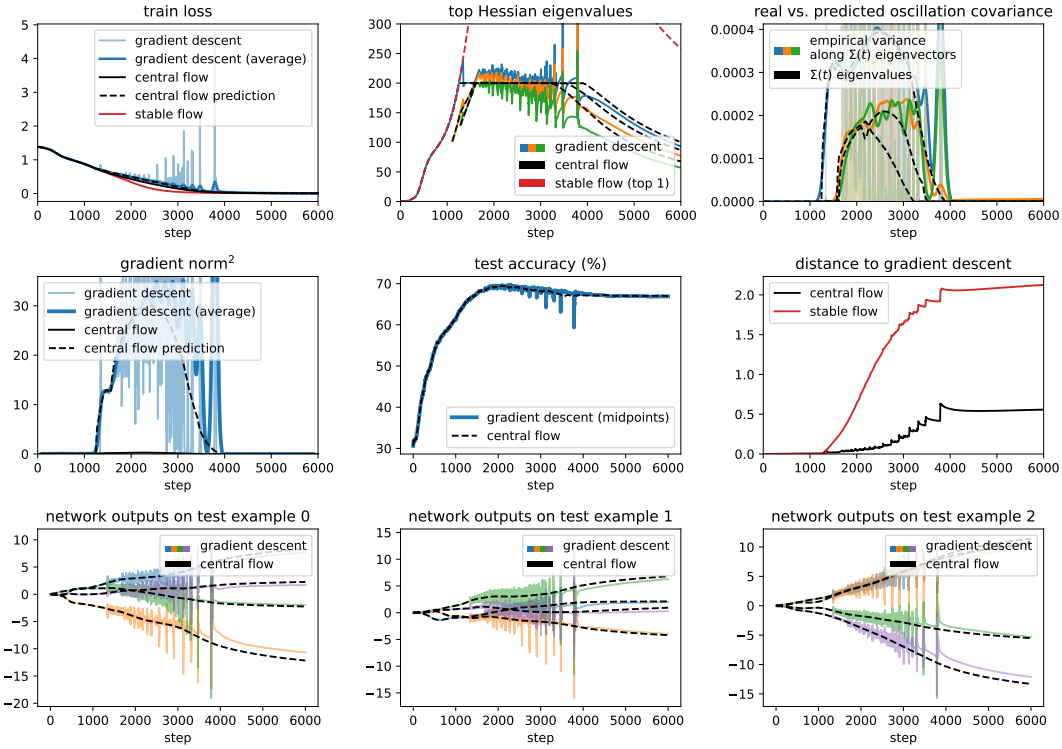


Figure 50.3: Gradient descent central flow for a CNN with CE loss, $\eta = 0.01$.

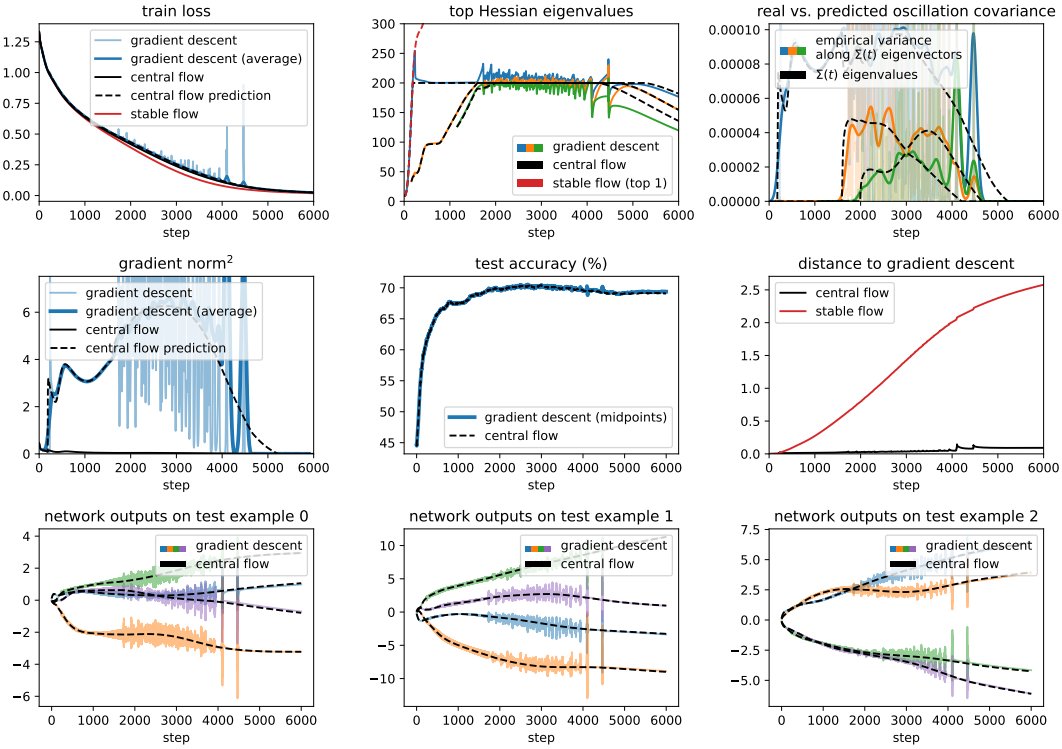


Figure 50.4: Gradient descent central flow for a ResNet with CE loss, $\eta = 0.01$.

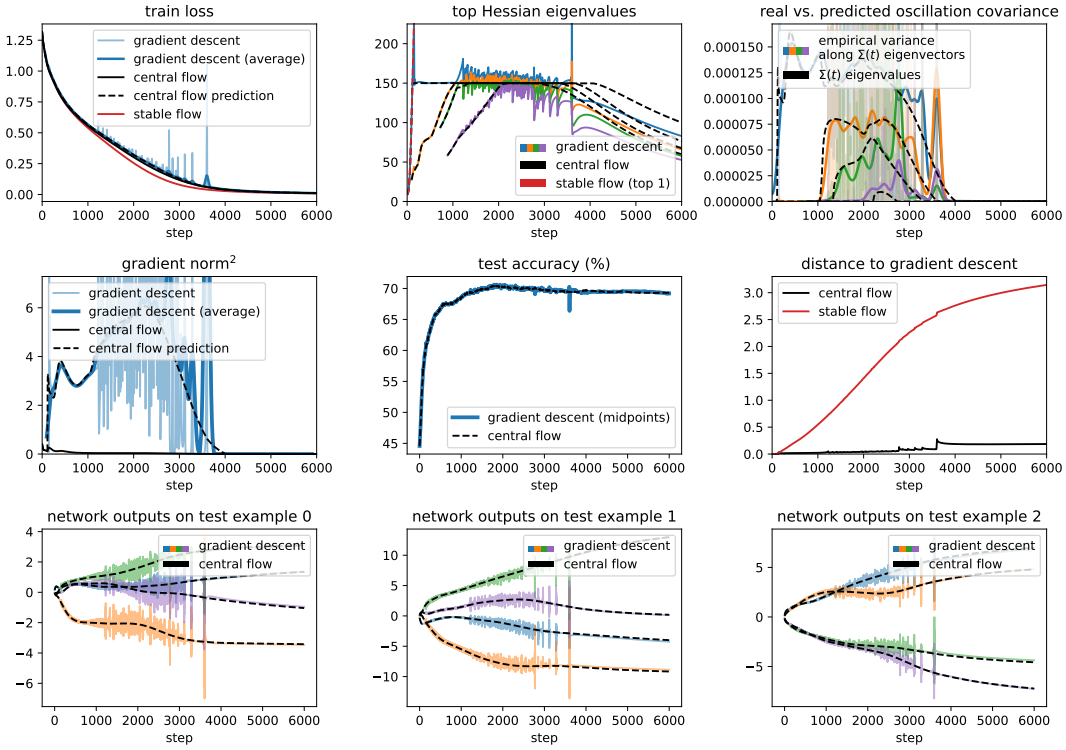


Figure 50.5: Gradient descent central flow for a ResNet with CE loss, $\eta = 0.013333$.

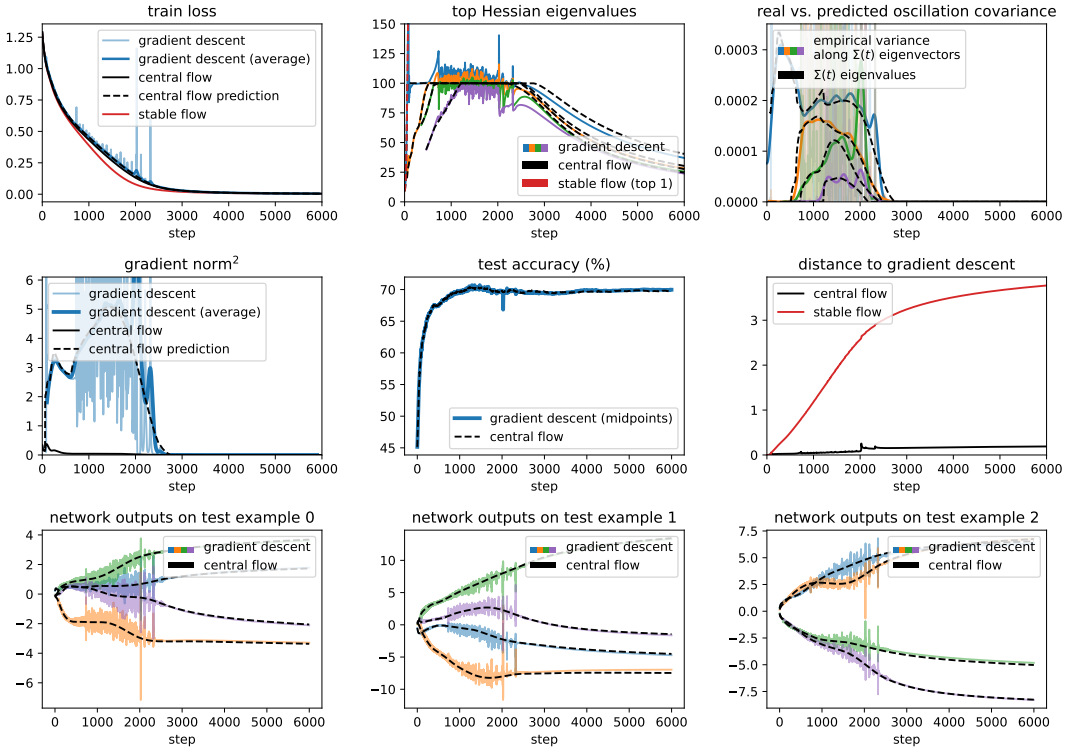


Figure 50.6: Gradient descent central flow for a ResNet with CE loss, $\eta = 0.02$.

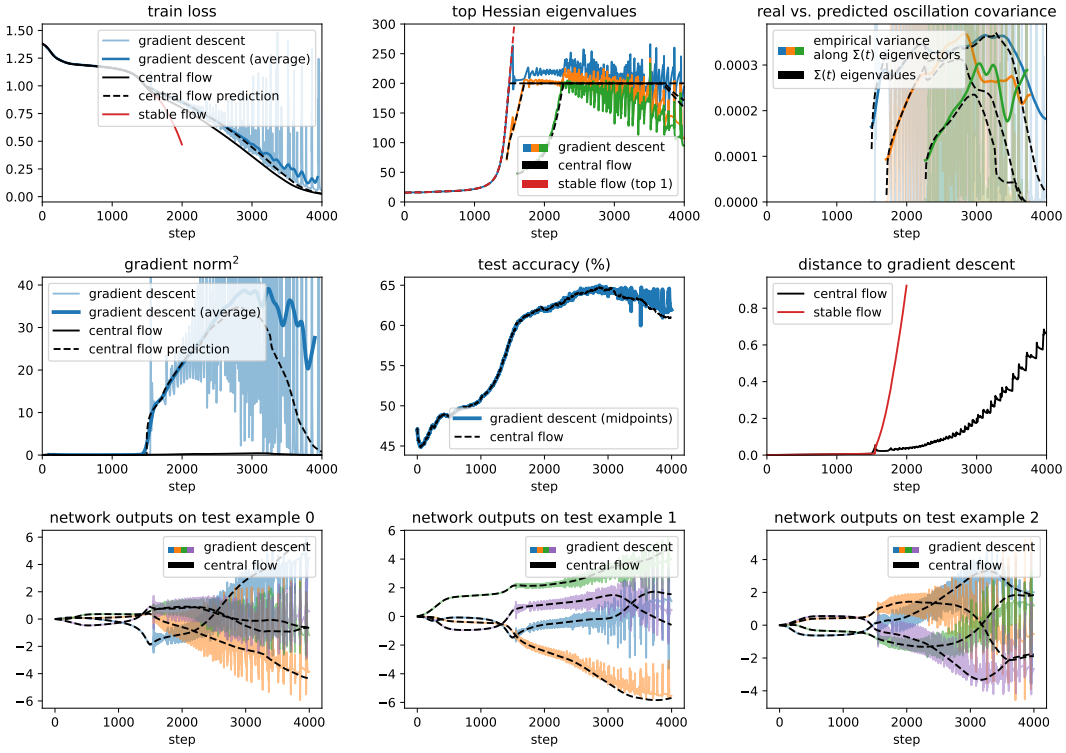


Figure 50.7: Gradient descent central flow for a ViT with CE loss, $\eta = 0.01$.

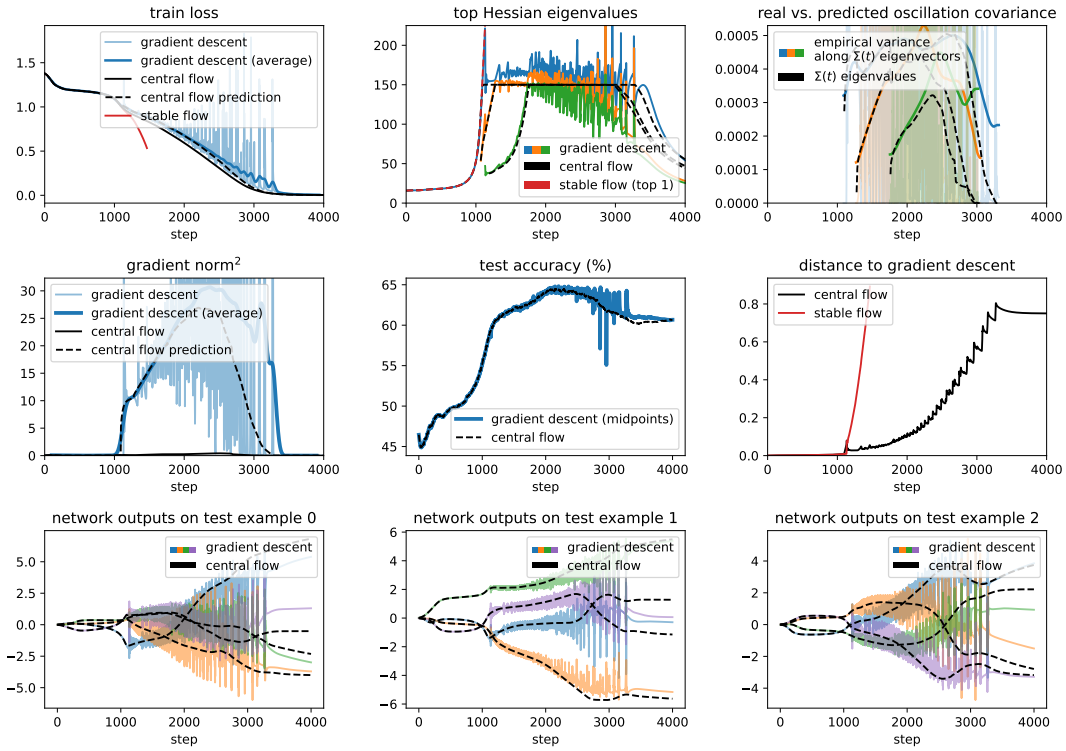


Figure 50.8: Gradient descent central flow for a ViT with CE loss, $\eta = 0.013333$.

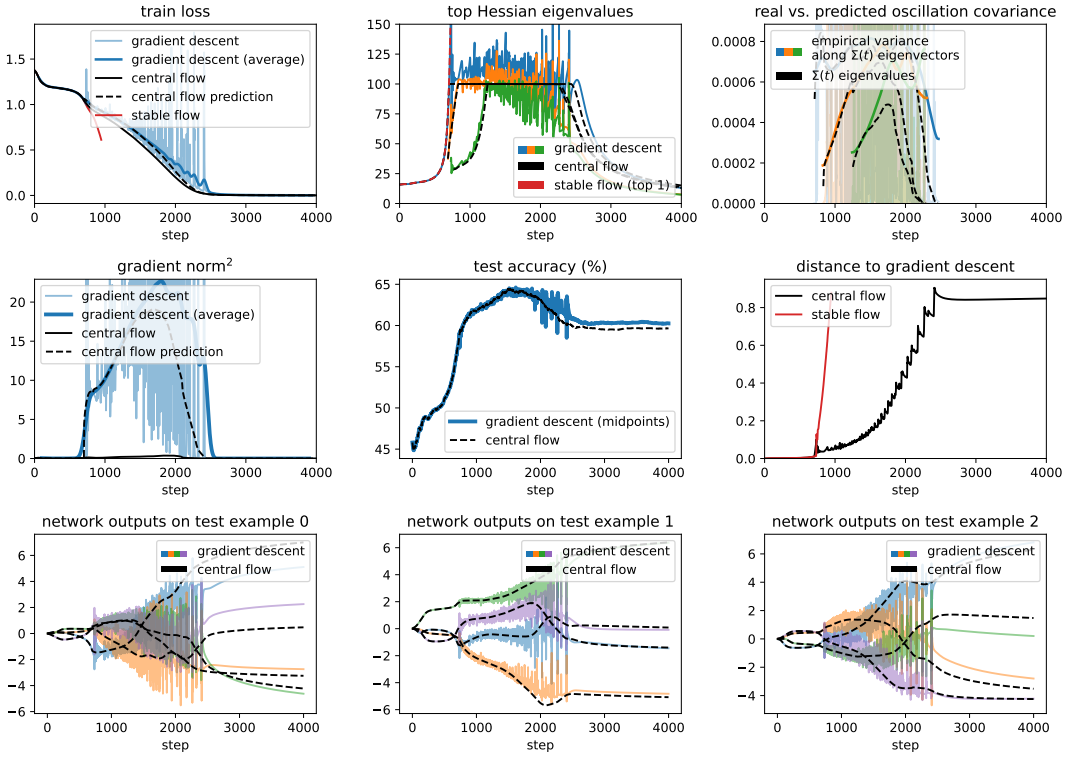


Figure 50.9: Gradient descent central flow for a ViT with CE loss, $\eta = 0.02$.

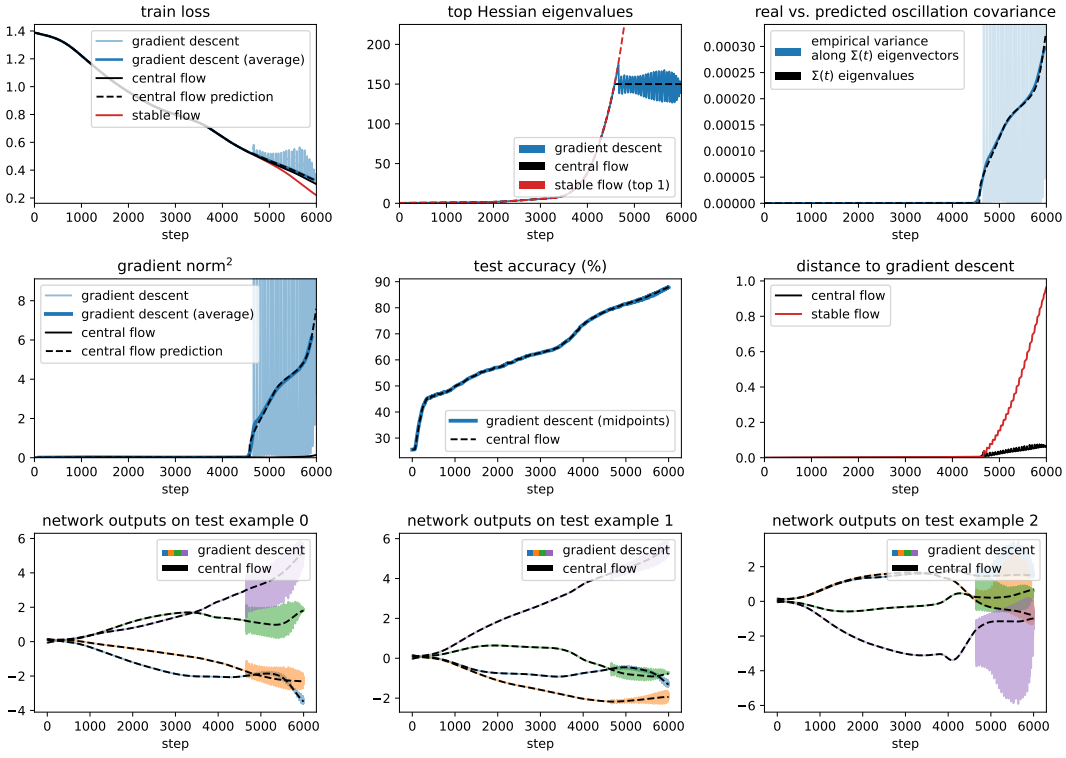


Figure 50.10: Gradient descent central flow for a LSTM with CE loss, $\eta = 0.01333$.

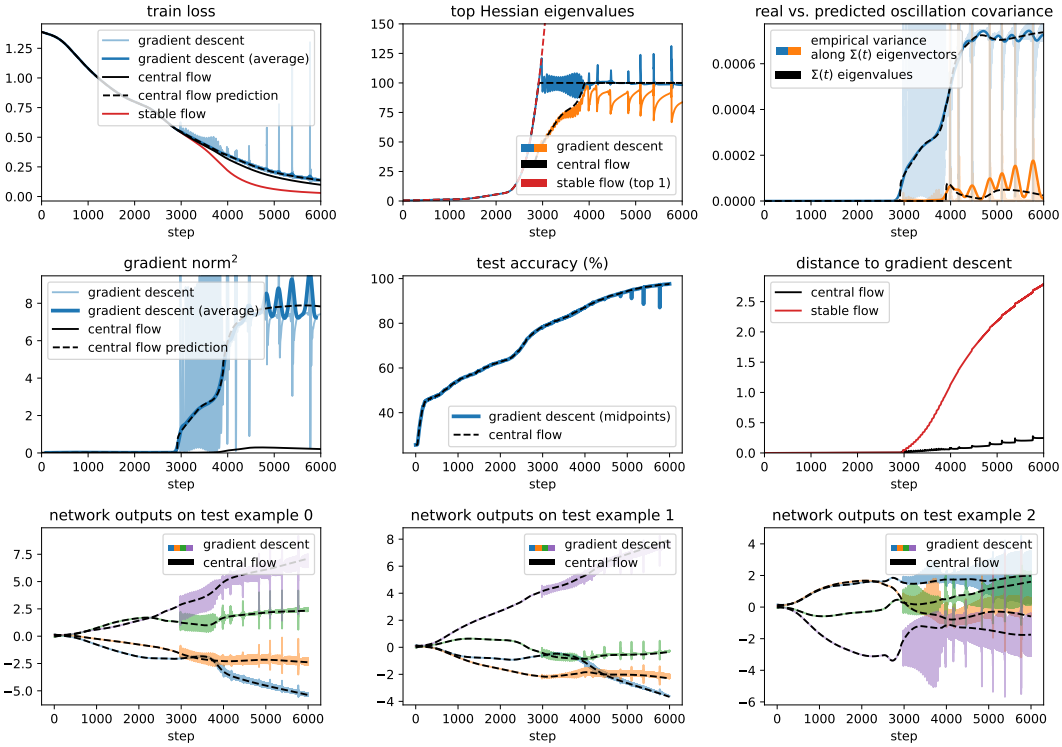


Figure 50.11: Gradient descent central flow for a LSTM with CE loss, $\eta = 0.02$.

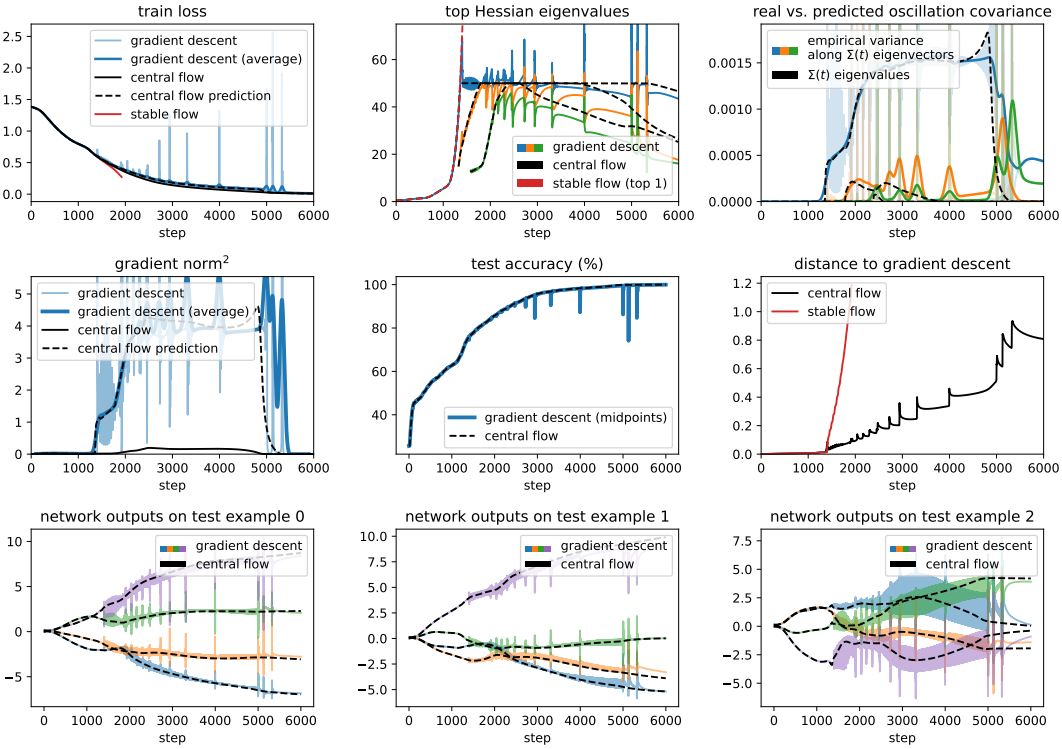


Figure 50.12: Gradient descent central flow for a LSTM with CE loss, $\eta = 0.04$.

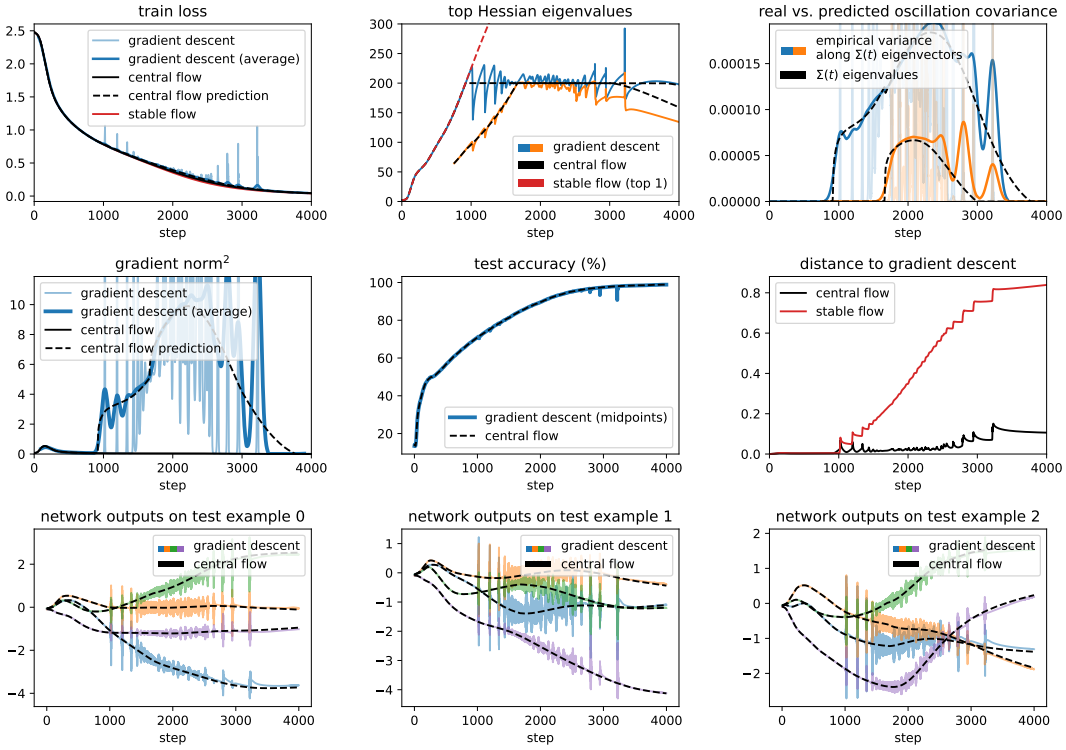


Figure 50.13: Gradient descent central flow for a Transformer with CE loss, $\eta = 0.01$.

