UNDERSTANDING OPTIMIZATION IN DEEP LEARNING WITH CENTRAL FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimization in deep learning remains poorly understood, even in the simple setting of deterministic (i.e. full-batch) training. A key difficulty is that much of an optimizer's behavior is *implicitly* determined by complex oscillatory dynamics, referred to as the "edge of stability." The main contribution of this paper is to show that an optimizer's implicit behavior can be explicitly captured by a *central flow*: a differential equation which models the time-averaged optimization trajectory. We show that these flows can empirically predict long-term optimization trajectories of generic neural networks with an unprecedentedly high degree of numerical accuracy. By interpreting these flows, we reveal for the first time 1) the precise sense in which RMSProp adapts to the local loss landscape, and 2) an *acceleration via regularization* mechanism, wherein adaptive optimizers implicitly navigate towards low-curvature regions in which they can take larger steps. This mechanism is key to the efficacy of these adaptive optimizers. Overall, we believe that central flows constitute a promising tool for reasoning about optimization in deep learning.

1 INTRODUCTION

Optimization in deep learning remains poorly understood, even in the simple setting of deterministic (i.e. full-batch) training. A key difficulty is that much of an optimizer's behavior is determined *implicitly* by complex oscillatory dynamics (Xing et al., 2018; Jastrzębski et al., 2019; 2020; Cohen et al., 2021). As a result, an optimizer's update rule often sheds little light on its actual behavior.

To resolve this difficulty, we show that an optimizer's implicit behavior can be characterized *explicitly* by a *central flow*: a differential equation that models the time-averaged optimization trajectory. The central flow averages out the oscillations themselves, but retains their lasting effect on the trajectory. This lasting effect manifests as an *implicit reduction in the curvature* of the loss function.[1] Accordingly, the central flow corresponding to each optimization algorithm takes the form of a curvature-penalized gradient flow. We derive these central flows using informal mathematical reasoning, and empirically show that **they can successfully predict long-term optimization trajectories of neural networks with a high degree of numerical accuracy**. We are unaware of any other theoretical analyses of optimization in generic deep learning settings with a similar degree of predictive power.

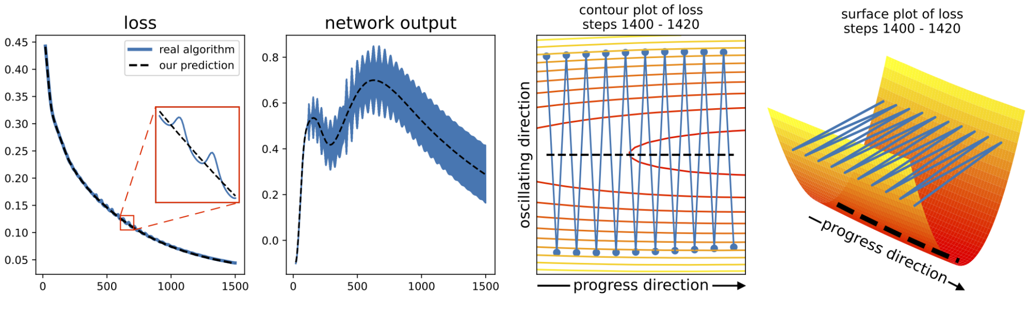


Figure 1: **The central flow models the time-averaged trajectory of the oscillatory optimizer**. We train a CNN on CIFAR-10 using full-batch RMSProp (blue). The optimizer oscillates in weight space (right two plots), which causes the loss, and the network's output on a test example, to oscillate (left two plots). Our central flow (black) directly models the time-averaged, or smoothed, trajectory.

Unlike an optimizer's update rule, the central flow directly exposes the optimizer's behavior, thereby making it possible to reason about optimization. We derive central flows for gradient descent (Section 3), a simple adaptive optimizer (Section 4), and RMSProp (Section 5), and we use these central flows to understand the behavior of the corresponding algorithms. For example, our analysis reveals, for the first time, **the precise sense in which RMSProp "adapts" its effective step sizes to the local loss landscape**. It also shows that the two adaptive optimizers implicitly regularize the curvature along their trajectories, giving rise to a mechanism we call *acceleration via regularization*, whereby implicitly regularizing curvature drives the trajectory towards low-curvature regions where the optimizer can take larger steps. **This mechanism is key to the efficacy of these optimizers.**

We are optimistic that our methodology holds promise as a framework for analyzing, and perhaps even inventing, deep learning optimization algorithms beyond the ones studied here.

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 & Bruna, 2023; Even et al., 2024; Kreisler et al., 2023; Song & Yun, 2023; Li et al., 2022; 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 analysis is most directly inspired by Damian et al. (2023), which rigorously analyzed EOS in the special case where gradient descent oscillates along a single direction. Whereas they analyze the *fine-grained* oscillatory dynamics, we argue that analyzing the *time-averaged* dynamics is simpler, and is sufficient for many purposes. We first reproduce their main result using a simple, albeit non-rigorous, time-averaging argument. We then show that this time-averaging methodology easily extends to the more realistic and challenging setting where gradient descent oscillates along *multiple* directions simultaneously, as well as to the analysis of two adaptive optimizers.

Understanding Adaptive Optimizers **?** 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 & 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 & Lin, 2024; Zhang et al., 2024; Wang et al., 2024c; Hübler et al., 2024).

3 GRADIENT DESCENT

We introduce our framework by analyzing the simplest first-order optimizer: gradient descent with a fixed step size η. This analysis will set the stage for our analyses of more complex optimizers.

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

Why does gradient descent work? The classical explanation requires the learning rate η to be set small relative to the curvature of the loss function, which is treated as an *a priori* property of the