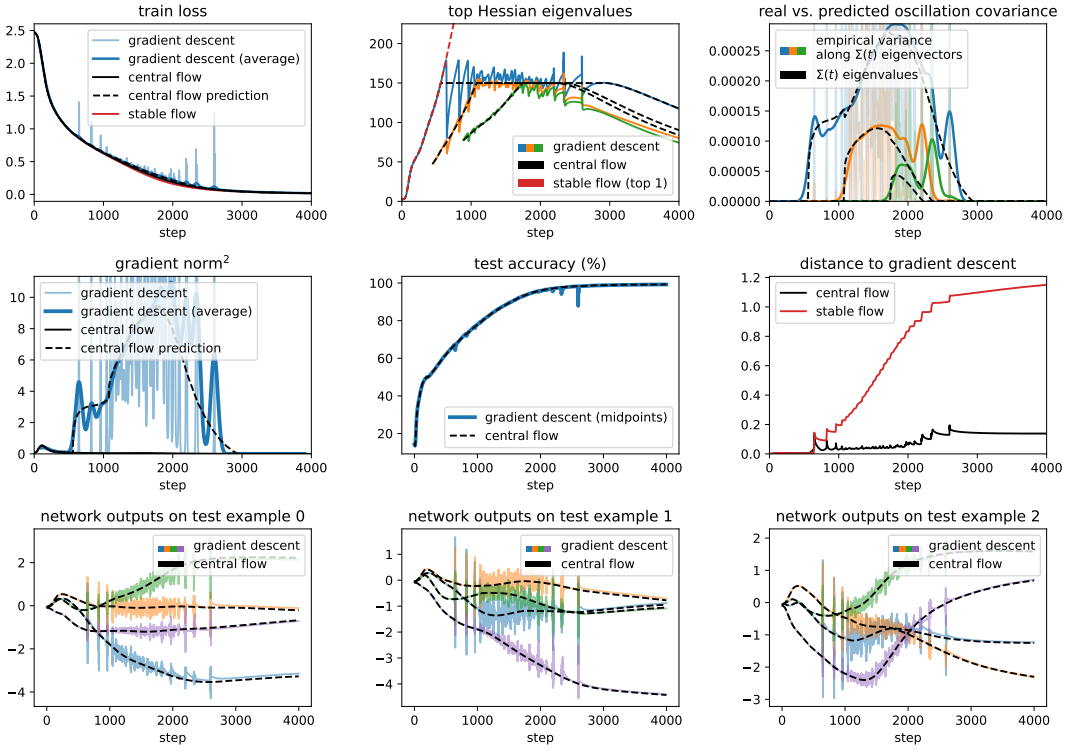


Figure 50.14: Gradient descent central flow for a Transformer with CE loss, $\eta = 0.013333$.

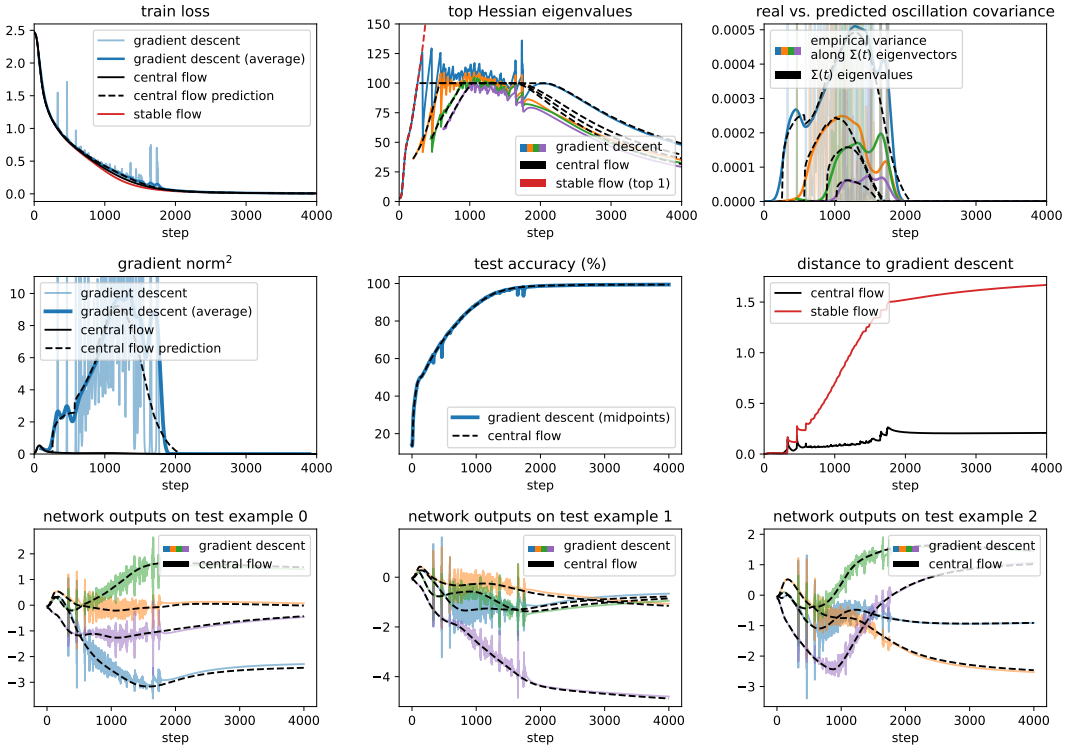


Figure 50.15: Gradient descent central flow for a Transformer with CE loss, $\eta = 0.02$.

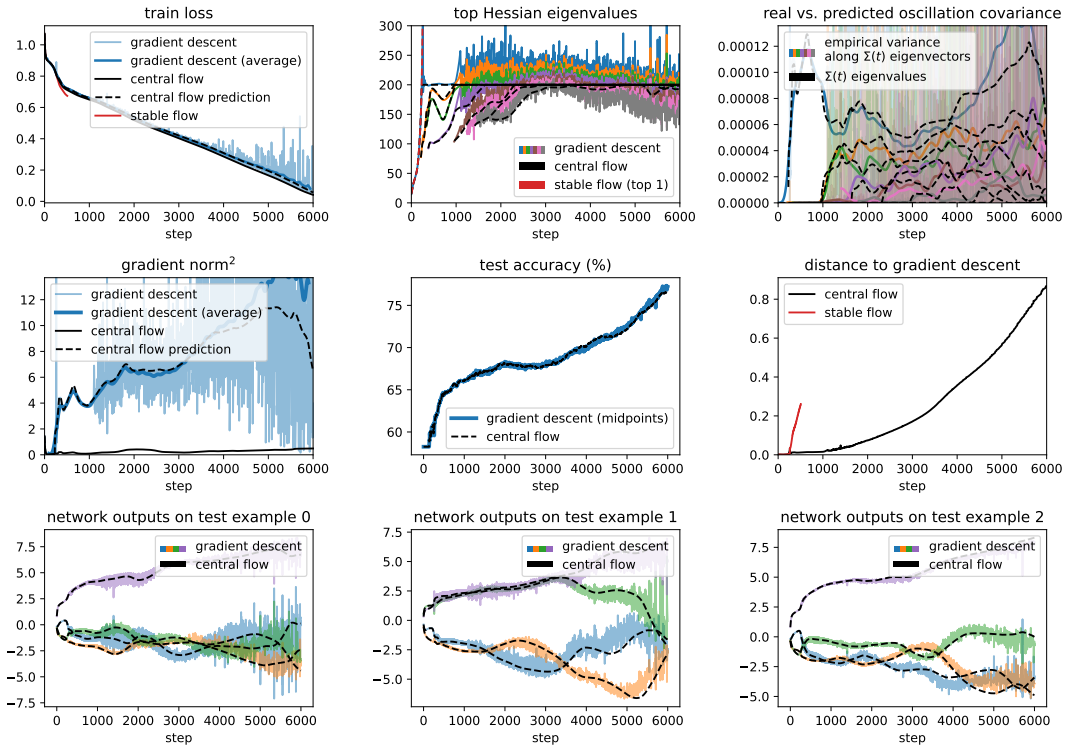


Figure 50.16: Gradient descent central flow for a Mamba with CE loss, $\eta = 0.01$.

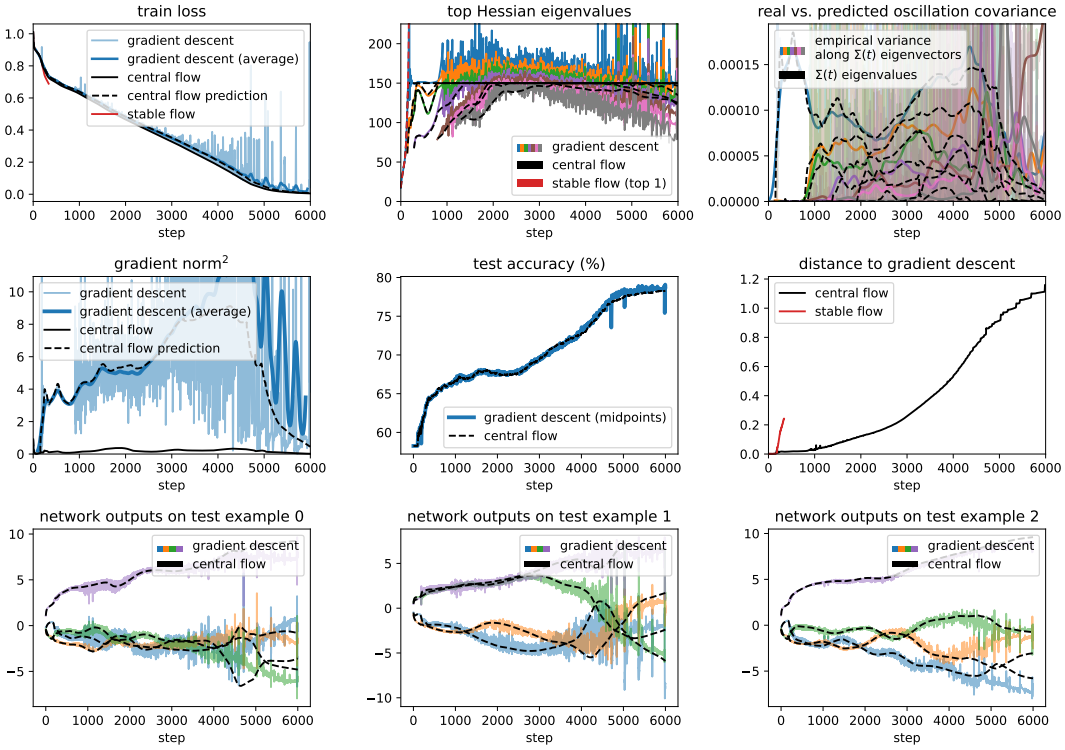


Figure 50.17: Gradient descent central flow for a Mamba with CE loss, $\eta = 0.013333$.

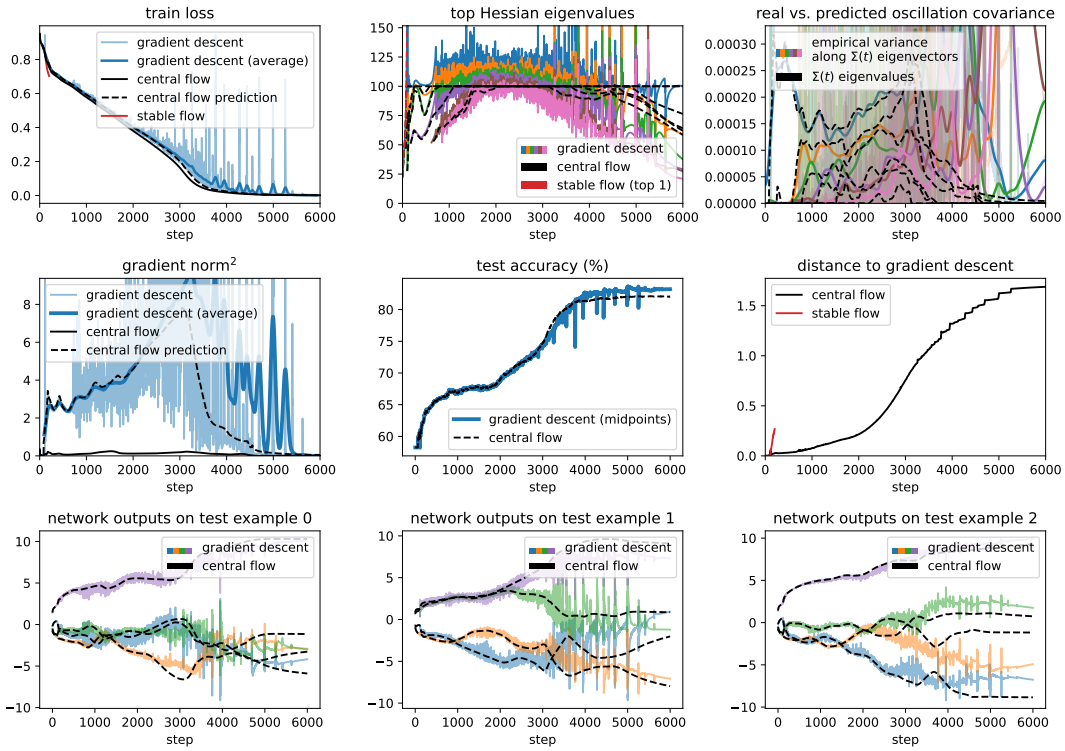


Figure 50.18: Gradient descent central flow for a Mamba with CE loss, $\eta = 0.02$.

E.2 Scalar RMSProp

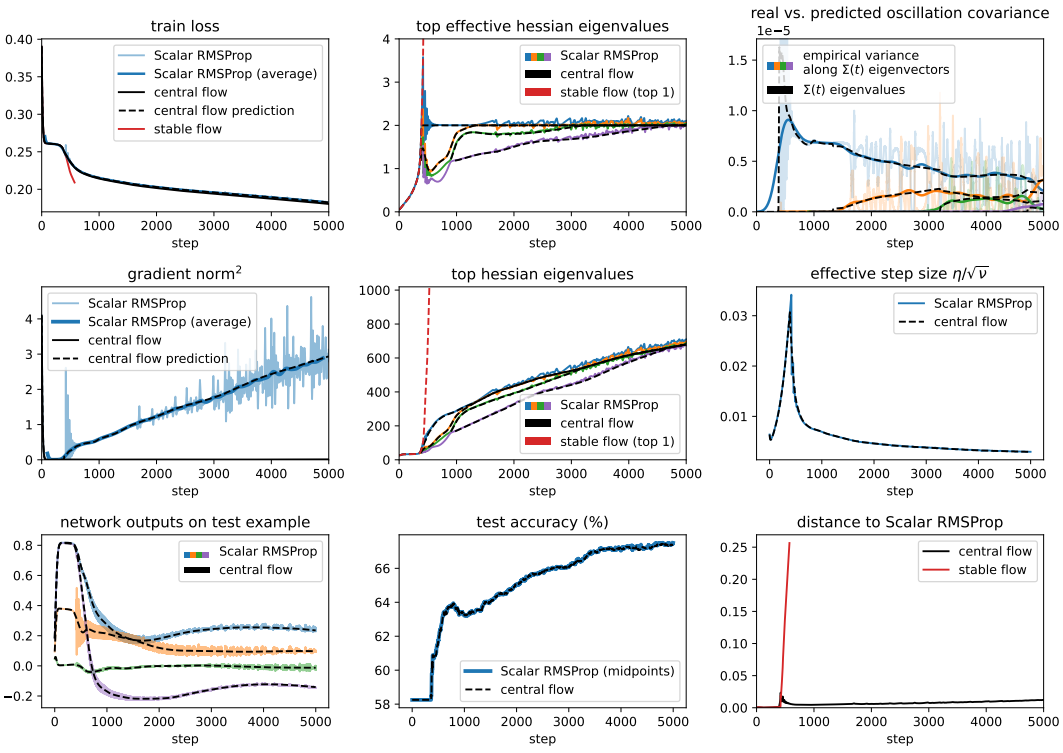


Figure 51: **Annotated example of a Scalar RMSProp experiment.** Using Scalar RMSProp with $\eta = 2/400$, $\beta_2 = 0.99$, and bias correction, we train a Mamba network on a synthetic sequence prediction task with MSE loss. The central flow (black) accurately models the long-term trajectory of Scalar RMSProp (blue), whereas the stable flow (red) takes a different path. As described in Appendix B.1, we terminate the stable flow once the effective sharpness gets too high.

Top left: See Figure 48 caption. The central flow's prediction for the time-averaged loss is given by eq. (110).
Top center: We plot the top several eigenvalues of the effective Hessian $\frac{\eta}{\sqrt{\nu}}H(w)$ under both Scalar RMSProp (colors) and its central flow (dashed black). Under Scalar RMSProp, these eigenvalues equilibrate around the critical threshold 2, whereas under the central flow they are fixed exactly at 2. We also plot the top eigenvalue under the "stable flow" baseline (red), and this increases far above 2.
Top right: See Figure 48 caption. This plot is validating eq. (112).
Middle left: See Figure 48 caption. The central flow's prediction for the time-average is given by eq. (111).
Middle center: We plot the top several eigenvalues of the "raw" Hessian $H(w)$, under both Scalar RMSProp (colors) and the central flow (dashed black). These evolve throughout training, even as top eigenvalues of the *effective* Hessian are equilibrating at the critical threshold (top center). In red, we plot the top Hessian eigenvalue under the stable flow.
Middle right: We plot the effective step size $\eta/\sqrt{\nu}$ under both Scalar RMSProp (blue) and the central flow (dashed black). This effective step size oscillates under Scalar RMSProp, but varies smoothly under the central flow.
Bottom left: We show the network's final-layer predictions on an arbitrary example. Under Scalar RMSProp (colors) these predictions oscillate due to the oscillations in weight space. Under the central flow (dashed black), the predictions evolve smoothly while following the same macroscopic path.
Bottom center: See Figure 48 caption.
Bottom right: See Figure 48 caption.

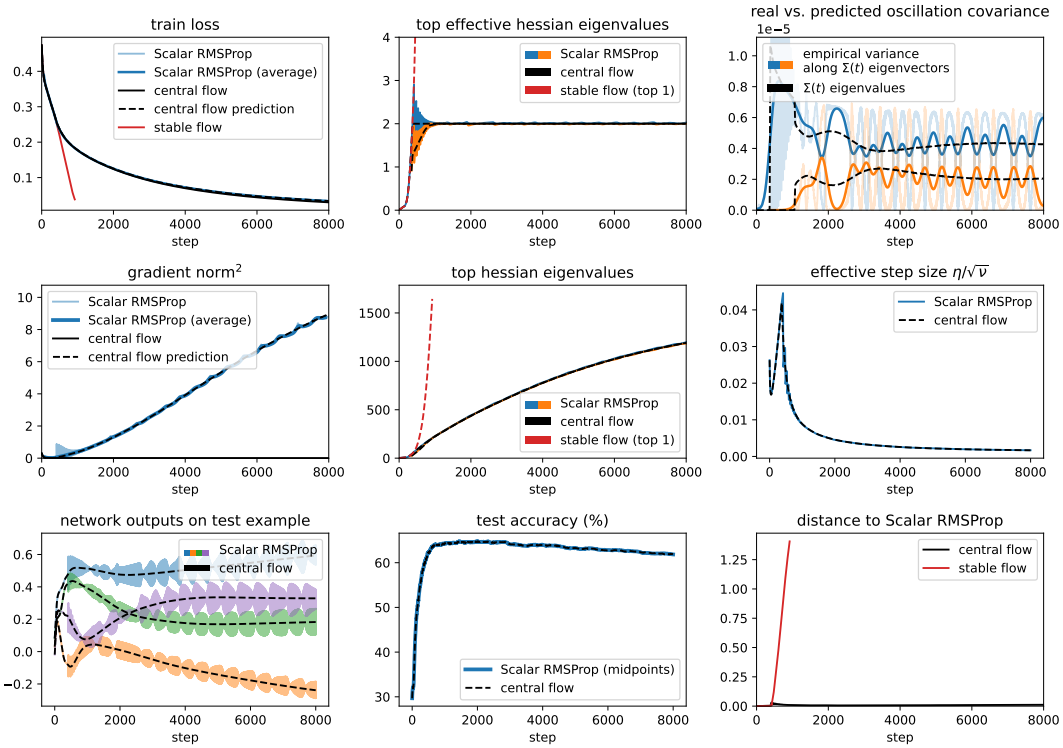


Figure 52.1: Scalar RMSProp central flow for a CNN with MSE loss, $\eta = 0.003$, $\beta_2 = 0.99$, and bias correction.

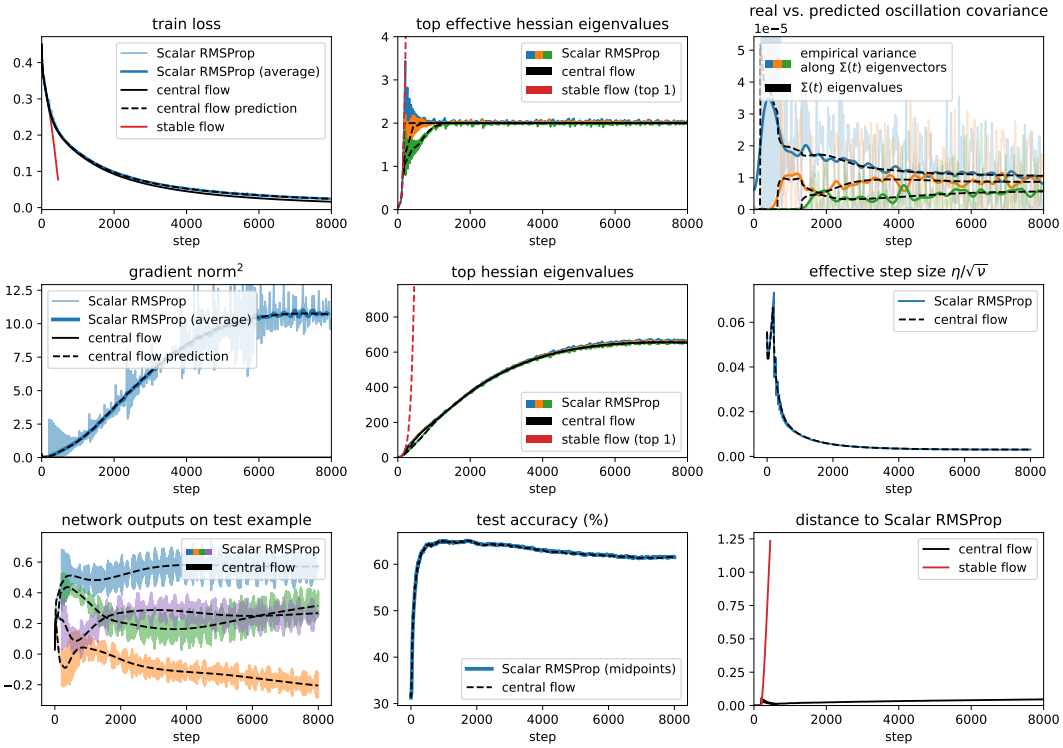


Figure 52.2: Scalar RMSProp central flow for a CNN with MSE loss, $\eta = 0.006$, $\beta_2 = 0.99$, and bias correction.

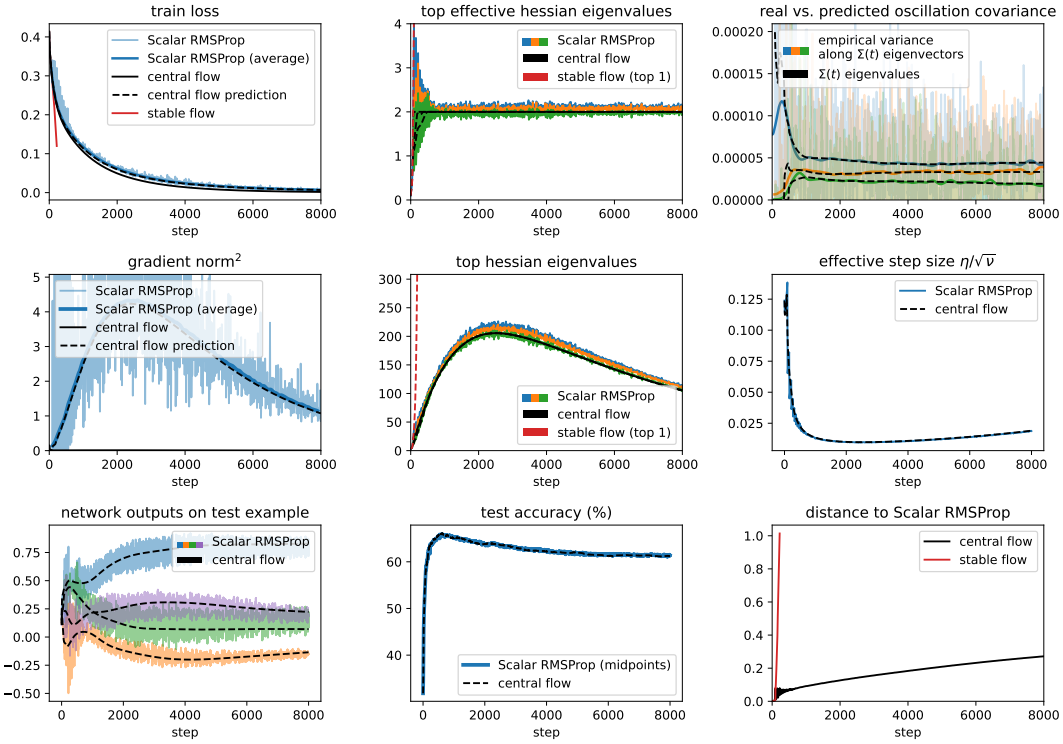


Figure 52.3: Scalar RMSProp central flow for a CNN with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

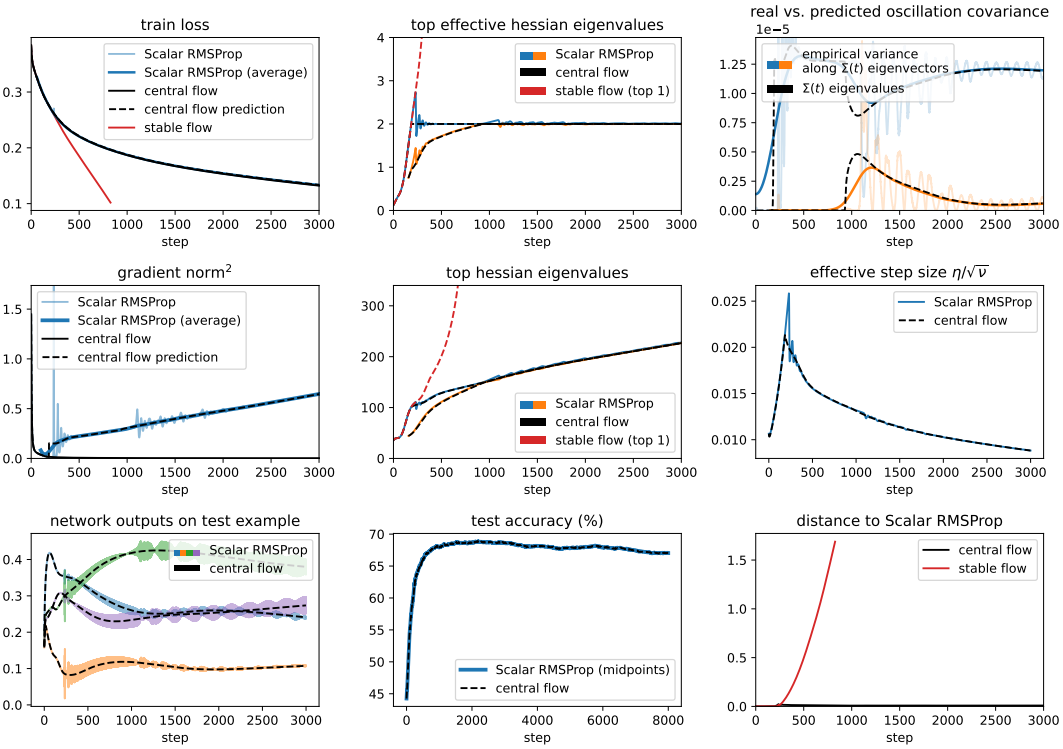


Figure 52.4: Scalar RMSProp central flow for a ResNet with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

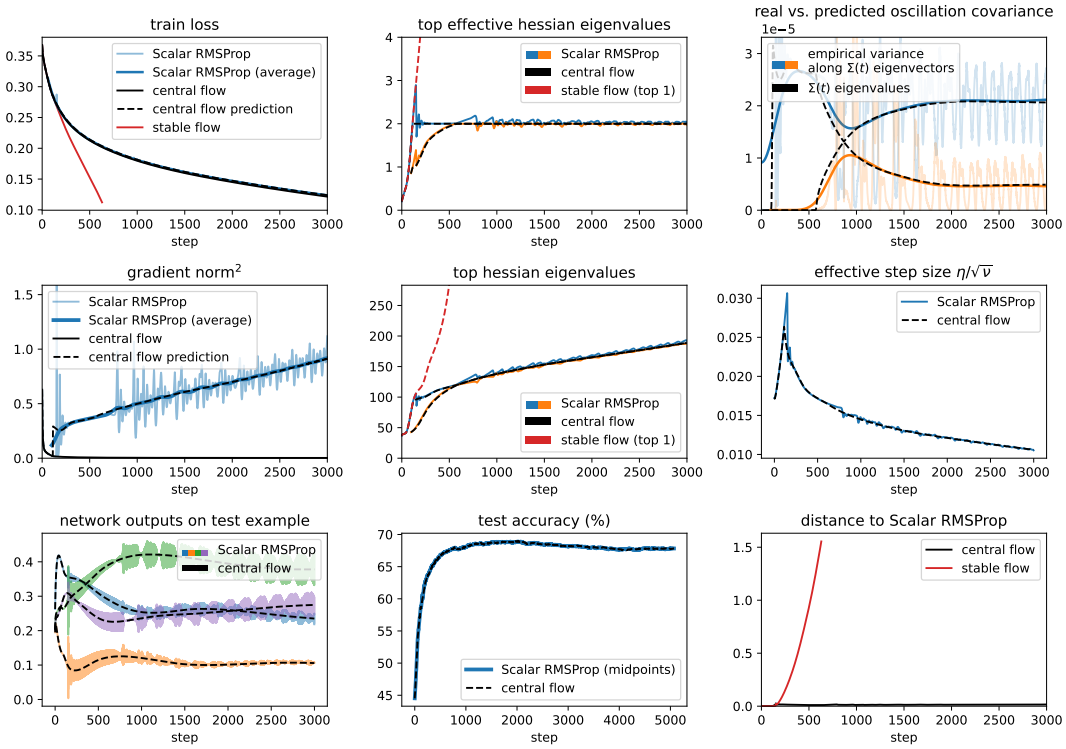


Figure 52.5: Scalar RMSProp central flow for a ResNet with MSE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

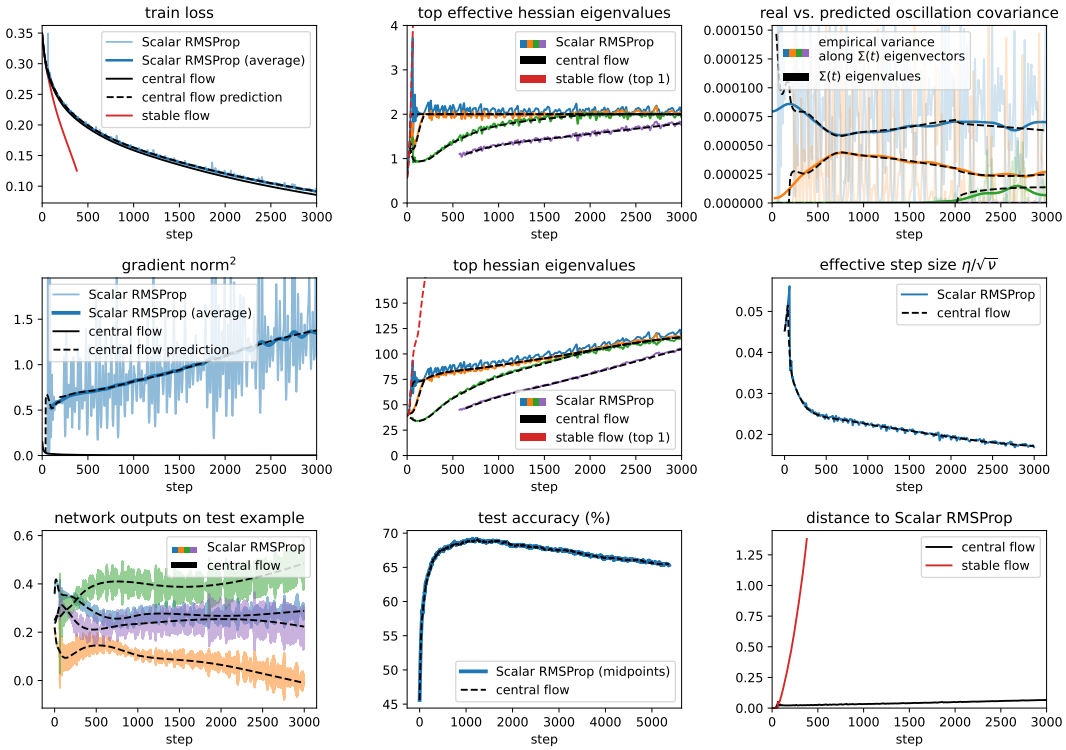


Figure 52.6: Scalar RMSProp central flow for a ResNet with MSE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

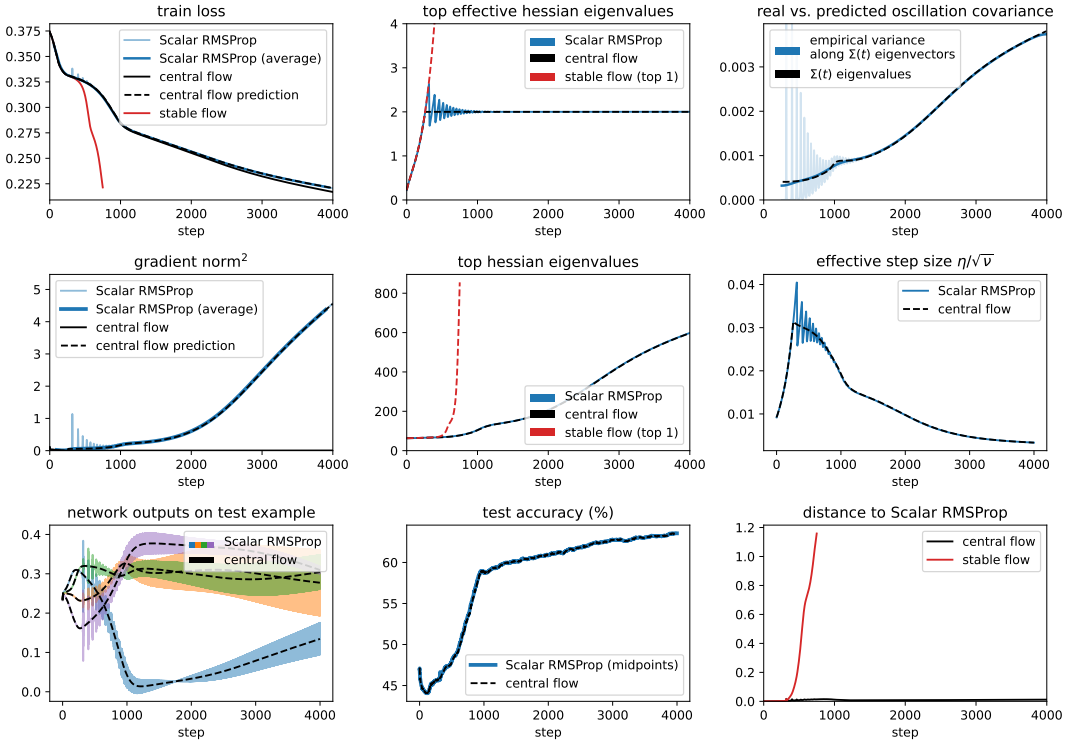


Figure 52.7: Scalar RMSProp central flow for a ViT with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

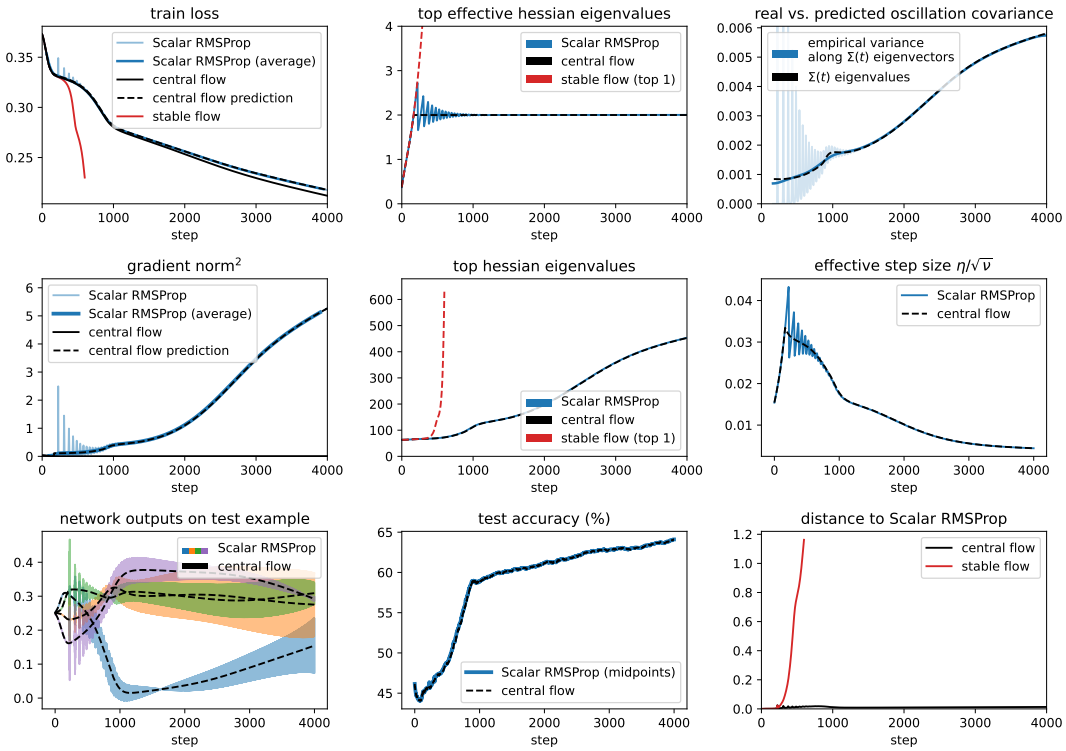


Figure 52.8: Scalar RMSProp central flow for a ViT with MSE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

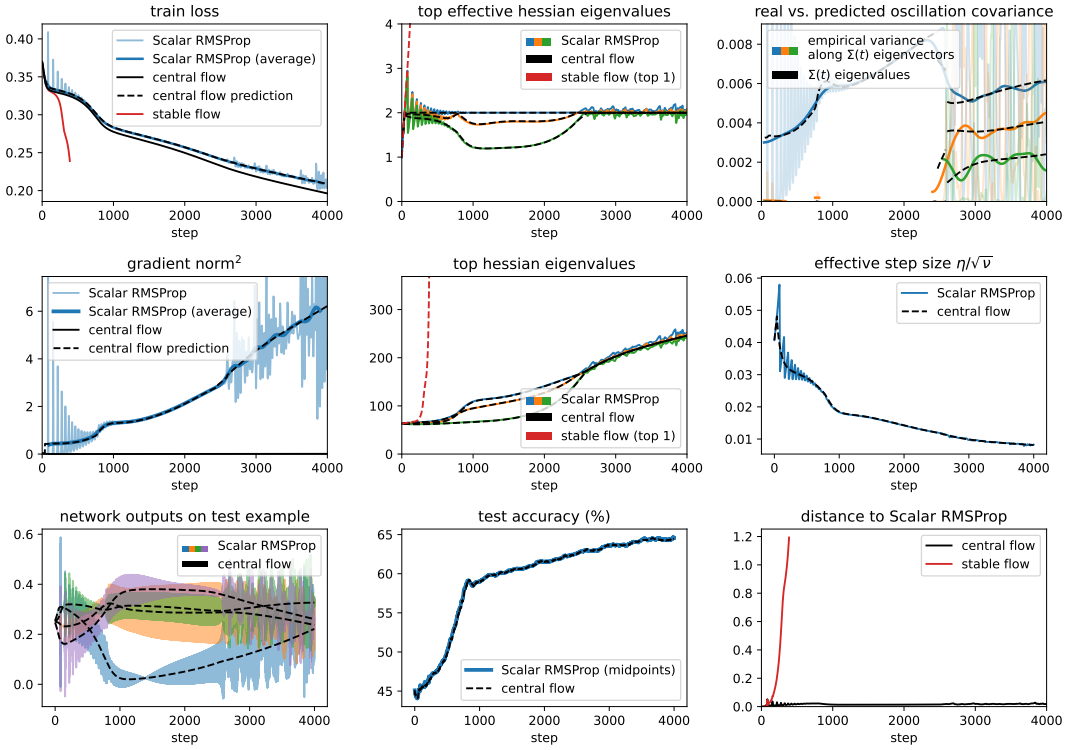


Figure 52.9: Scalar RMSProp central flow for a ViT with MSE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

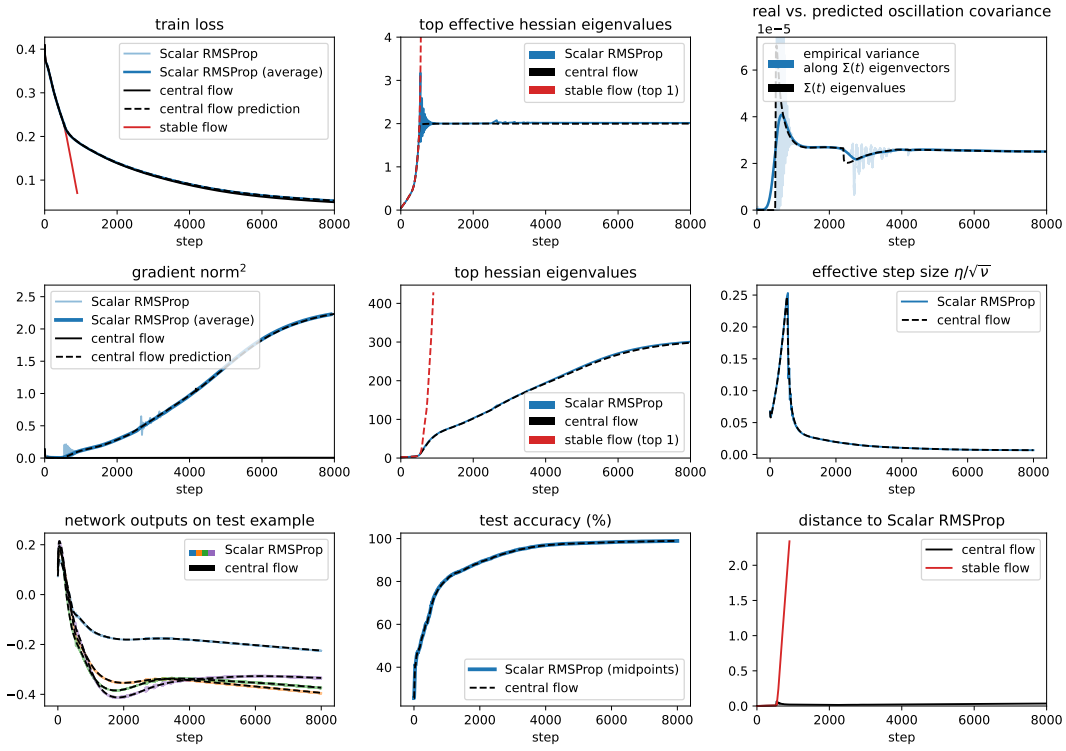


Figure 52.10: Scalar RMSProp central flow for a LSTM with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

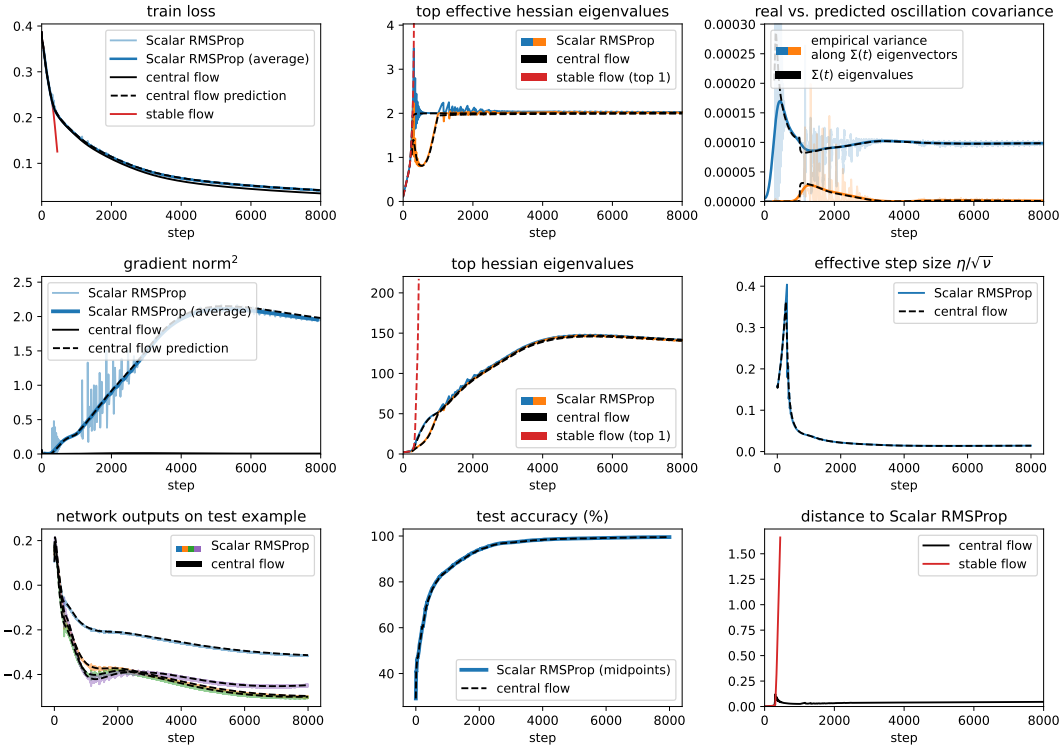


Figure 52.11: Scalar RMSProp central flow for a LSTM with MSE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

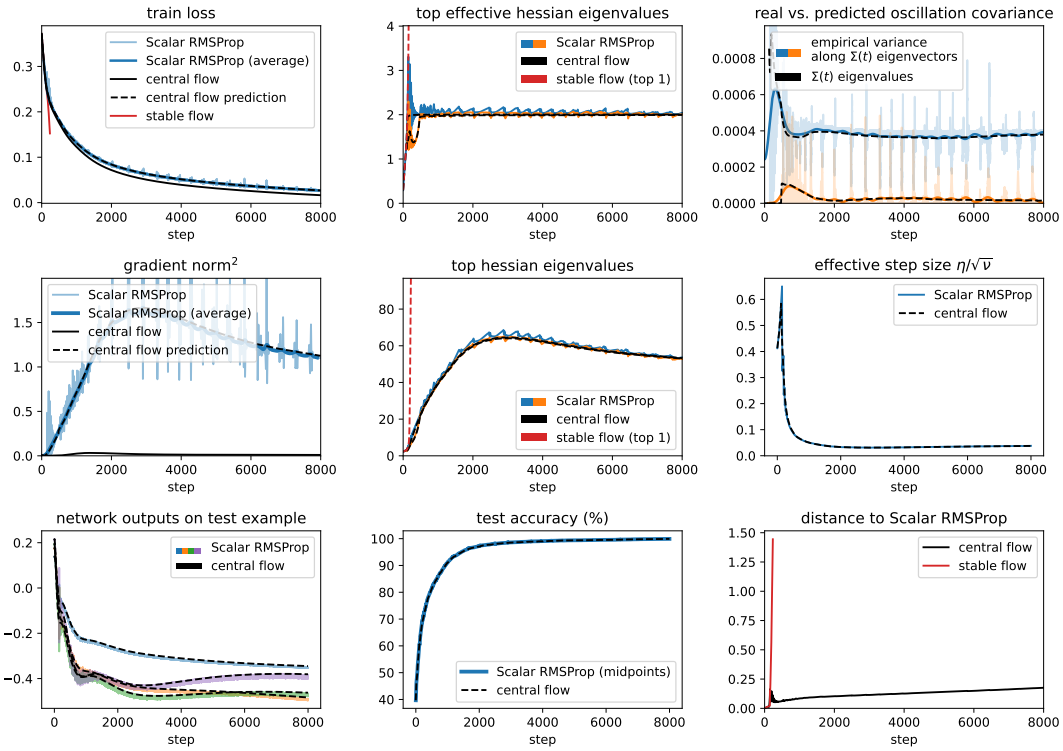


Figure 52.12: Scalar RMSProp central flow for a LSTM with MSE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

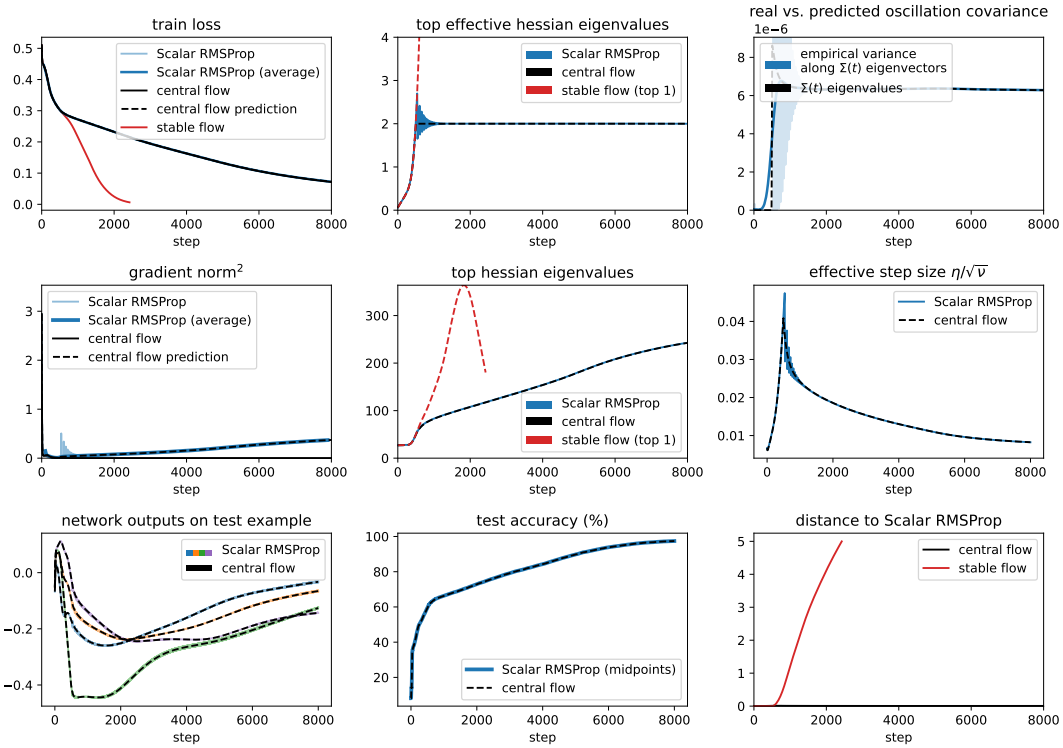


Figure 52.13: Scalar RMSProp central flow for a Transformer with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

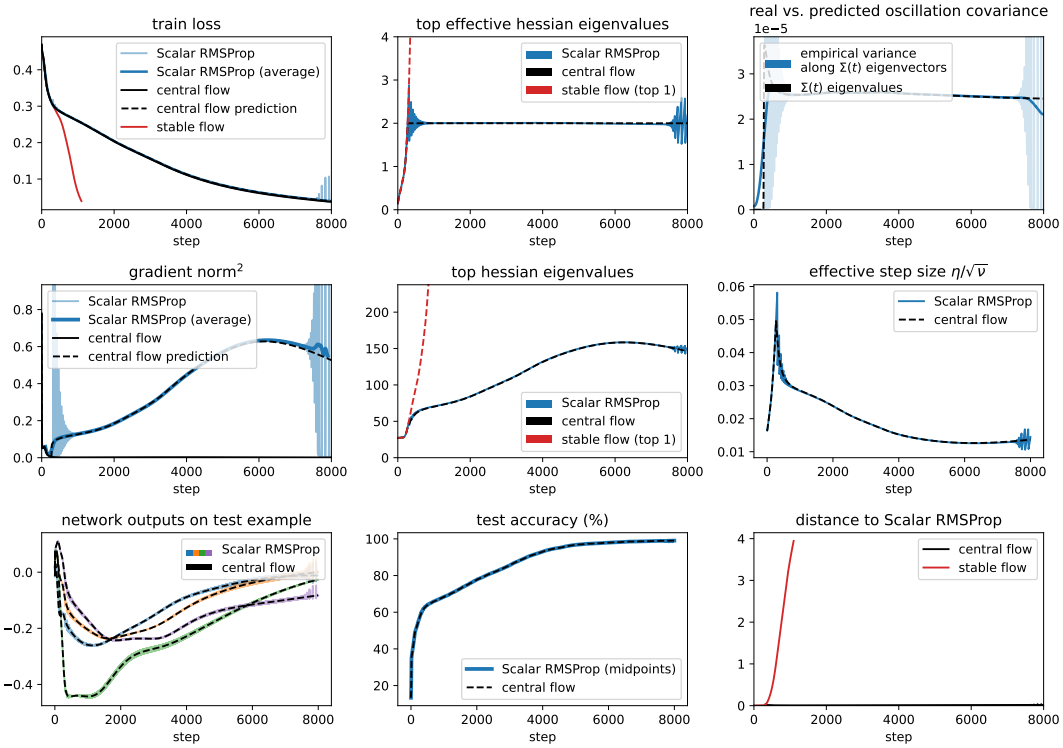


Figure 52.14: Scalar RMSProp central flow for a Transformer with MSE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

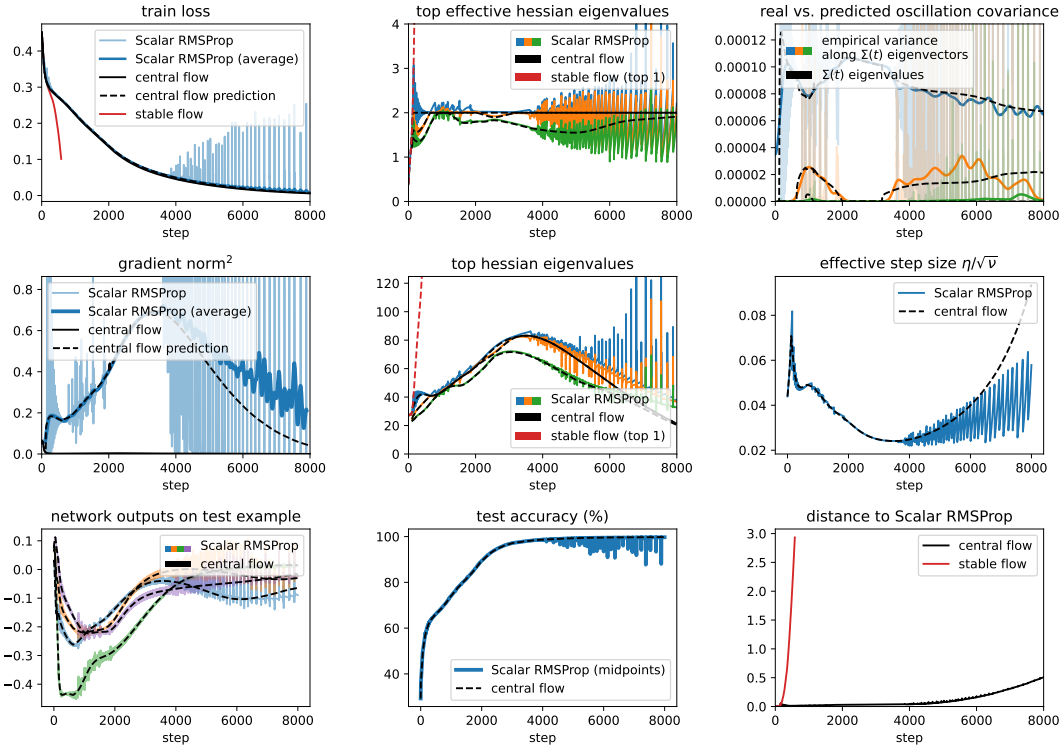


Figure 52.15: Scalar RMSProp central flow for a Transformer with MSE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

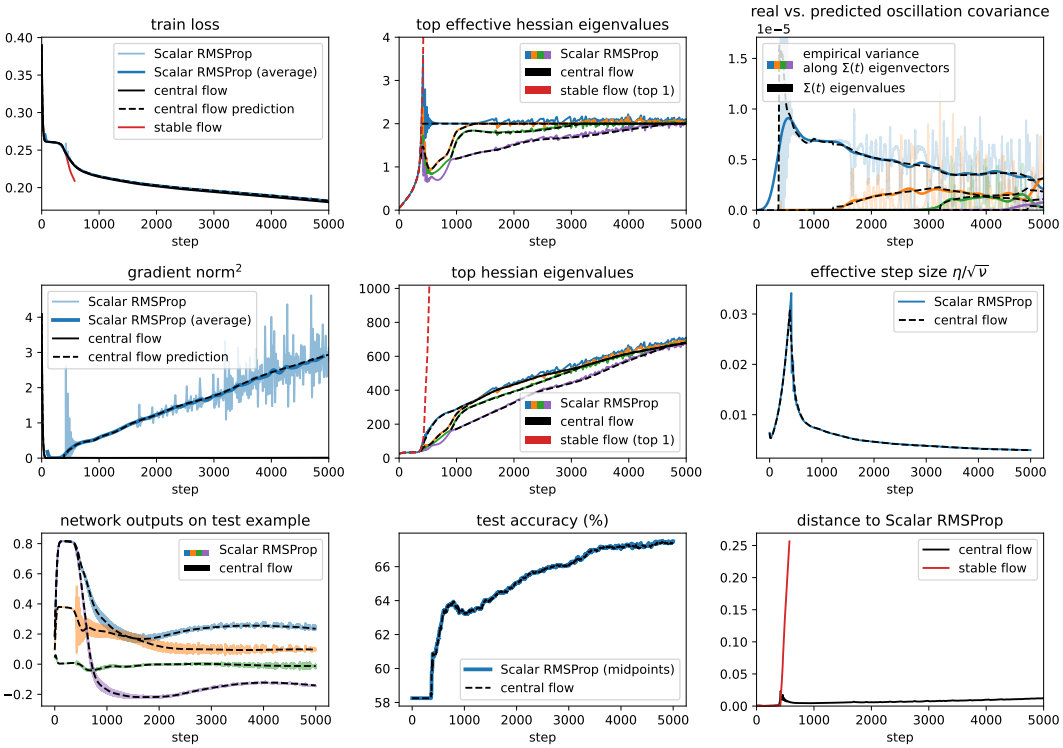


Figure 52.16: Scalar RMSProp central flow for a Mamba with MSE loss, $\eta = 0.007$, $\beta_2 = 0.99$, and bias correction.

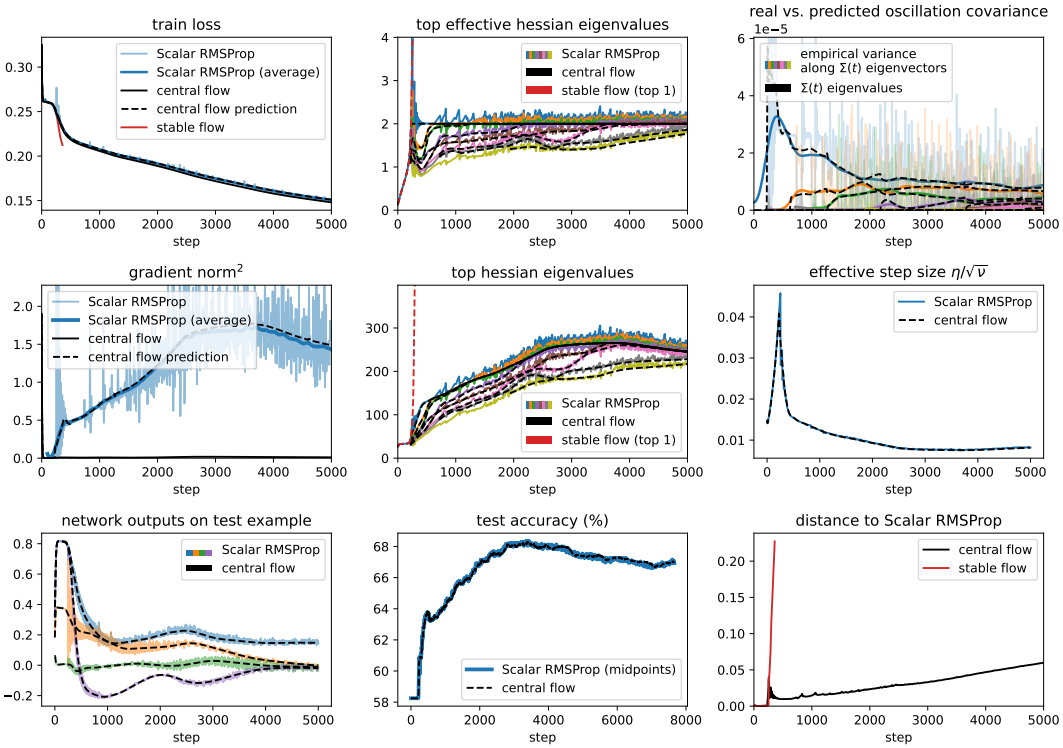


Figure 52.17: Scalar RMSProp central flow for a Mamba with MSE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

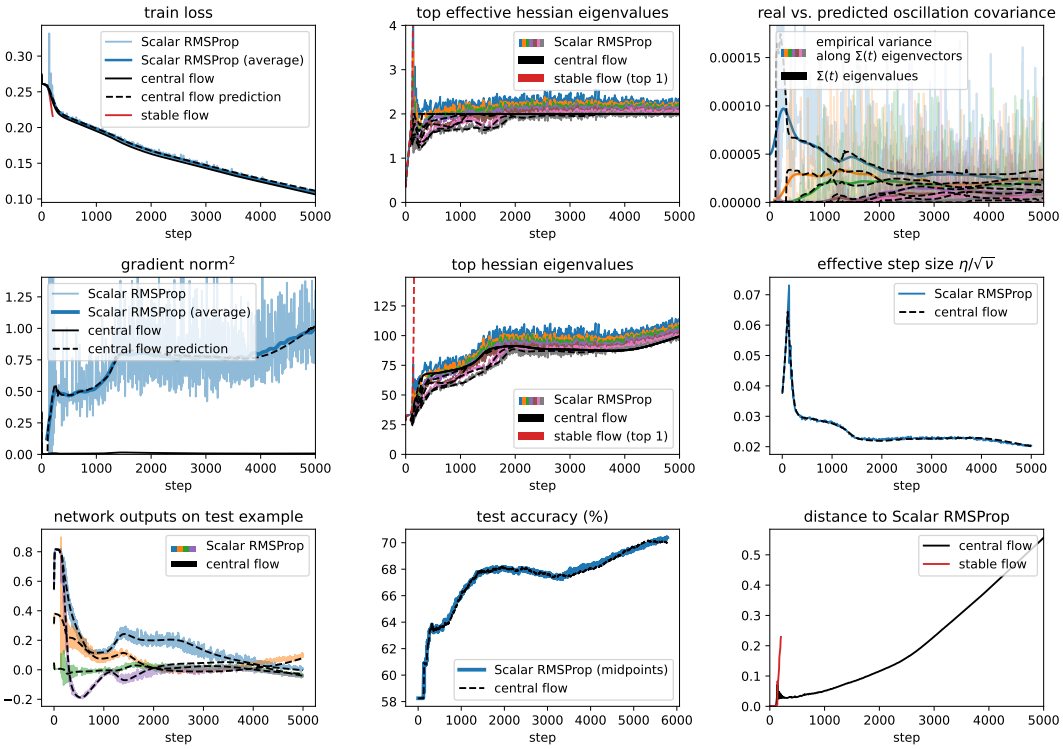


Figure 52.18: Scalar RMSProp central flow for a Mamba with MSE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

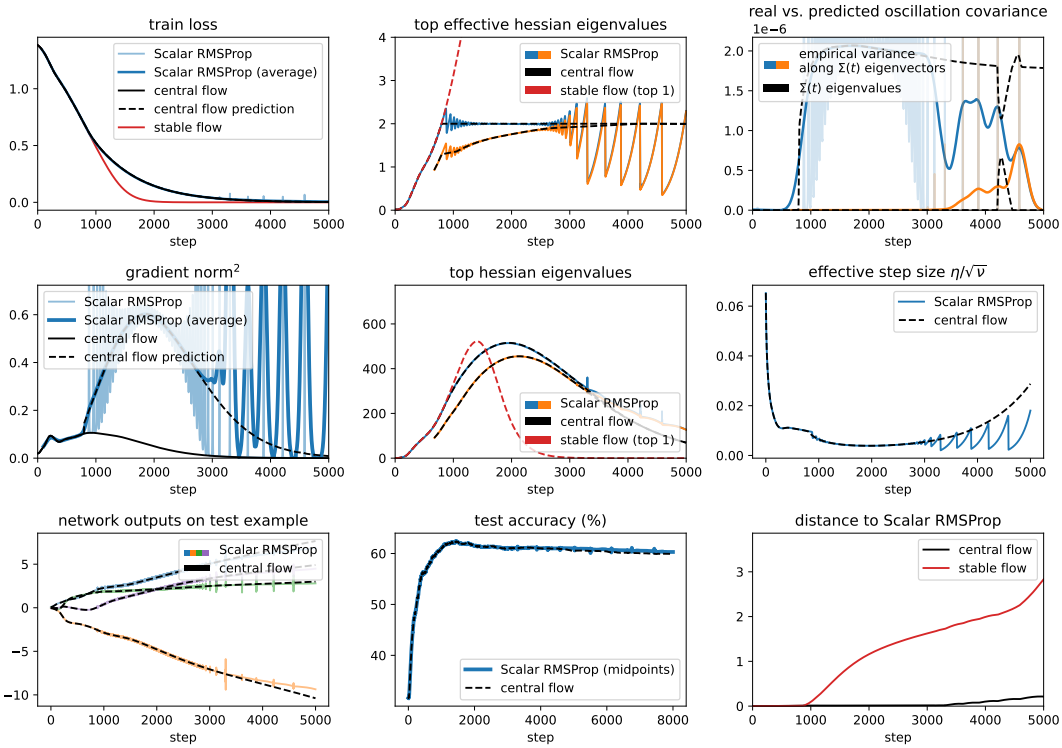


Figure 53.1: Scalar RMSProp central flow for a CNN with CE loss, $\eta = 0.003$, $\beta_2 = 0.99$, and bias correction.

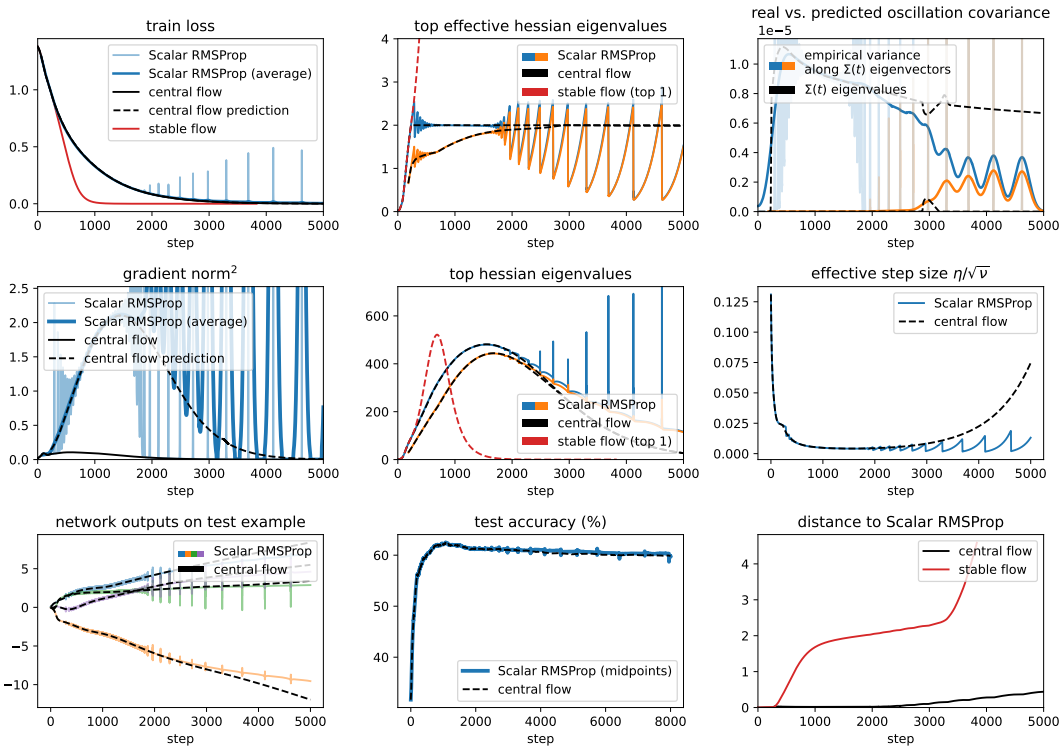


Figure 53.2: Scalar RMSProp central flow for a CNN with CE loss, $\eta = 0.006$, $\beta_2 = 0.99$, and bias correction.

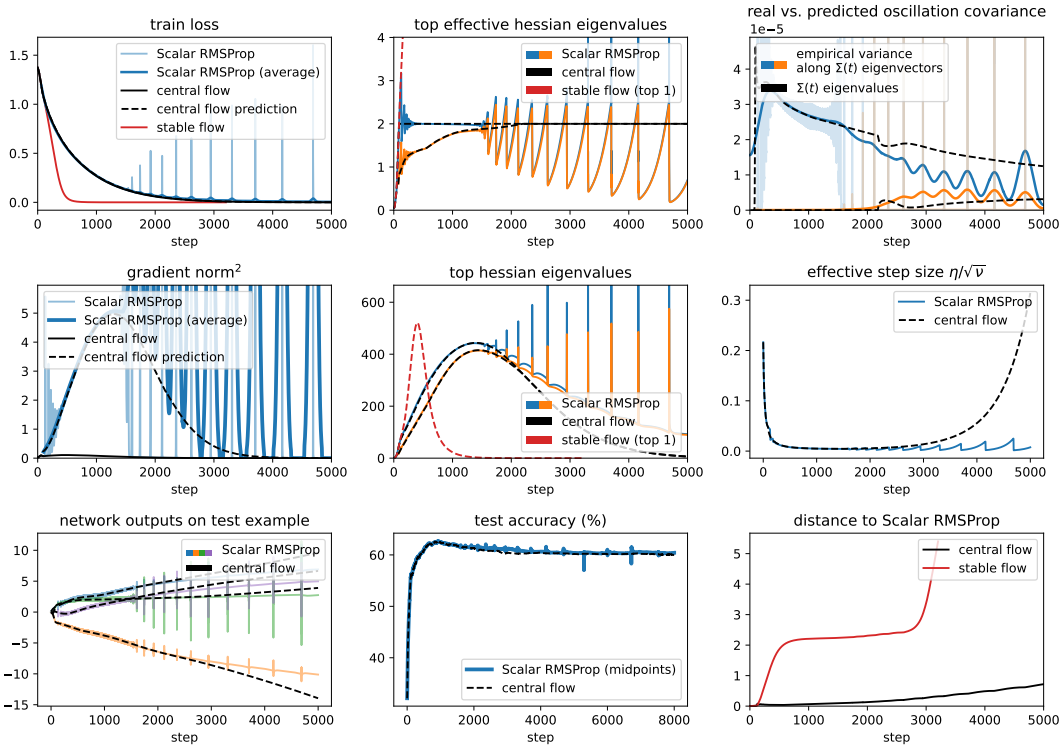


Figure 53.3: Scalar RMSProp central flow for a CNN with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

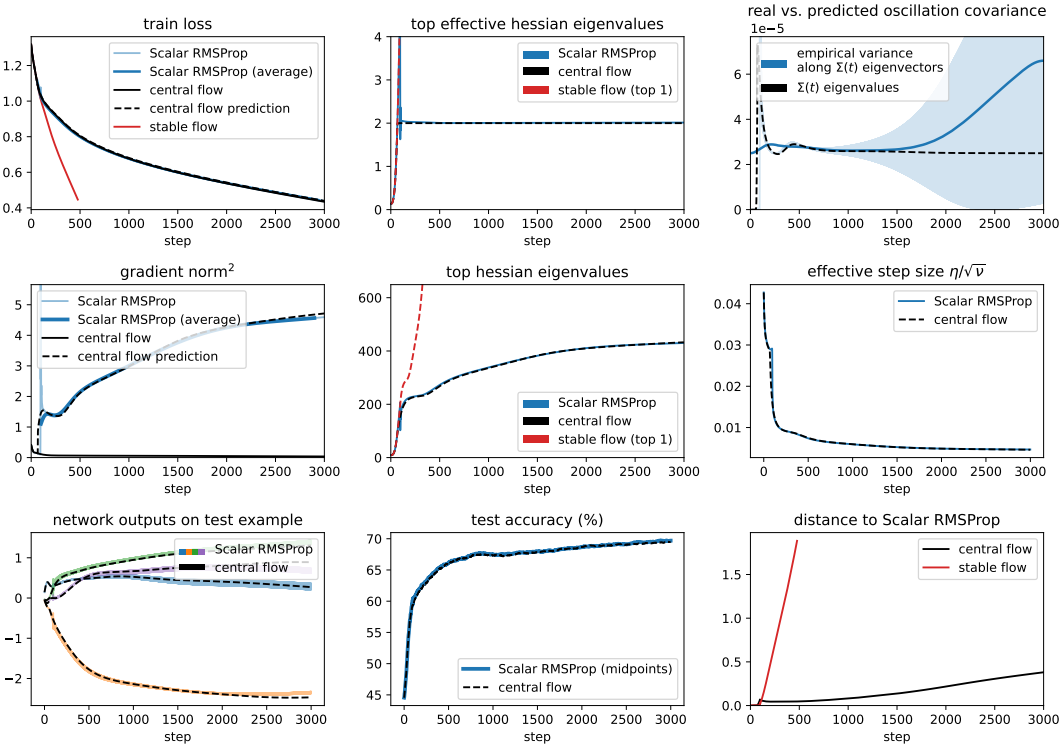


Figure 53.4: Scalar RMSProp central flow for a ResNet with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

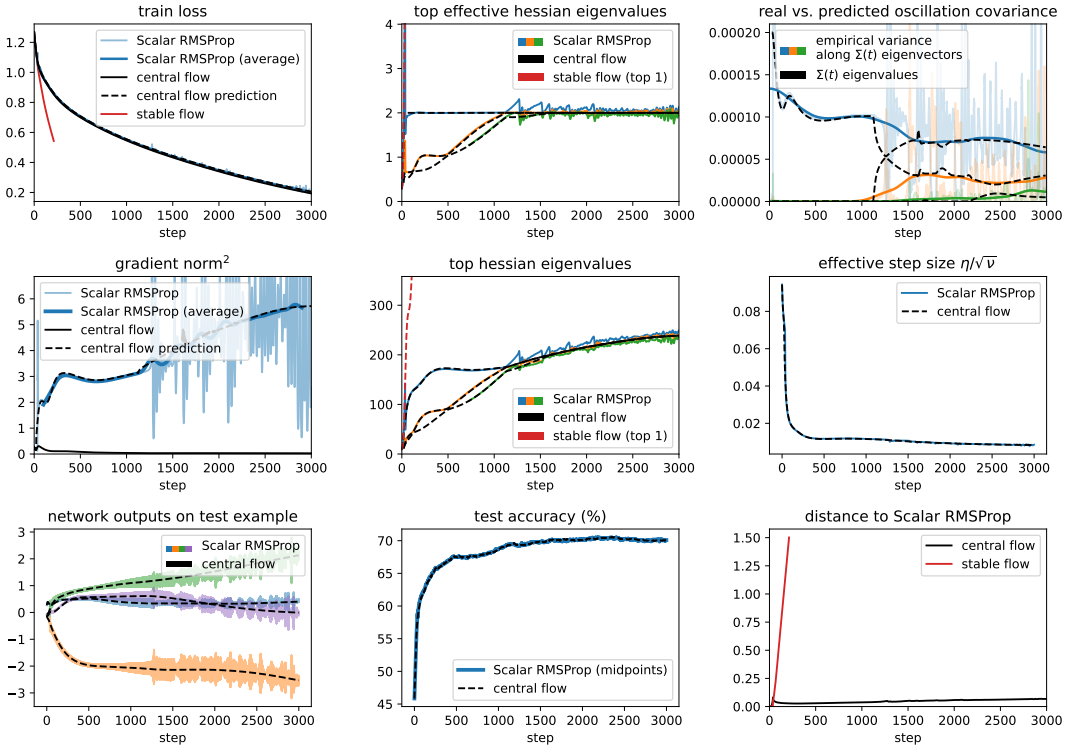


Figure 53.5: Scalar RMSProp central flow for a ResNet with CE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

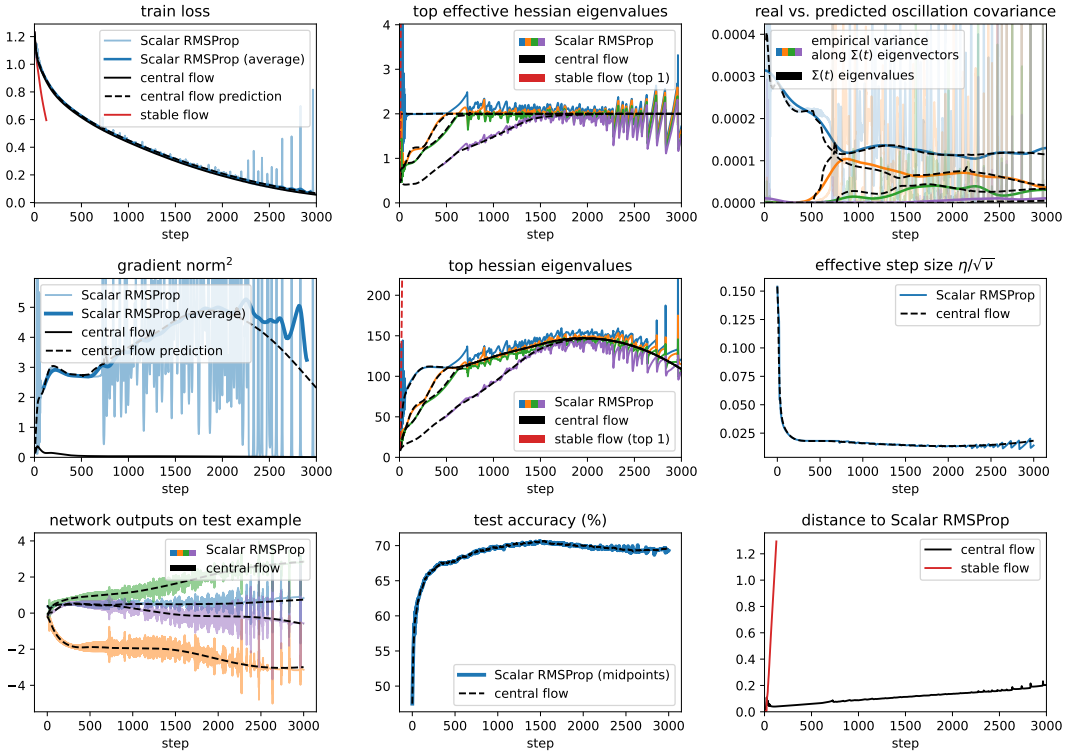


Figure 53.6: Scalar RMSProp central flow for a ResNet with CE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

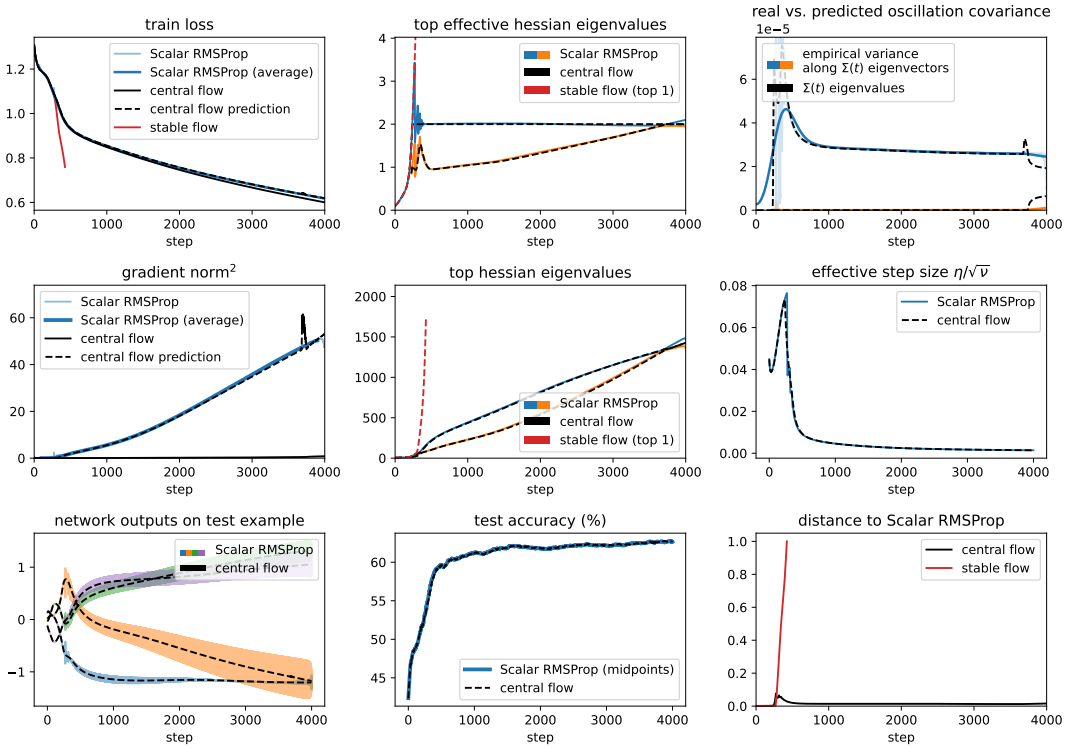


Figure 53.7: Scalar RMSProp central flow for a ViT with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

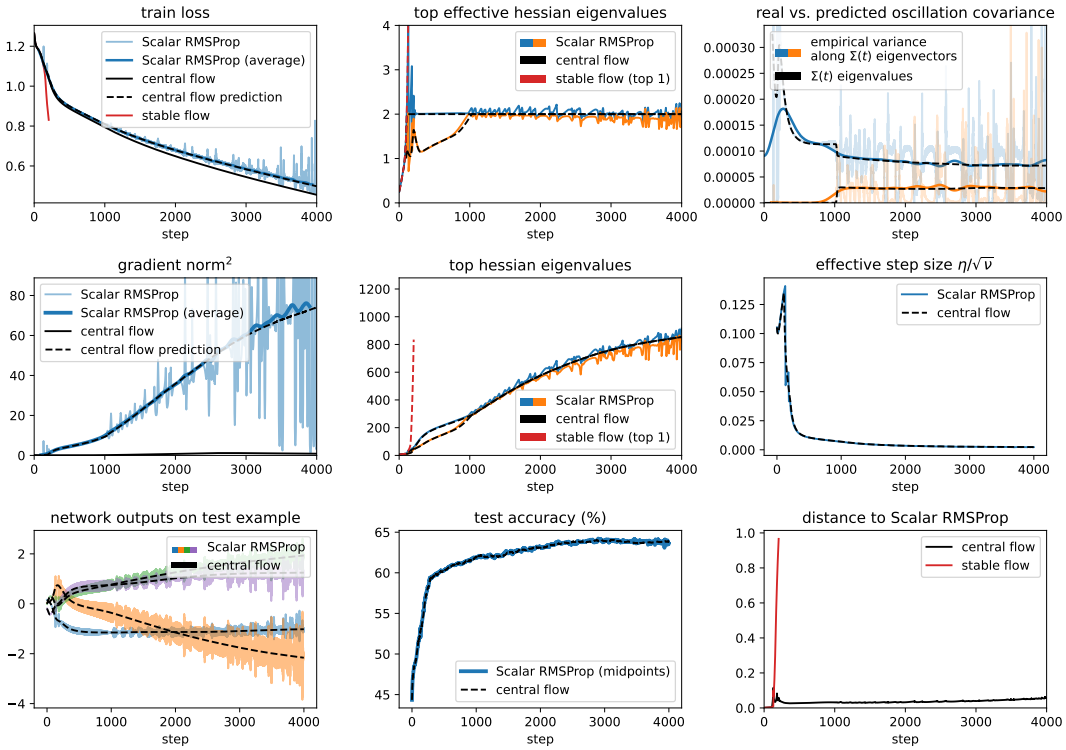


Figure 53.8: Scalar RMSProp central flow for a ViT with CE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

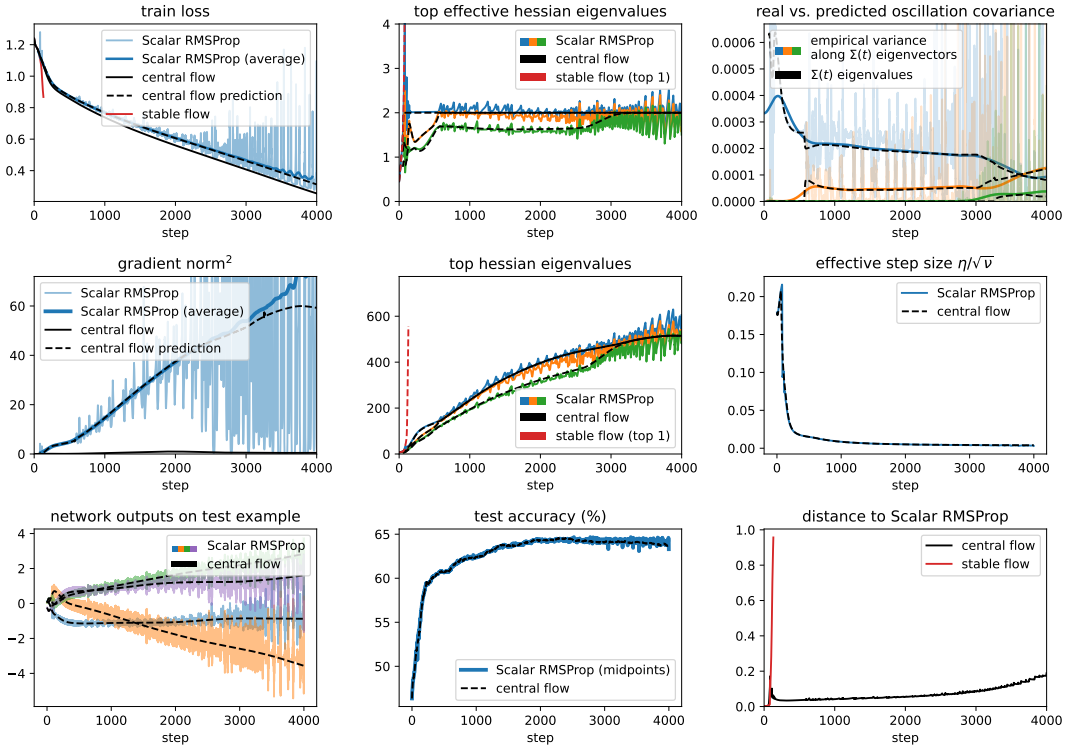


Figure 53.9: Scalar RMSProp central flow for a ViT with CE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

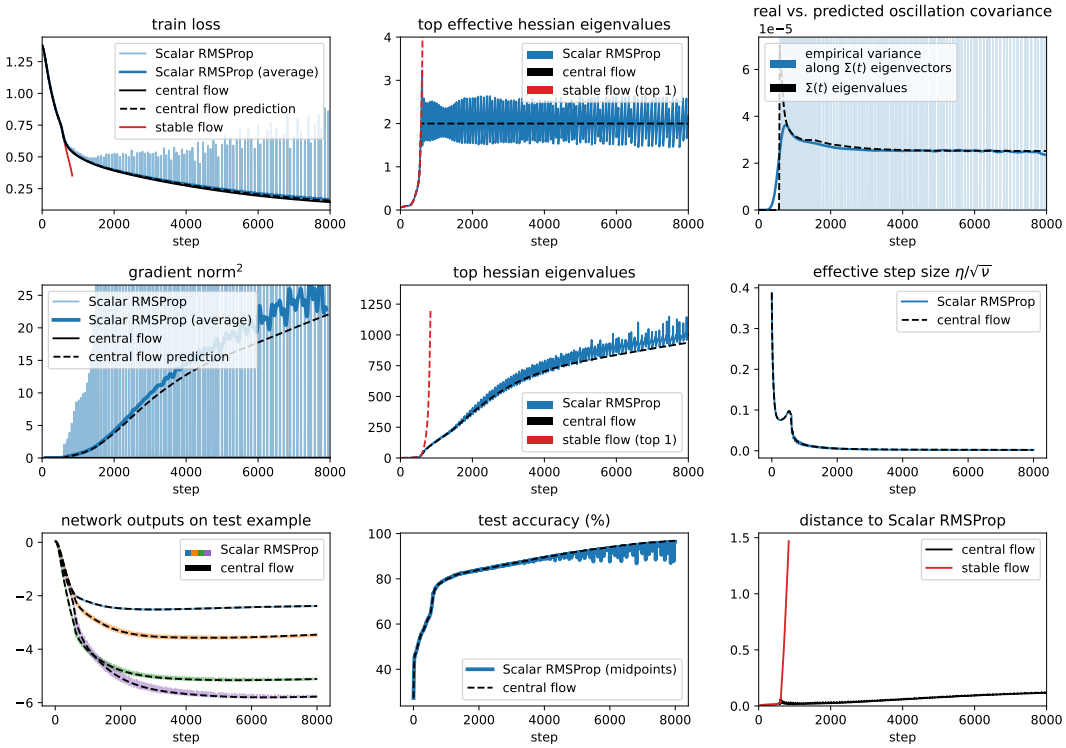


Figure 53.10: Scalar RMSProp central flow for a LSTM with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

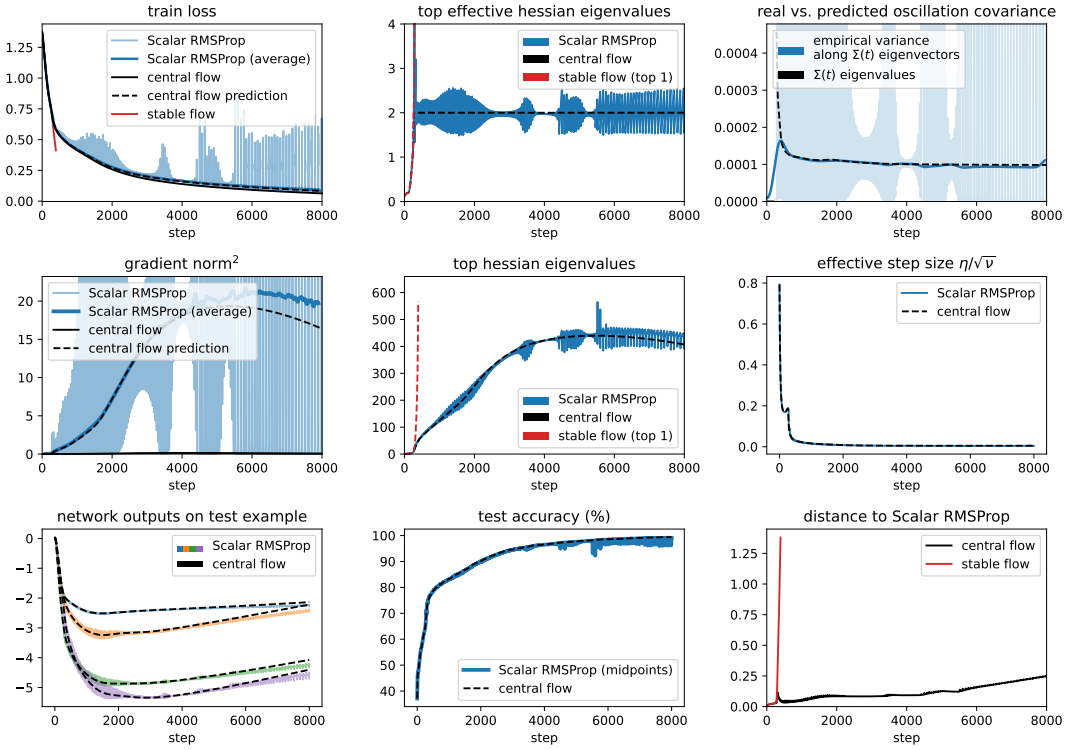


Figure 53.11: Scalar RMSProp central flow for a LSTM with CE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

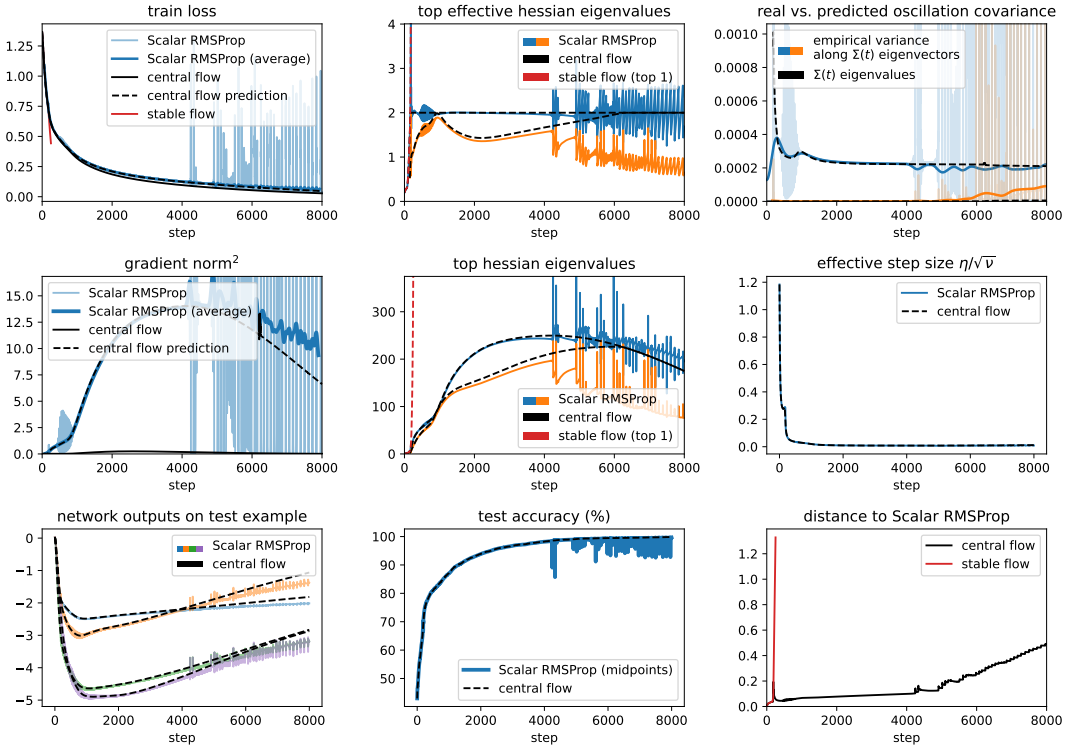


Figure 53.12: Scalar RMSProp central flow for a LSTM with CE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

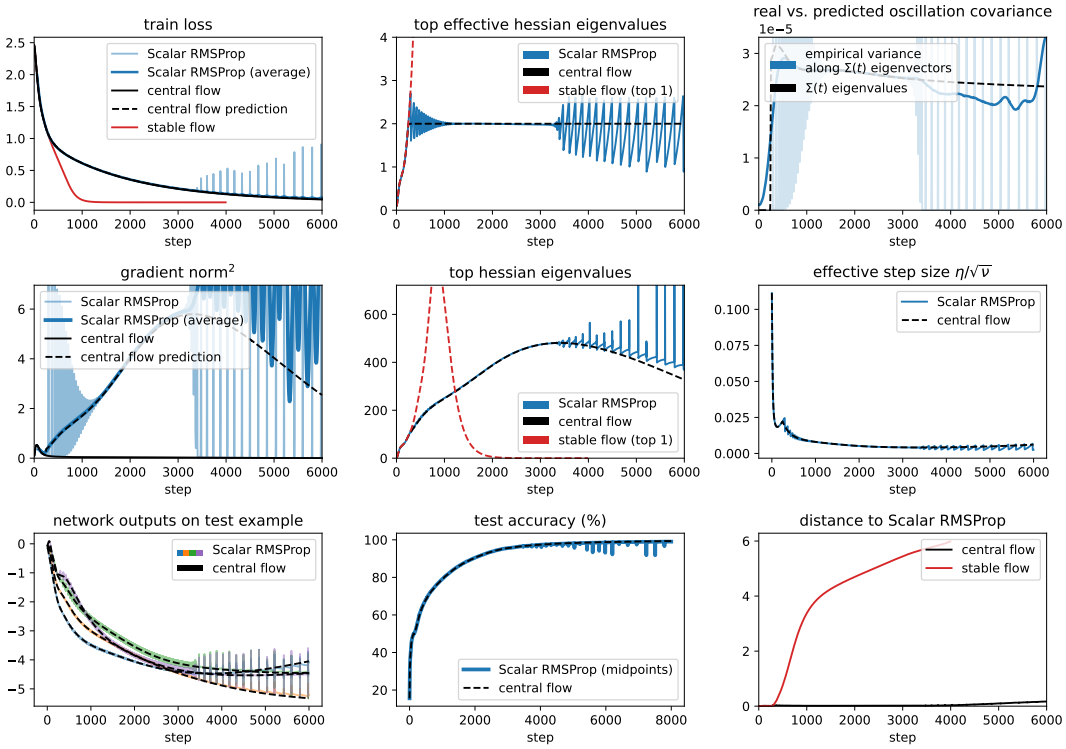


Figure 53.13: Scalar RMSProp central flow for a Transformer with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

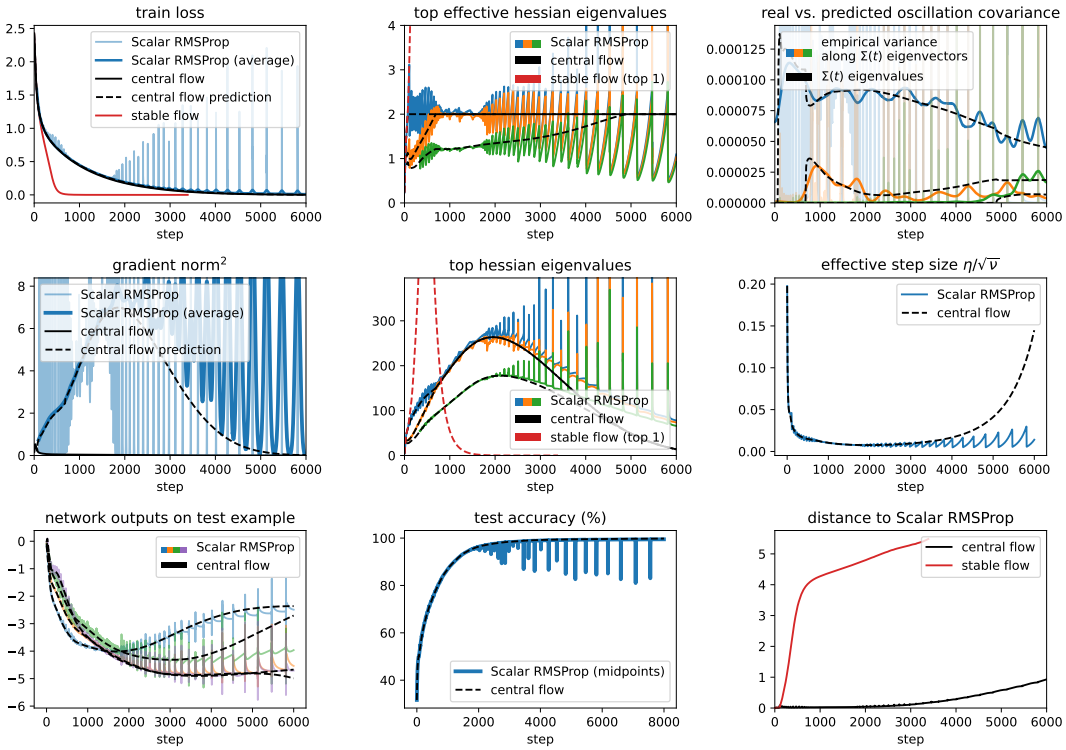


Figure 53.14: Scalar RMSProp central flow for a Transformer with CE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

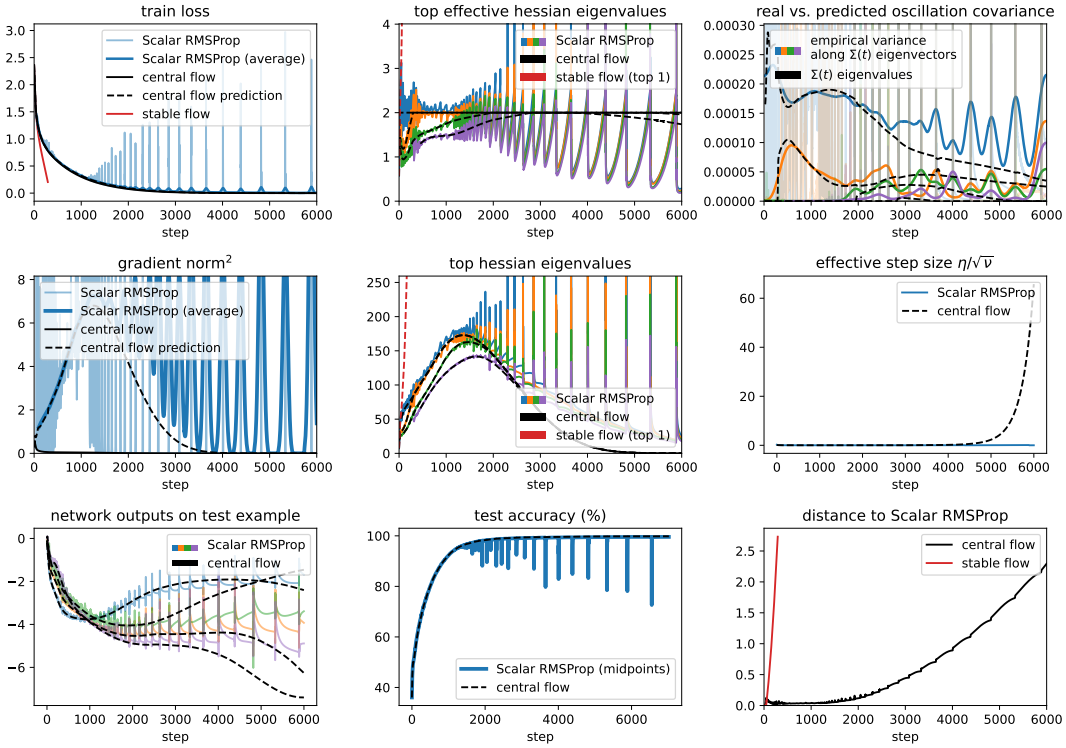


Figure 53.15: Scalar RMSProp central flow for a Transformer with CE loss, $\eta = 0.03$, $\beta_2 = 0.99$, and bias correction.

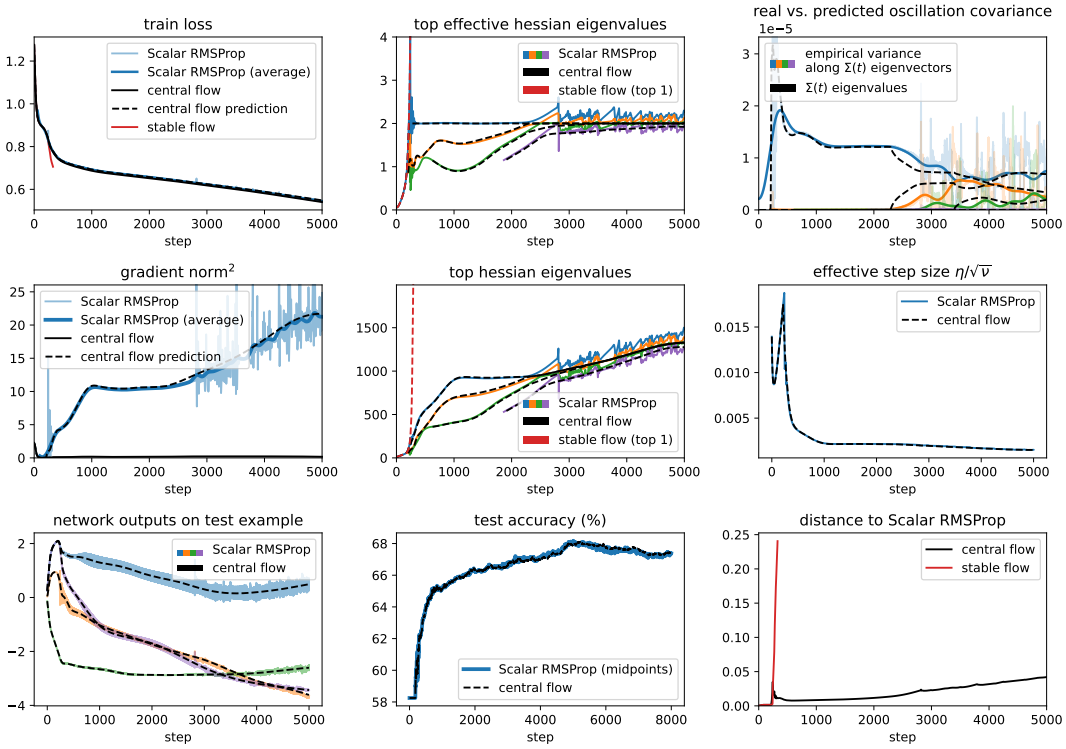


Figure 53.16: Scalar RMSProp central flow for a Mamba with CE loss, $\eta = 0.007$, $\beta_2 = 0.99$, and bias correction.

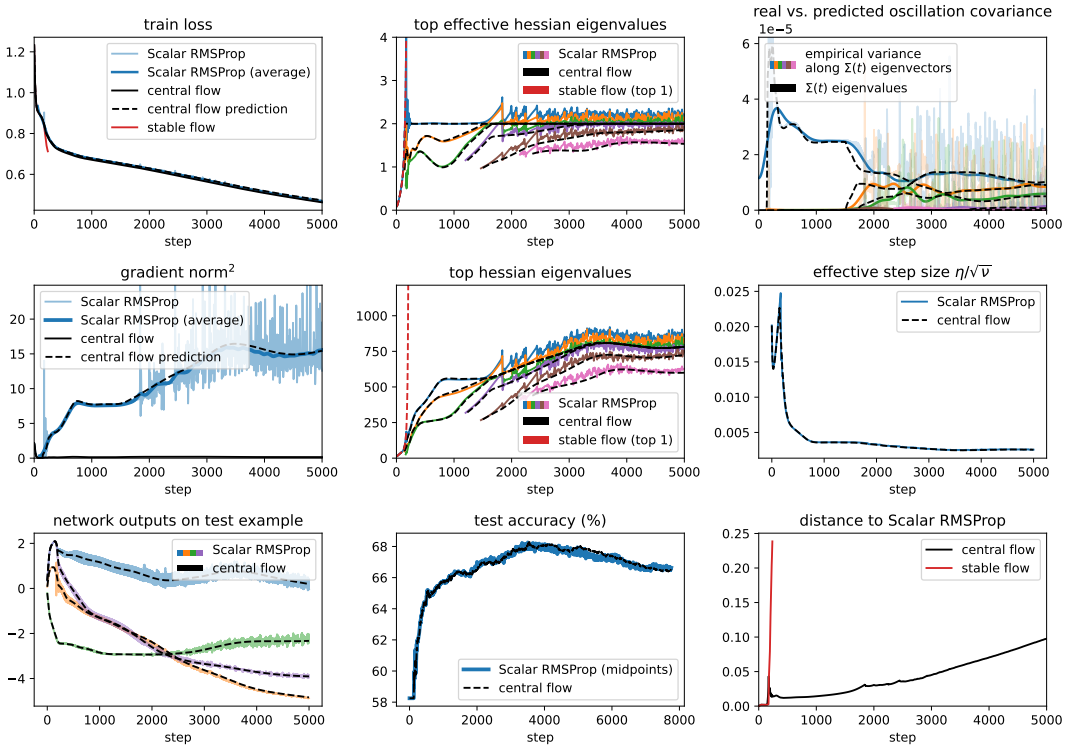


Figure 53.17: Scalar RMSProp central flow for a Mamba with CE loss, $\eta = 0.01$, $\beta_2 = 0.99$, and bias correction.

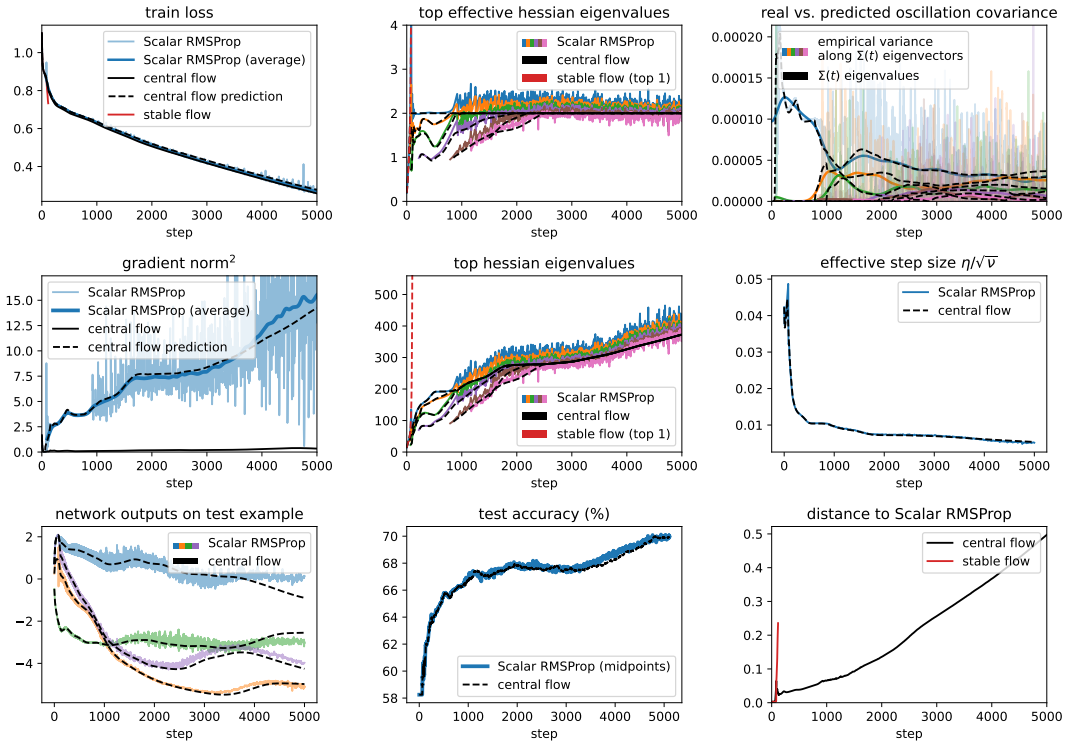


Figure 53.18: Scalar RMSProp central flow for a Mamba with CE loss, $\eta = 0.02$, $\beta_2 = 0.99$, and bias correction.

E.3 RMSProp

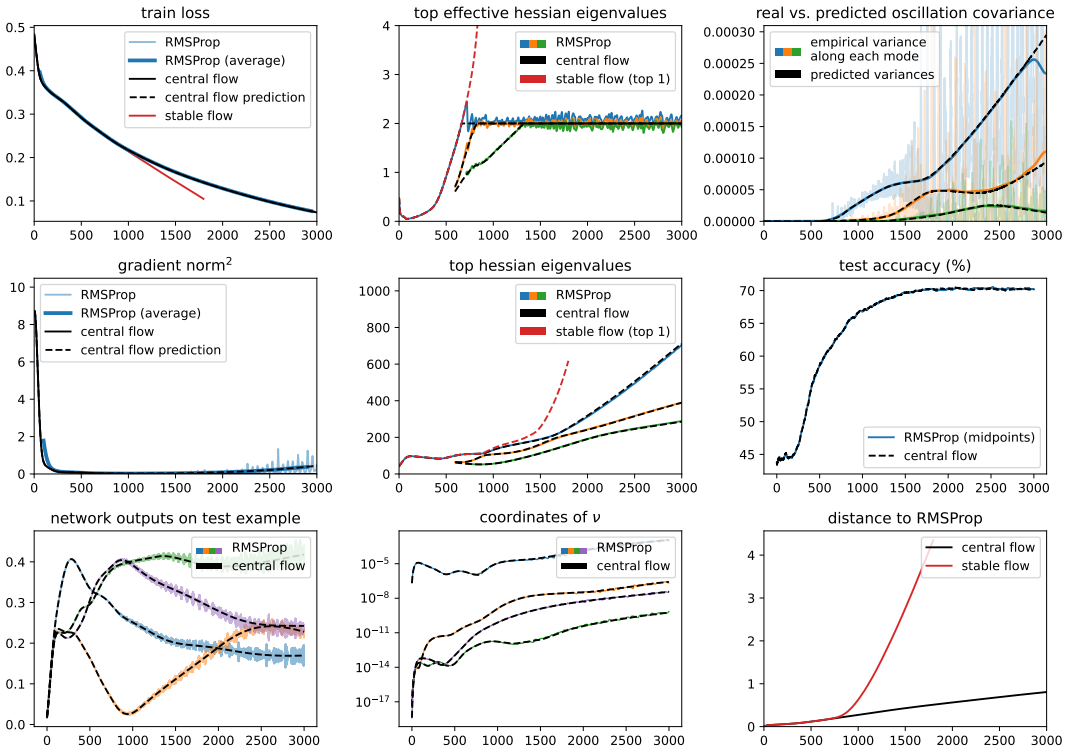


Figure 54: **Annotated example of a RMSProp experiment.** Using RMSProp with $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$ and bias correction, we train a ResNet on a subset of CIFAR-10 with MSE loss. The central flow (black) accurately models the long-term trajectory of RMSProp (blue), whereas the stable flow (red) takes a different path. As described in Appendix B.1, we terminate the stable flow once the effective sharpness gets sufficiently large.

Top left: See Figure 48 caption. The central flow's prediction for the time-averaged loss is given by eq. (118).

Top center: We plot the top several eigenvalues of the effective Hessian $\text{diag}\left[\frac{\eta}{\sqrt{\nu}}\right] H(w)$ under both RMSProp (colors) and its central flow (dashed black). Under RMSProp, these eigenvalues equilibrate around the critical threshold 2, whereas under the central flow they are fixed exactly at 2. We also plot the top eigenvalue under the "stable flow" baseline (red), which increases far above 2.

Top right: We show that the central flow accurately predicts the covariance of the oscillations. In particular, we show that each nonzero eigenvalue $\lambda_i(t)$ of $P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}$ accurately predicts the P-whitened variance of oscillations along the corresponding eigenvector $v_i(t)$, as we expect from eq. (120). In black, we plot the nonzero eigenvalues of $P(t)^{1/2}\,\Sigma(t)\,P(t)^{1/2}$. In faint colors, we plot the squared magnitude of the P-whitened displacement between RMSProp and the central flow along each eigenvector $v_i(t)$ (see eq. (120)), and in thick colors, we plot the time-averages of these displacements, i.e. the empirical variances of the oscillations. Observe that each eigenvalue accurately predicts the variance of the oscillations along the corresponding eigenvector.

Middle left: See Figure 48 caption. The central flow's prediction for the time-average is given by eq. (119).

Middle center: See Figure 51 caption. The central flow's prediction for the time-averaged loss is given by eq. (118).

Middle right: See Figure 48 caption.

Bottom left: See Figure 51 caption.

Bottom center: We plot four arbitrary coordinates of the EMA ν under both RMSProp (colors) and the central flow (dashed black). The coordinates of ν are oscillatory along the RMSProp trajectory, while varying smoothly along the central flow.

Bottom right: See Figure 48 caption.

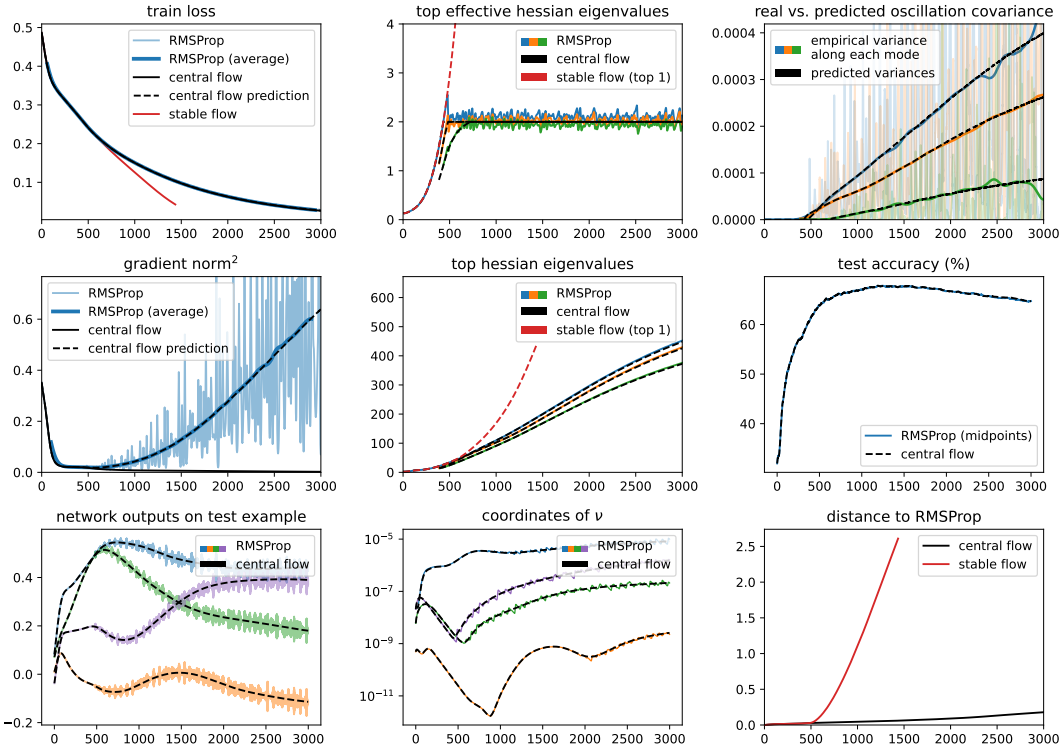


Figure 55.1: RMSProp central flow for a CNN with MSE loss, $\eta = 1\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

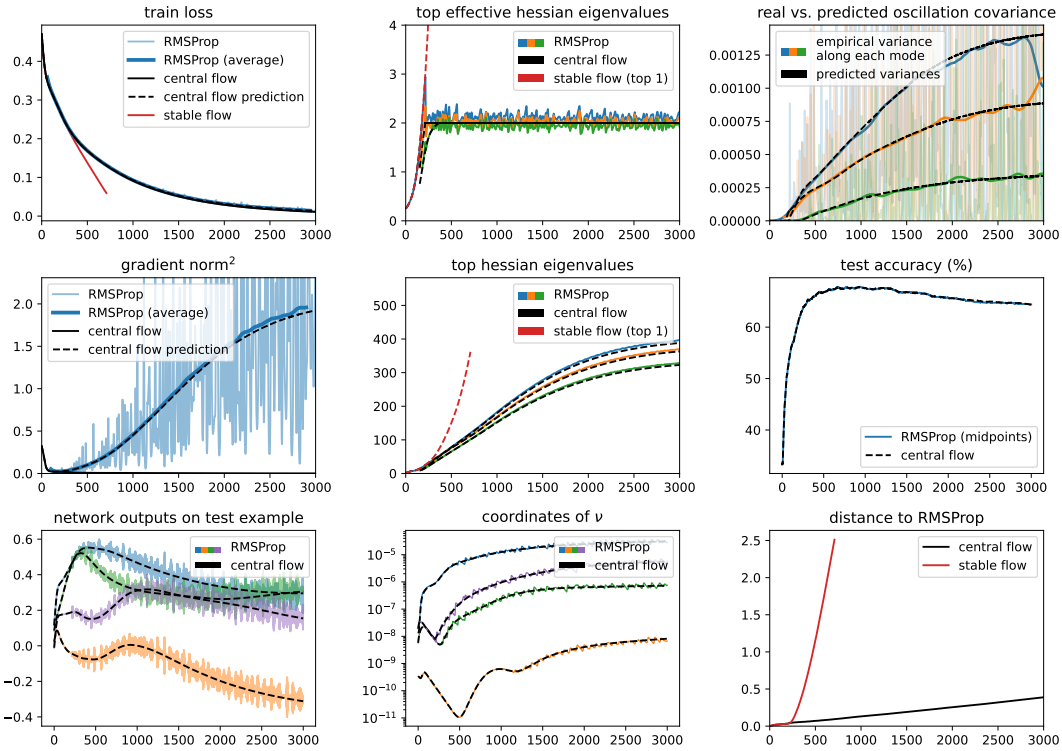


Figure 55.2: RMSProp central flow for a CNN with MSE loss, $\eta = 2\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

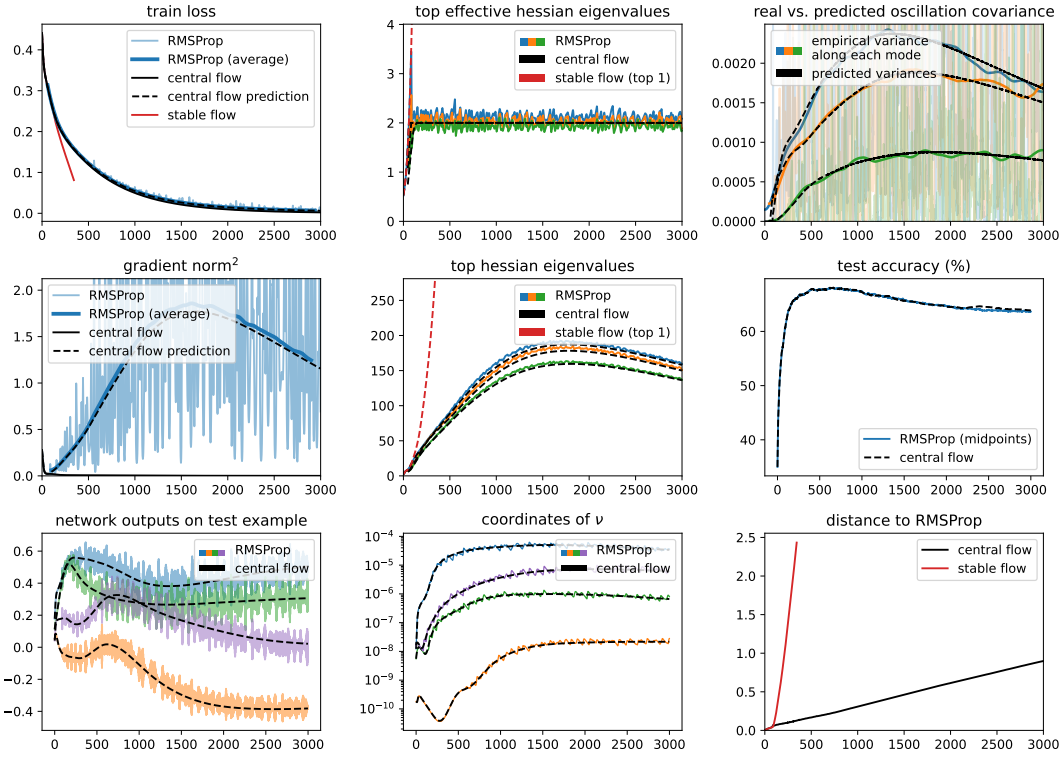


Figure 55.3: RMSProp central flow for a CNN with MSE loss, $\eta = 4\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

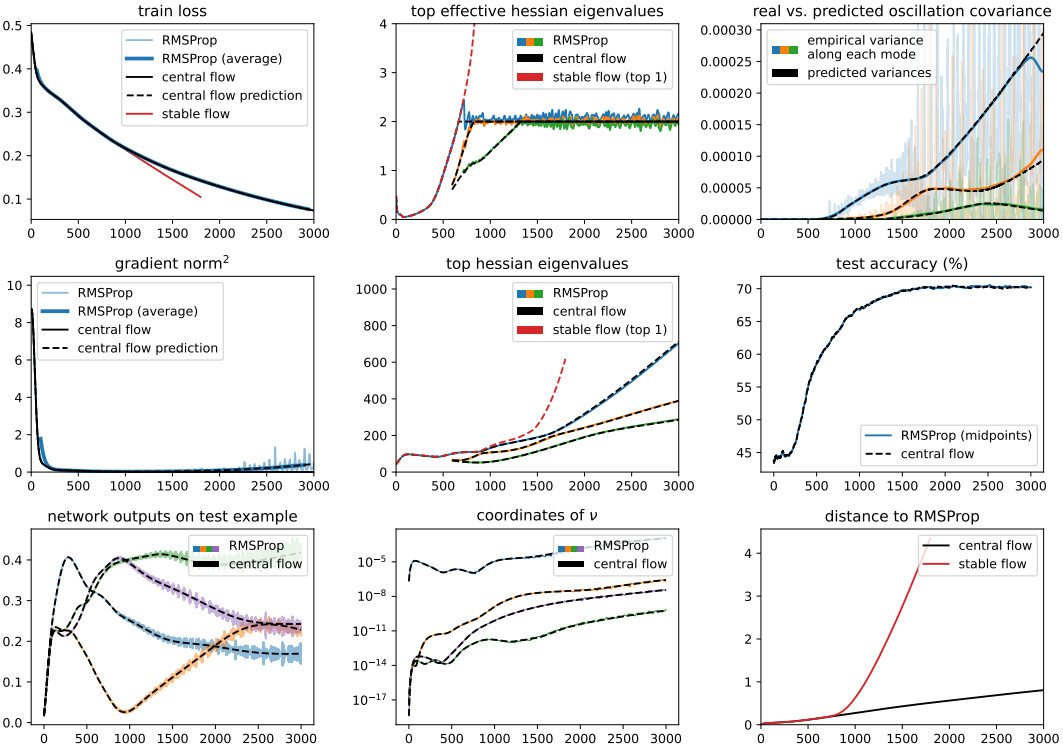


Figure 55.4: RMSProp central flow for a ResNet with MSE loss, $\eta = 2\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

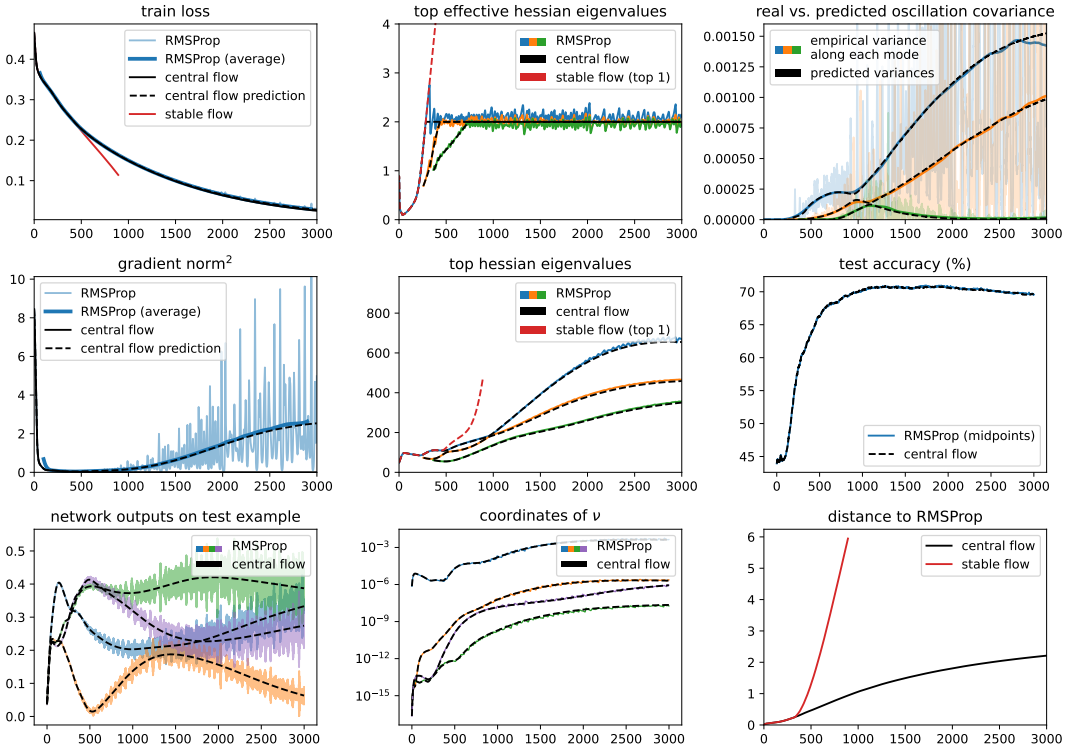


Figure 55.5: RMSProp central flow for a ResNet with MSE loss, $\eta = 4\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

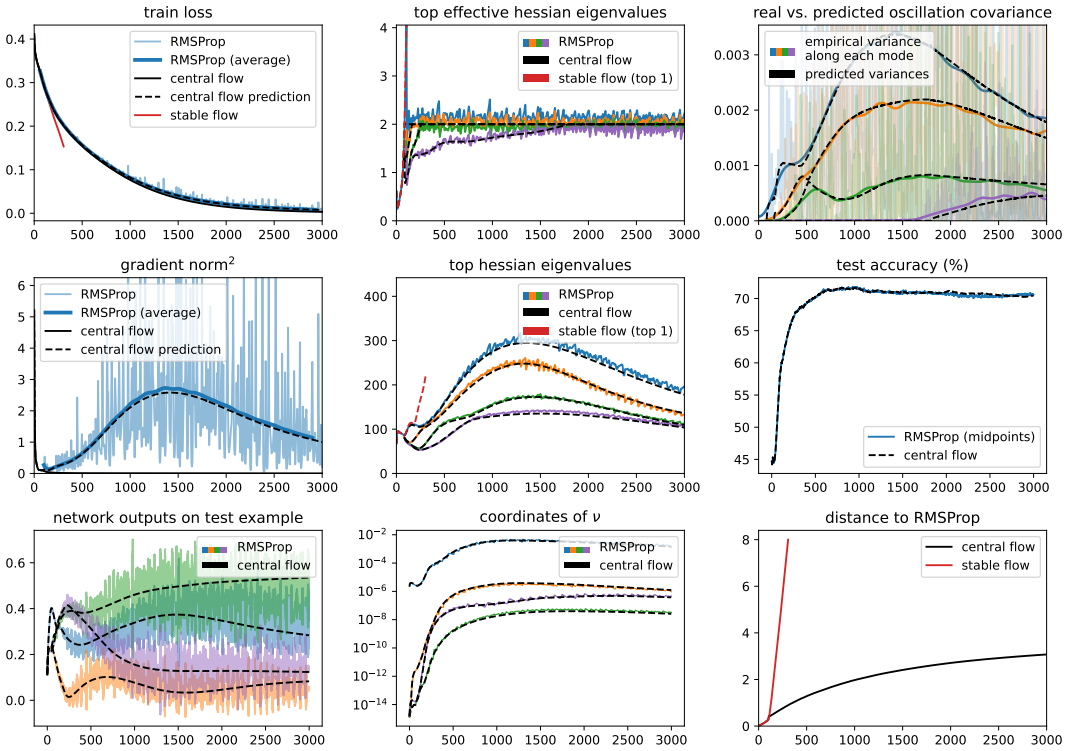


Figure 55.6: RMSProp central flow for a ResNet with MSE loss, $\eta = 0.0001$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

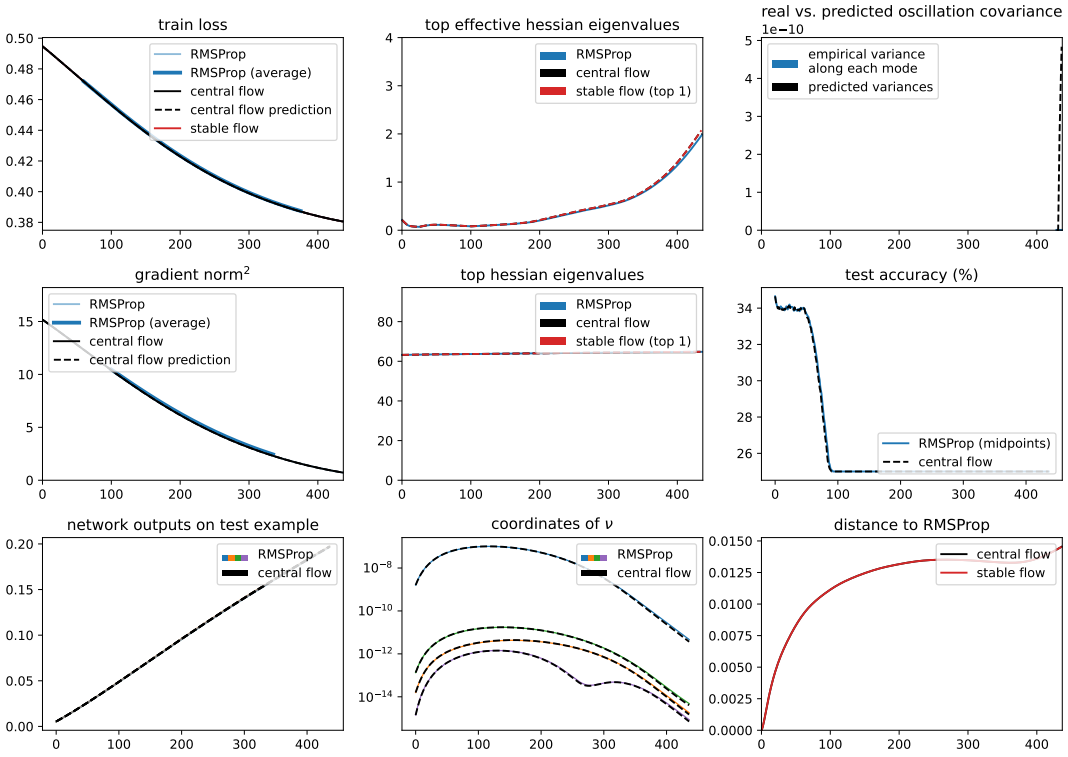


Figure 55.7: RMSProp central flow for a ViT with MSE loss, $\eta = 7\text{e-}06$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

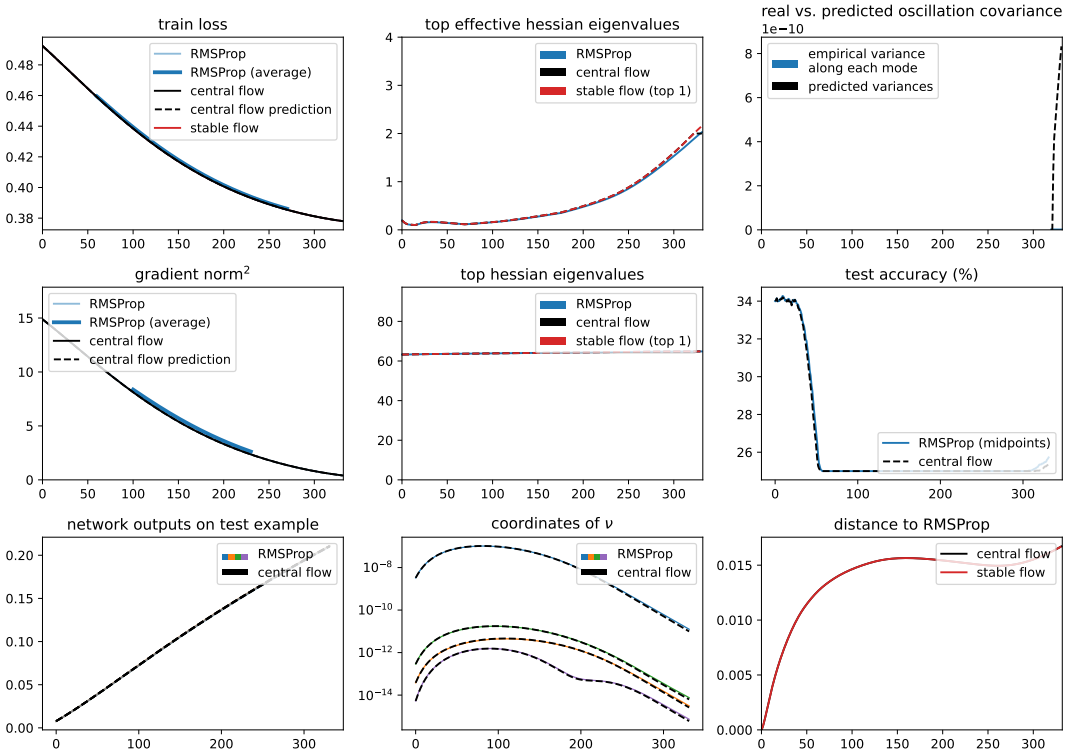


Figure 55.8: RMSProp central flow for a ViT with MSE loss, $\eta = 1\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

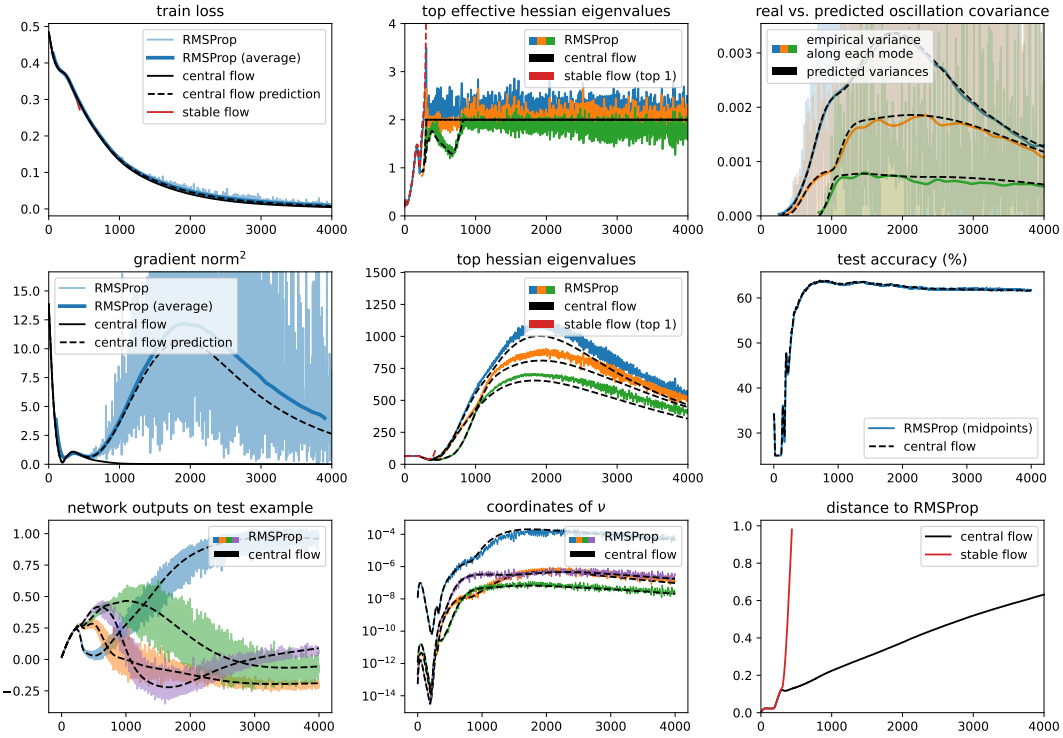


Figure 55.9: RMSProp central flow for a ViT with MSE loss, $\eta = 2\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

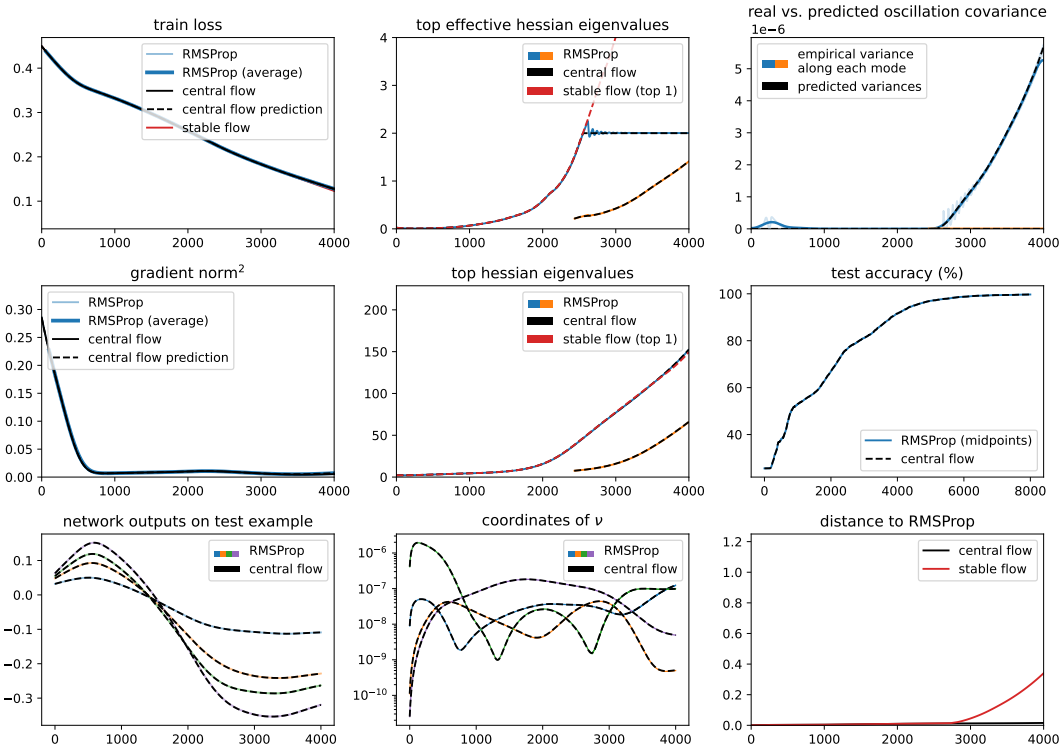


Figure 55.10: RMSProp central flow for a LSTM with MSE loss, $\eta = 1\text{e-}05$, $\beta_2 = 0.99$, $\epsilon = 1\text{e-}08$, and bias correction.

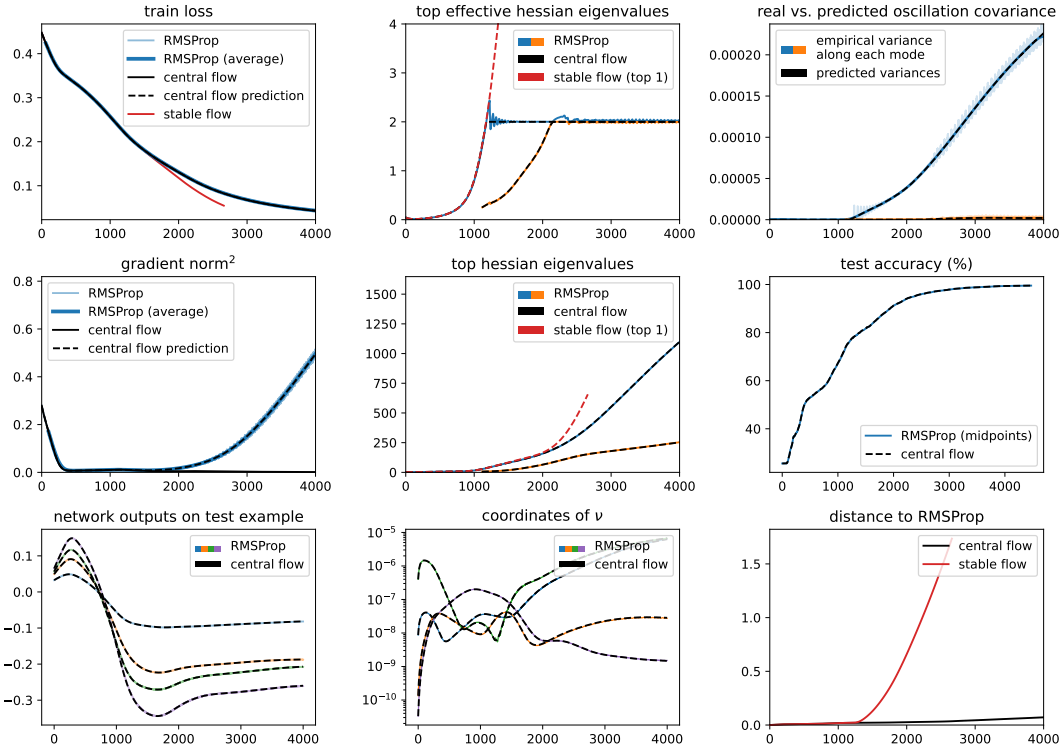


Figure 55.11: RMSProp central flow for a LSTM with MSE loss, η = 2e-05, β_2 = 0.99, ϵ = 1e-08, and bias correction.

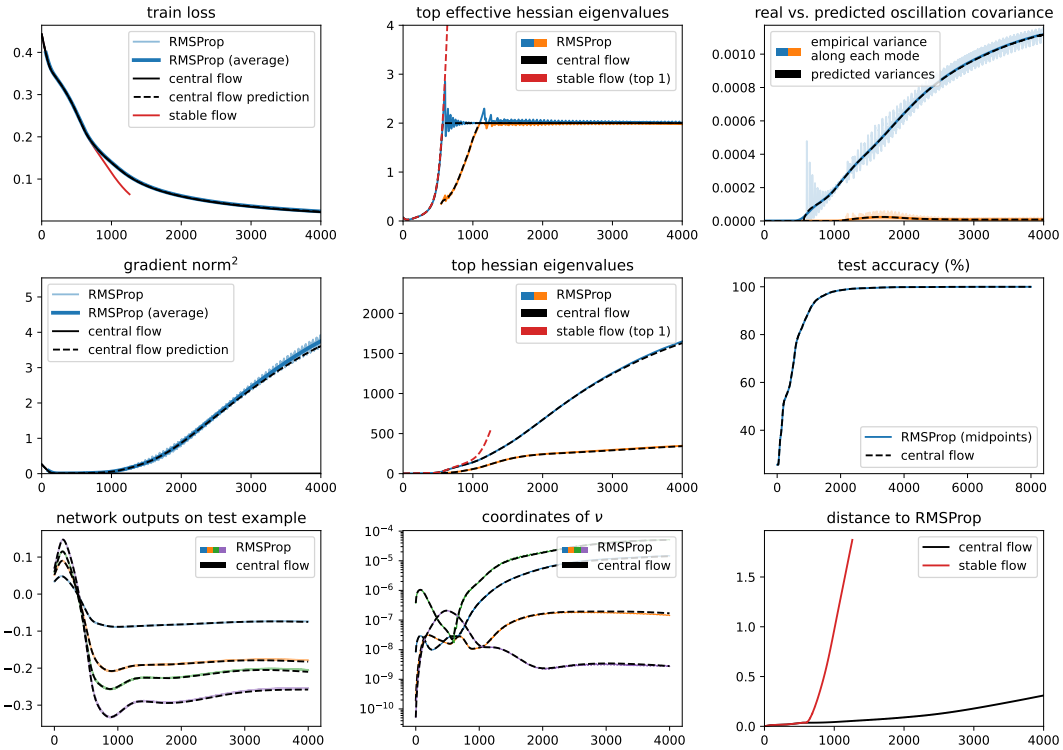


Figure 55.12: RMSProp central flow for a LSTM with MSE loss, η = 4e-05, β_2 = 0.99, ϵ = 1e-08, and bias correction.

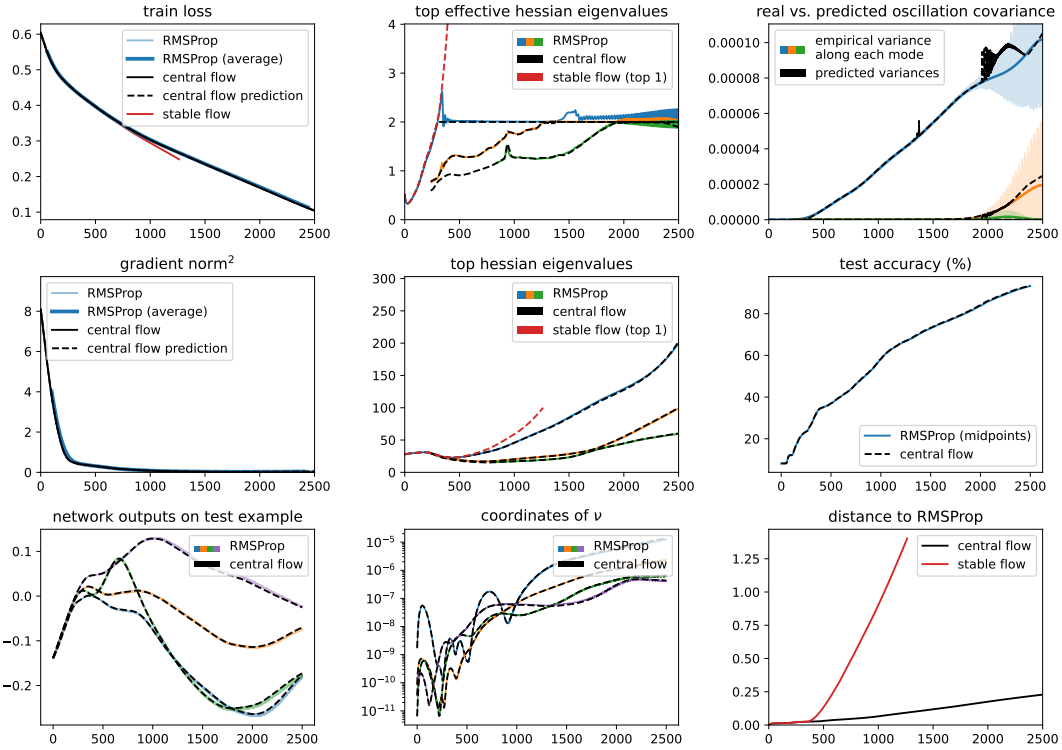


Figure 55.13: RMSProp central flow for a Transformer with MSE loss, $\eta =$ 2e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

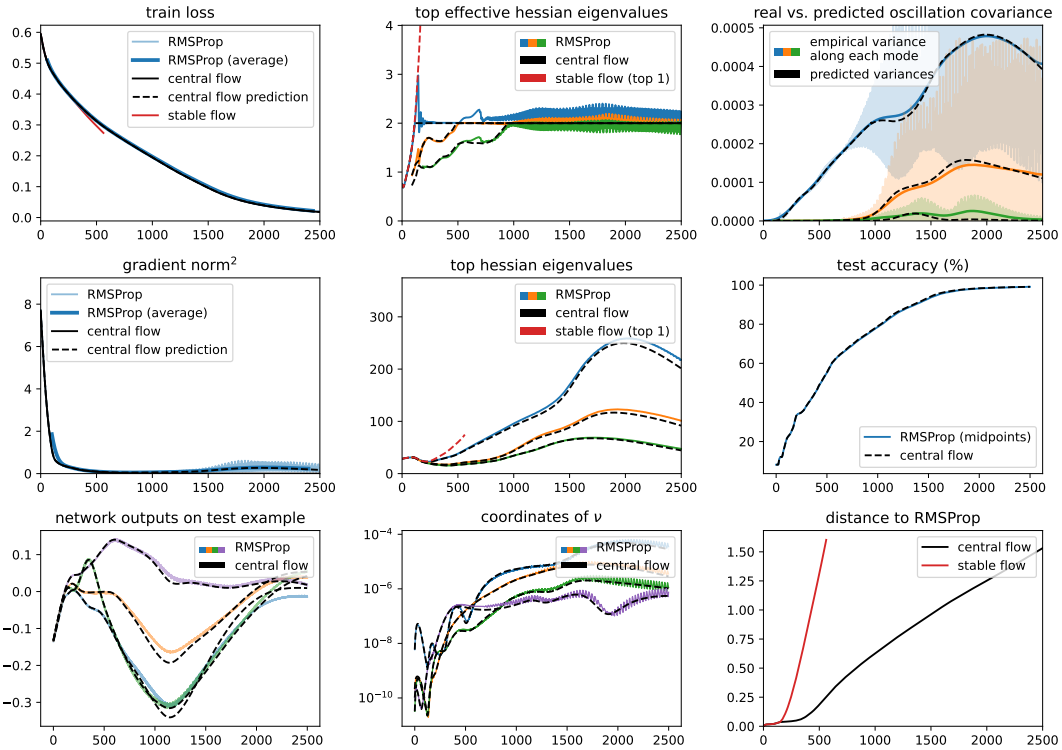


Figure 55.14: RMSProp central flow for a Transformer with MSE loss, $\eta =$ 4e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

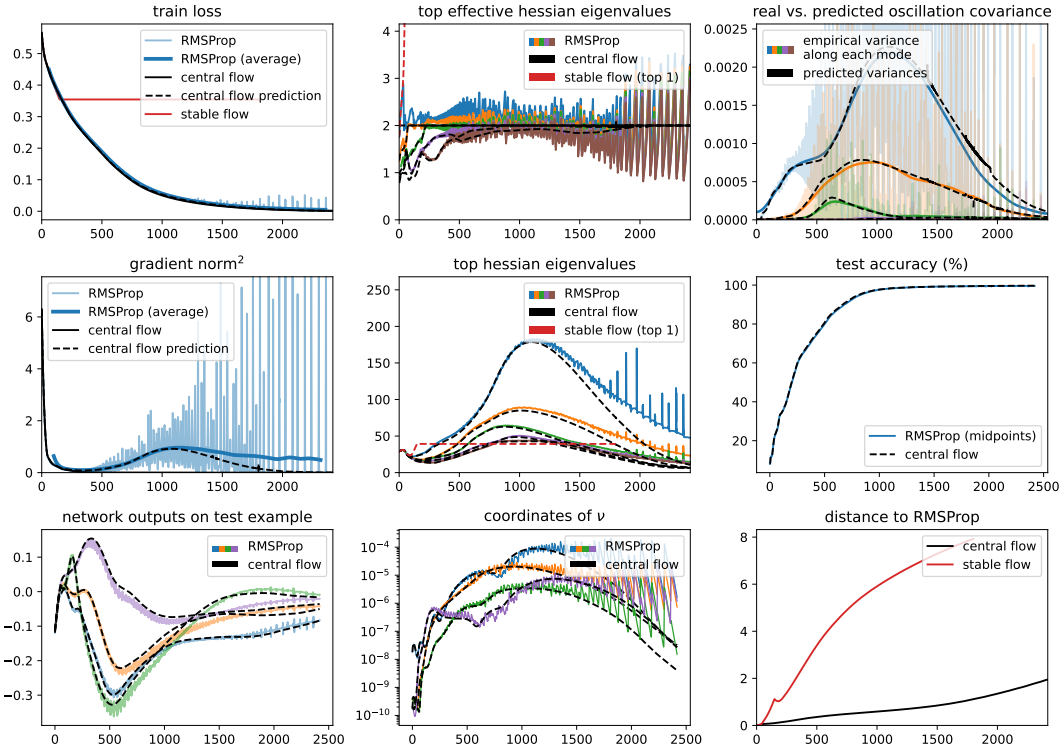


Figure 55.15: RMSProp central flow for a Transformer with MSE loss, $\eta = 0.0001$, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

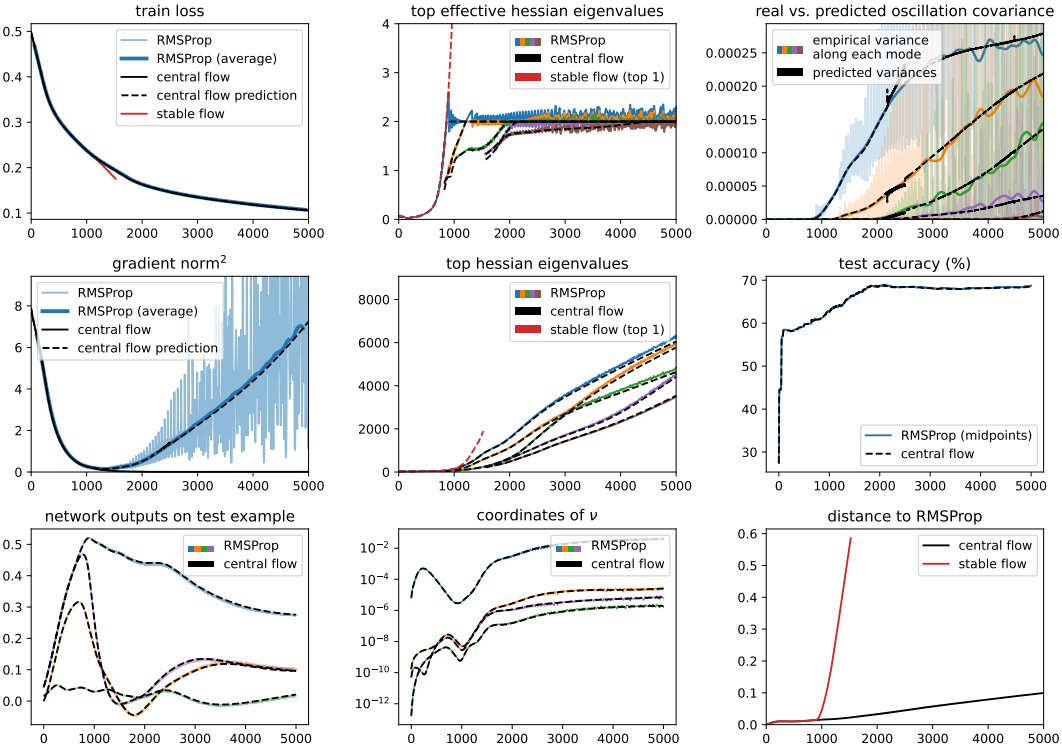


Figure 55.16: RMSProp central flow for a Mamba with MSE loss, $\eta =$ 1e-05, $\beta_2 = 0.99$, $\epsilon =$ 1e-08, and bias correction.

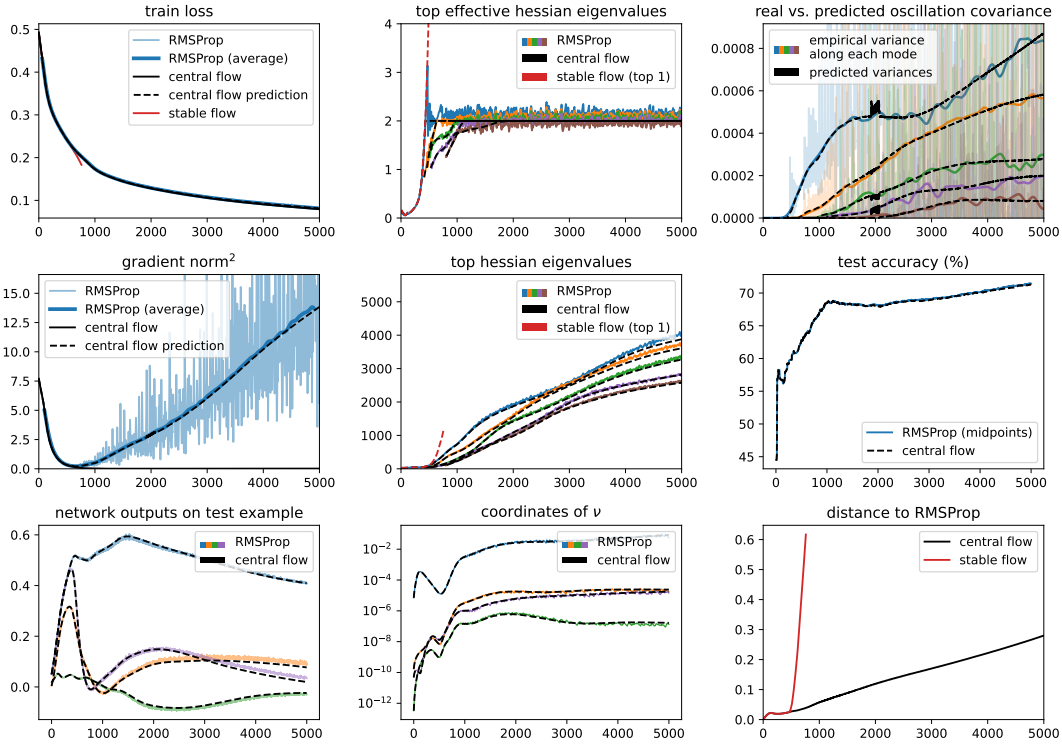


Figure 55.17: RMSProp central flow for a Mamba with MSE loss, η = 2e-05, β_2 = 0.99, ϵ = 1e-08, and bias correction.

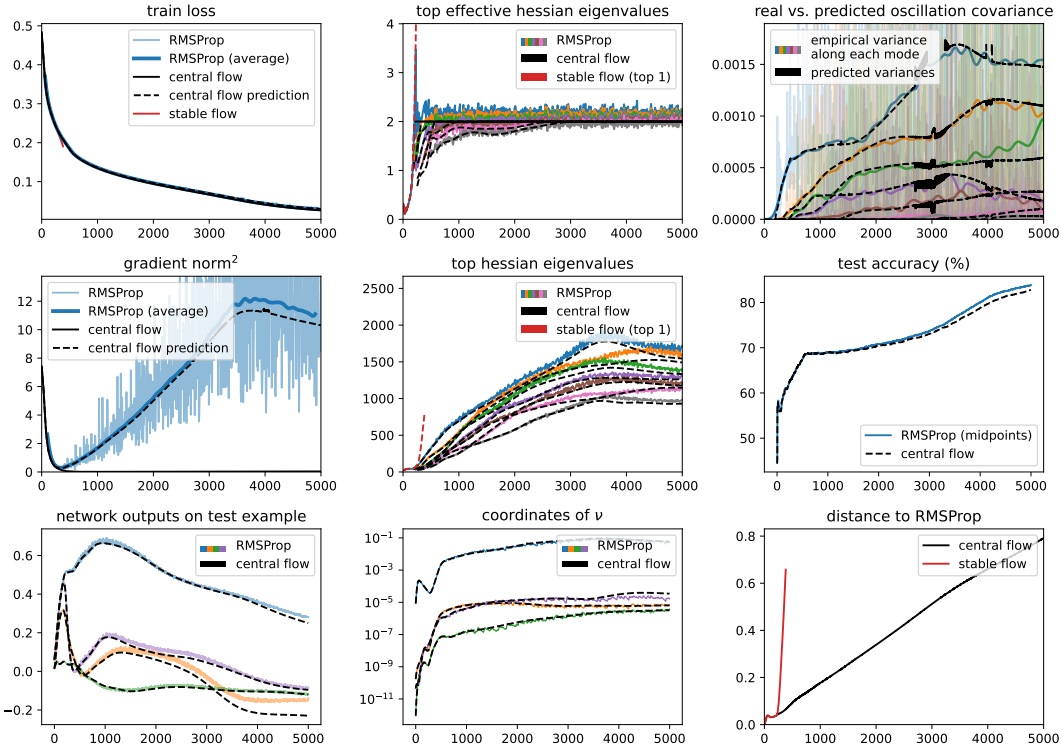


Figure 55.18: RMSProp central flow for a Mamba with MSE loss, η = 4e-05, β_2 = 0.99, ϵ = 1e-08, and bias correction.

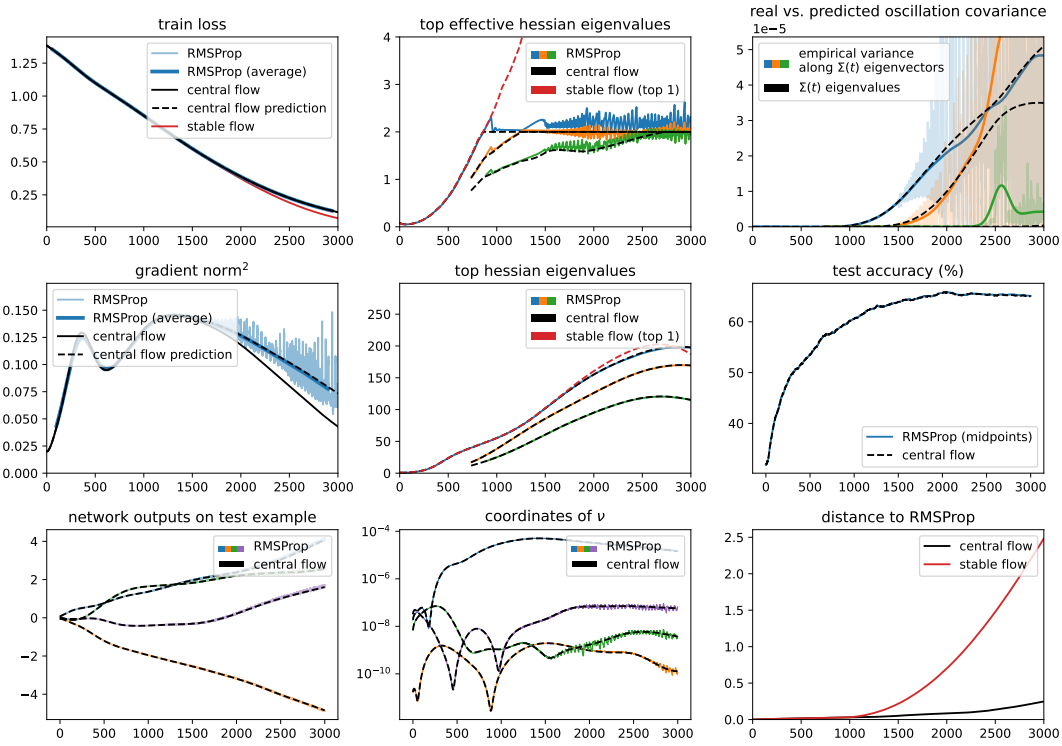


Figure 56.1: RMSProp central flow for a CNN with CE loss, $\eta =$ 7e-06, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

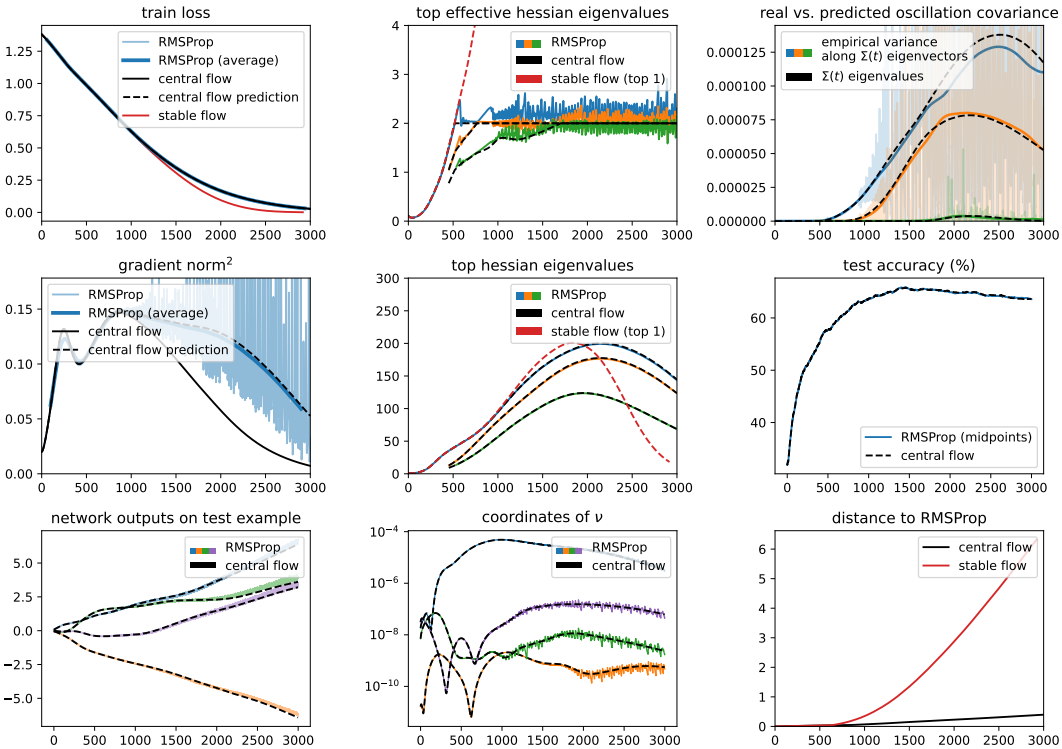


Figure 56.2: RMSProp central flow for a CNN with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

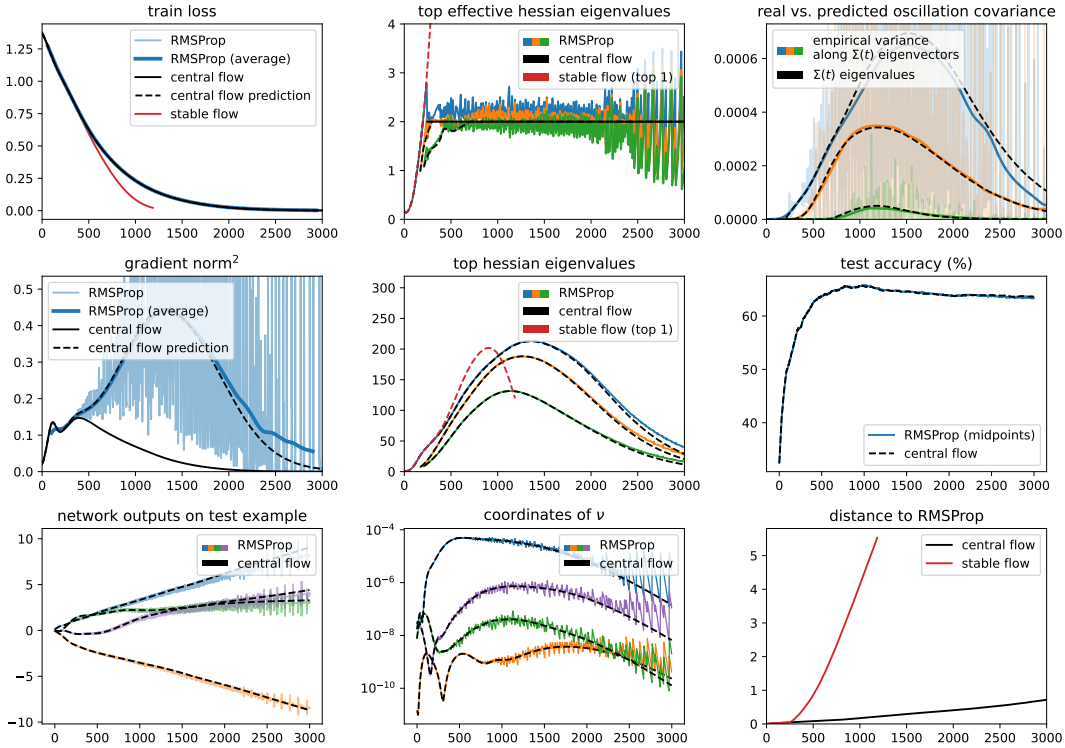


Figure 56.3: RMSProp central flow for a CNN with CE loss, $\eta =$ 2e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

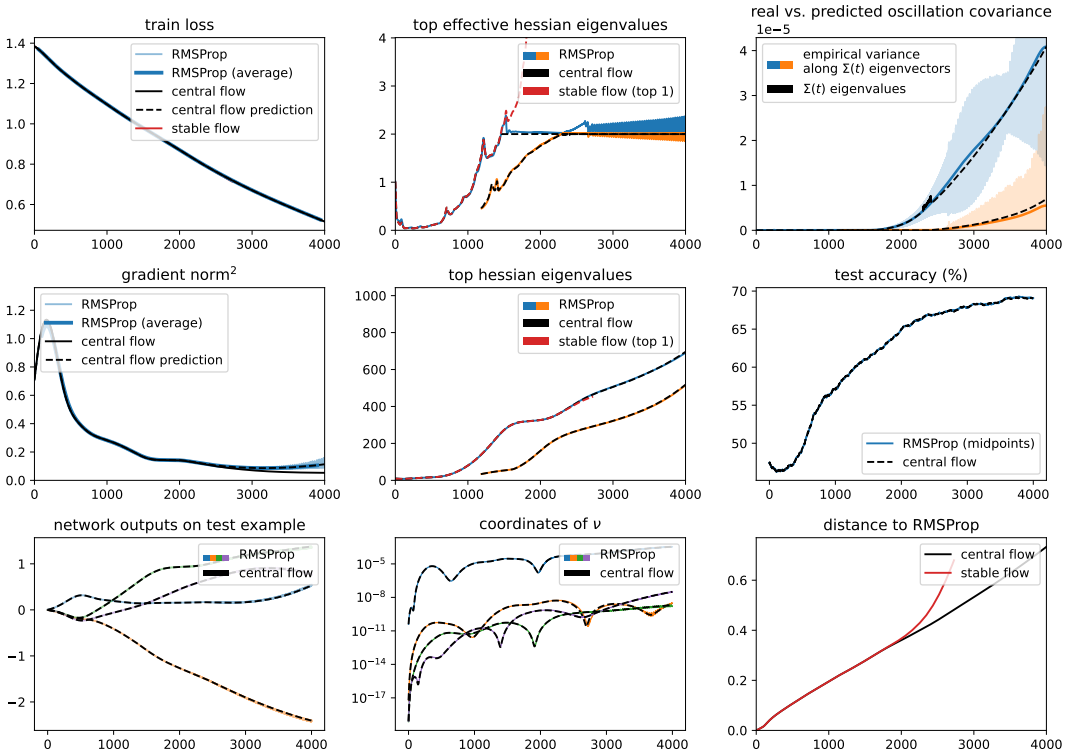


Figure 56.4: RMSProp central flow for a ResNet with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

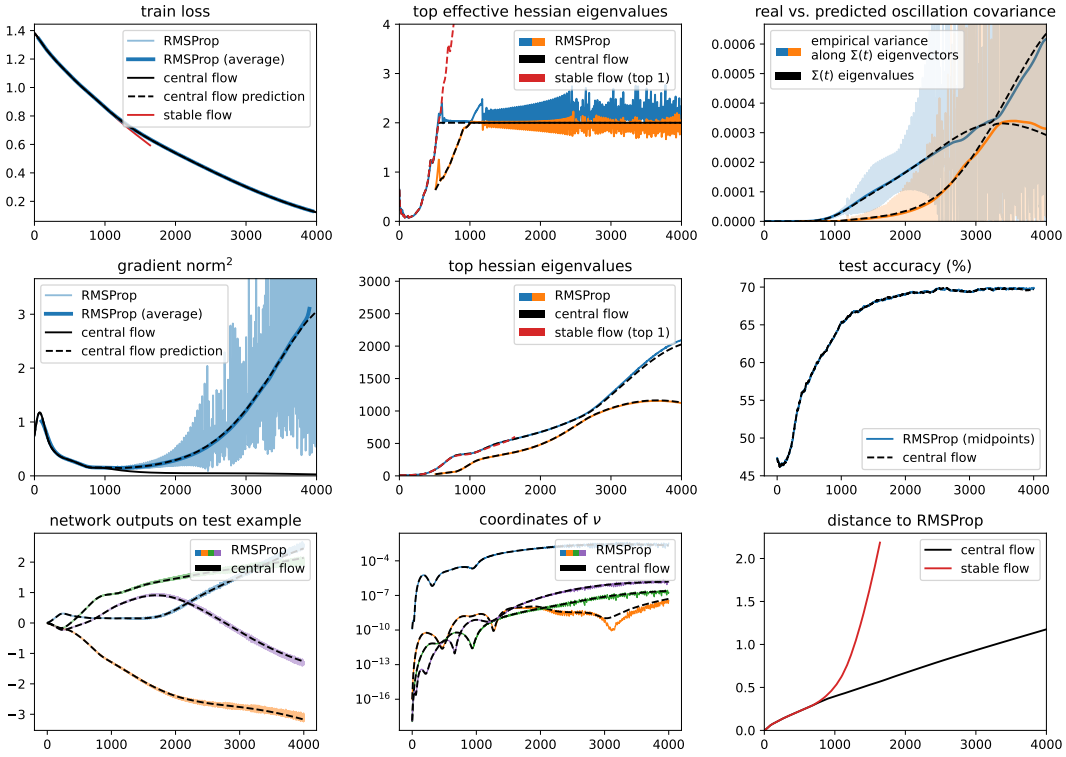


Figure 56.5: RMSProp central flow for a ResNet with CE loss, $\eta = 2\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

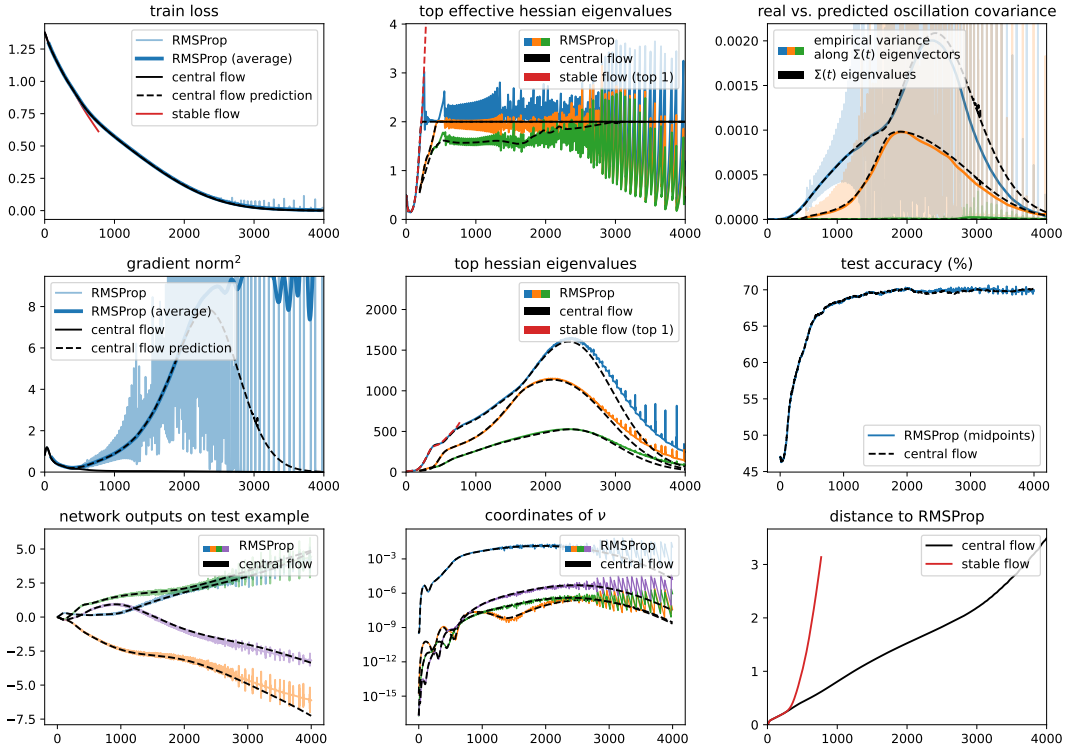


Figure 56.6: RMSProp central flow for a ResNet with CE loss, $\eta = 4\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

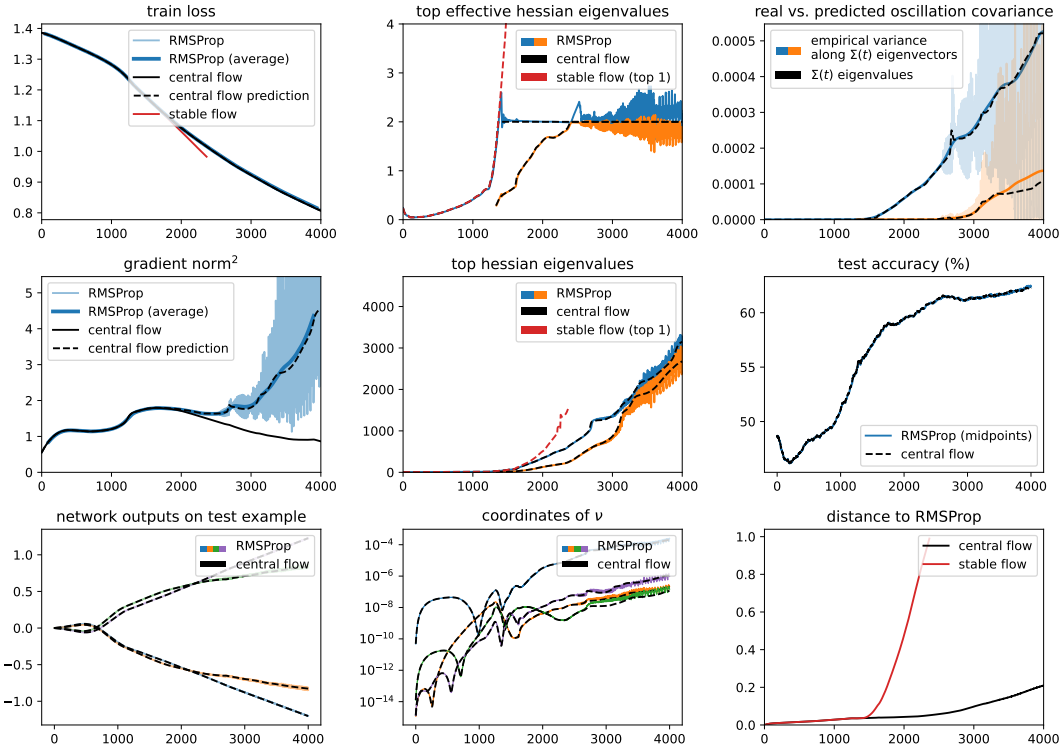


Figure 56.7: RMSProp central flow for a ViT with CE loss, $\eta = 5\text{e-}06$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

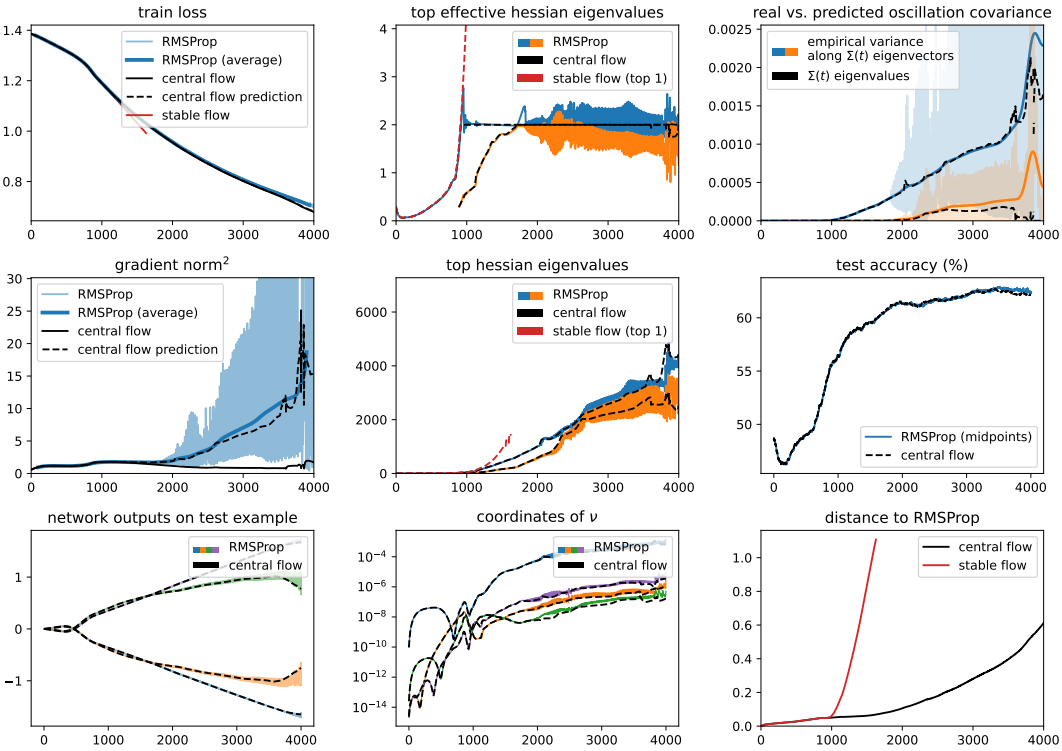


Figure 56.8: RMSProp central flow for a ViT with CE loss, $\eta = 7\text{e-}06$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

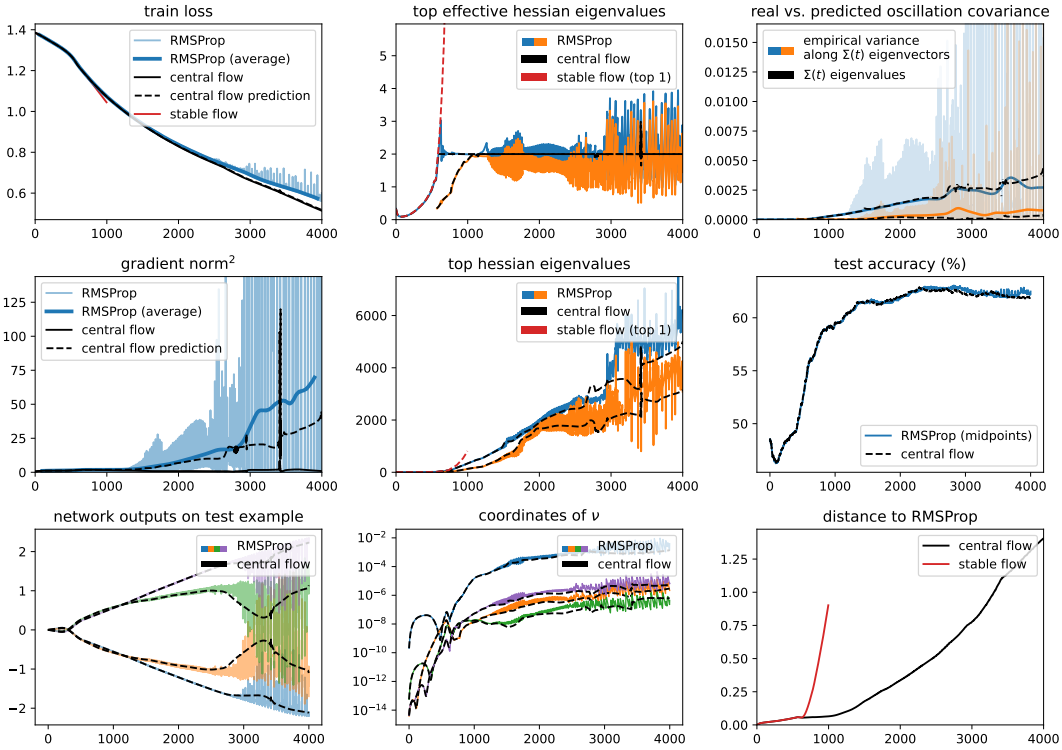


Figure 56.9: RMSProp central flow for a ViT with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

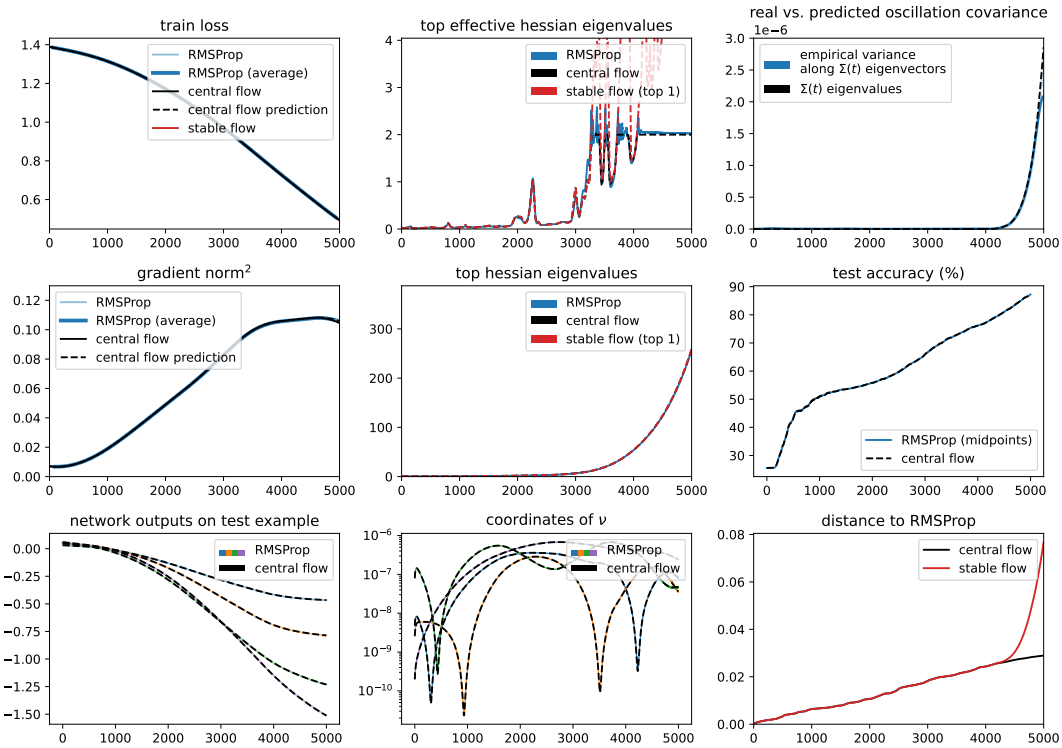


Figure 56.10: RMSProp central flow for a LSTM with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

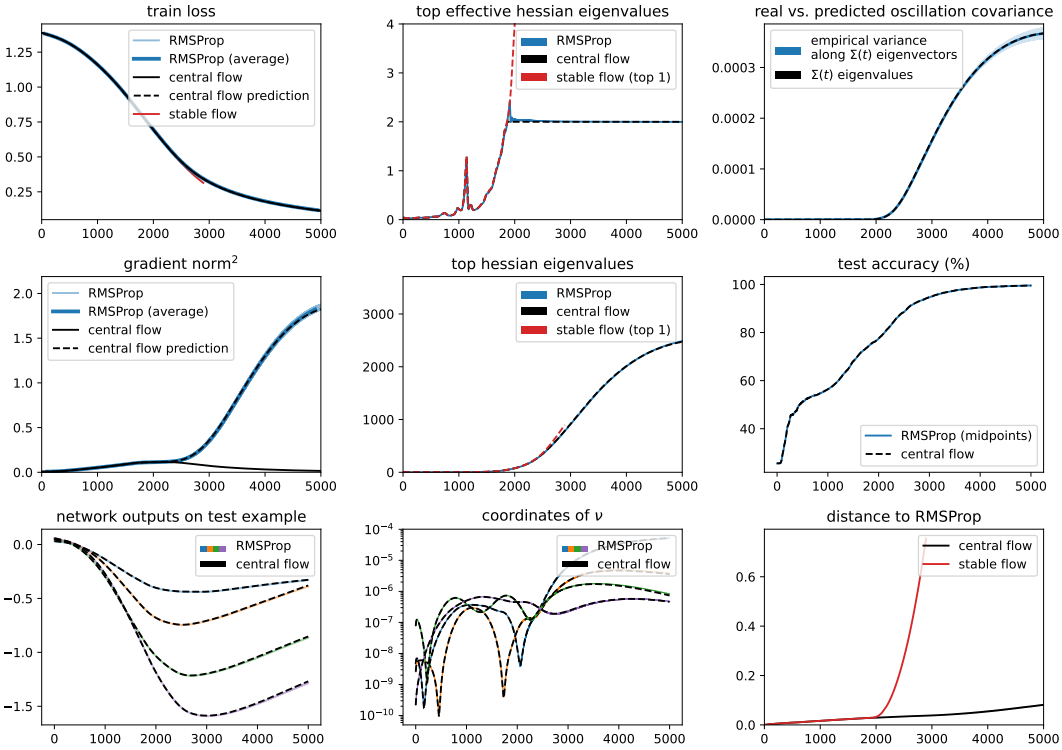


Figure 56.11: RMSProp central flow for a LSTM with CE loss, $\eta = 2\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

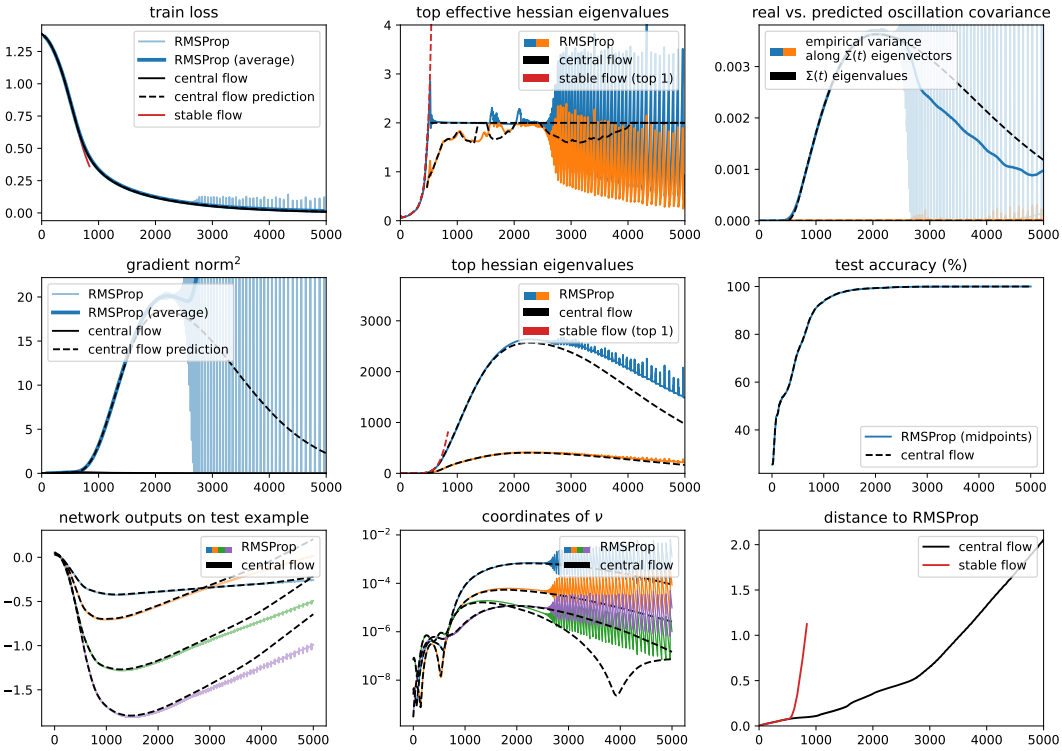


Figure 56.12: RMSProp central flow for a LSTM with CE loss, $\eta = 6\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

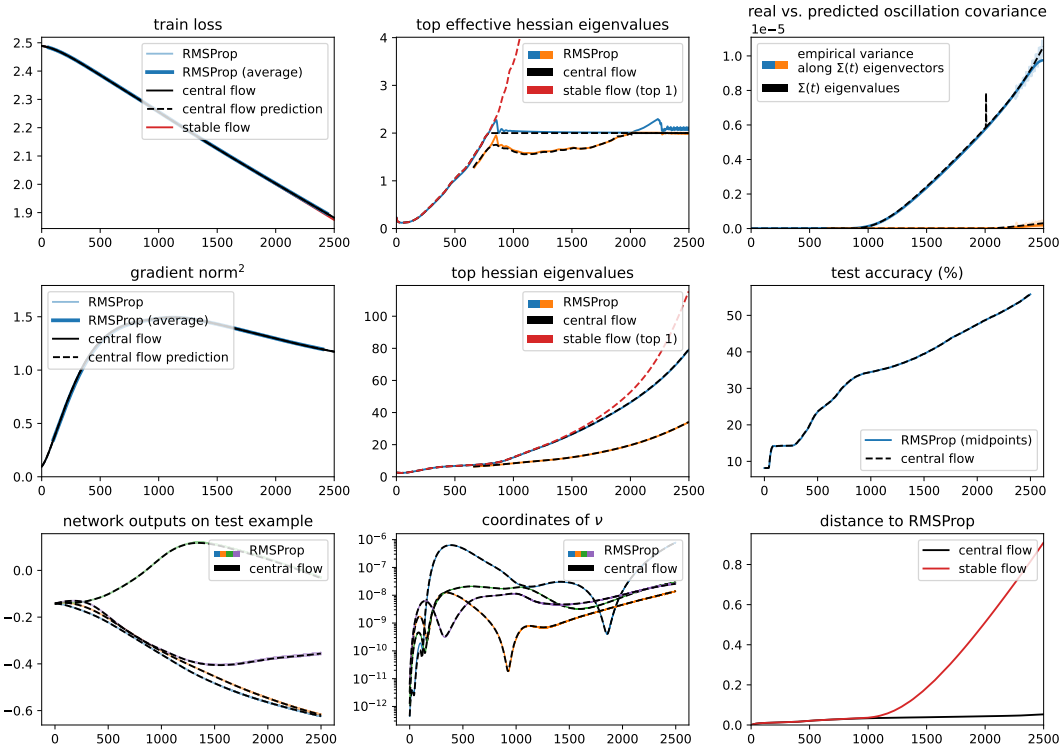


Figure 56.13: RMSProp central flow for a Transformer with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

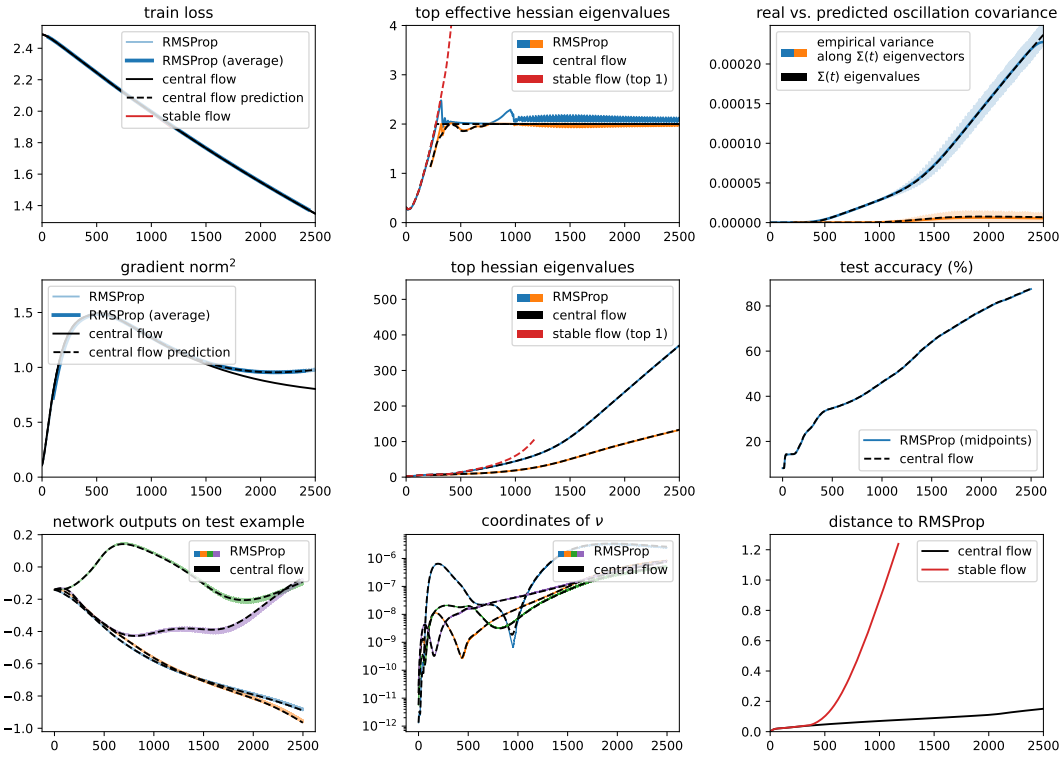


Figure 56.14: RMSProp central flow for a Transformer with CE loss, $\eta =$ 2e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

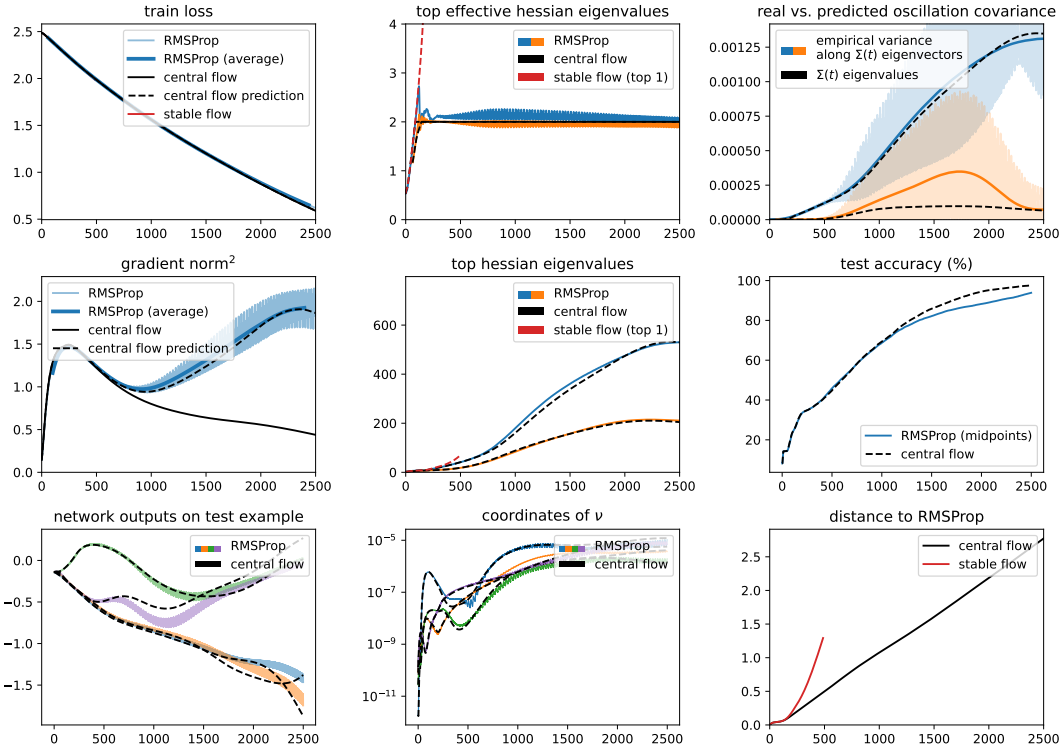


Figure 56.15: RMSProp central flow for a Transformer with CE loss, $\eta = 4\text{e-}05$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

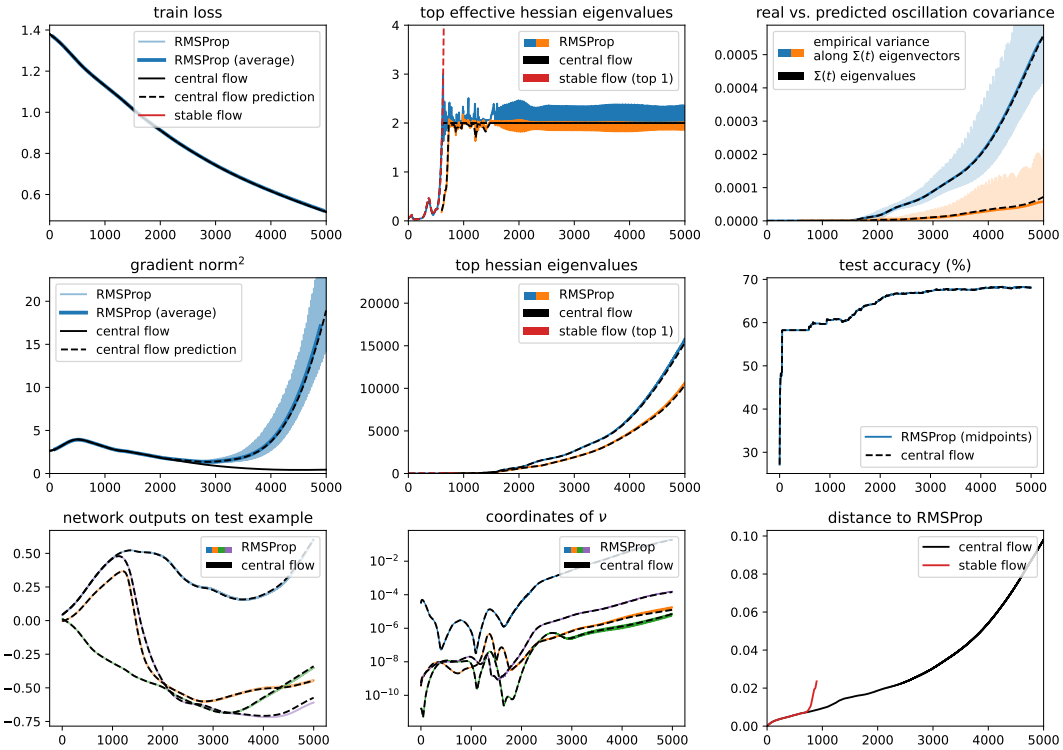


Figure 56.16: RMSProp central flow for a Mamba with CE loss, $\eta = 7\text{e-}06$, $\beta_2 = 0.95$, $\epsilon = 1\text{e-}08$, and bias correction.

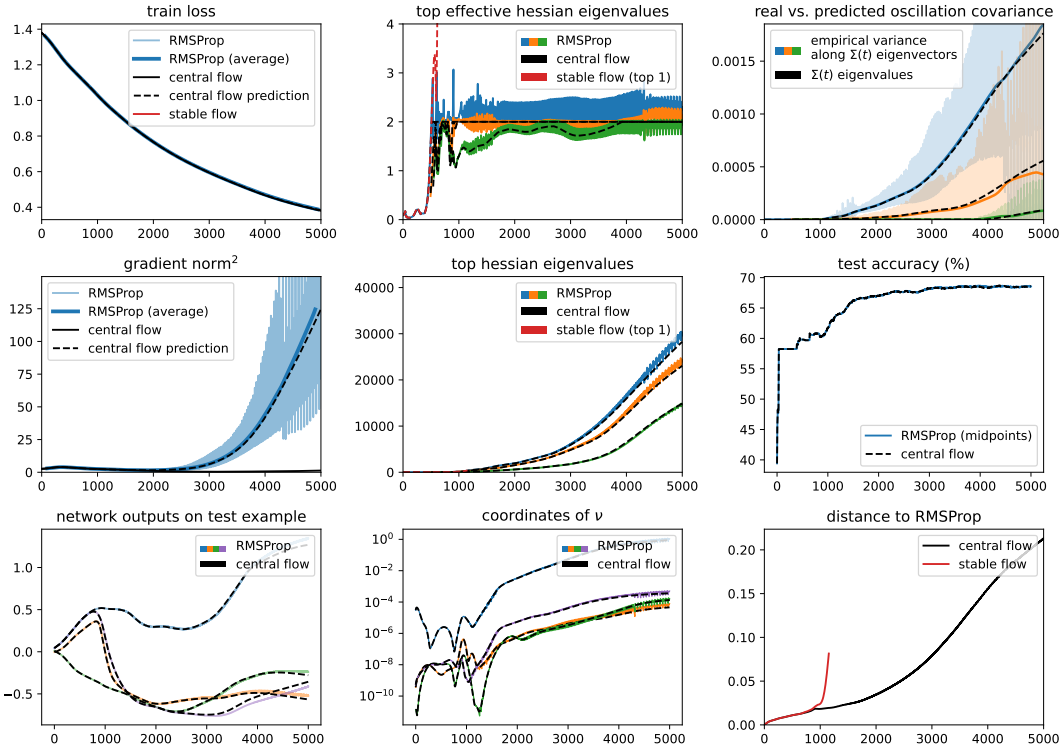


Figure 56.17: RMSProp central flow for a Mamba with CE loss, $\eta =$ 1e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.

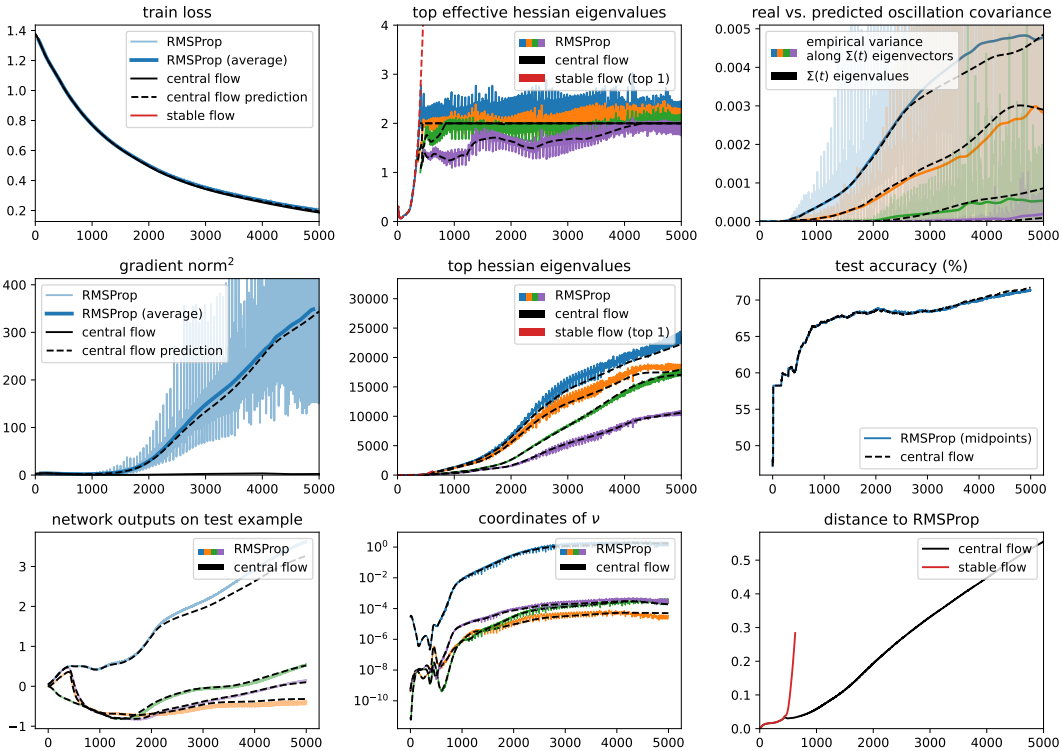


Figure 56.18: RMSProp central flow for a Mamba with CE loss, $\eta =$ 2e-05, $\beta_2 = 0.95$, $\epsilon =$ 1e-08, and bias correction.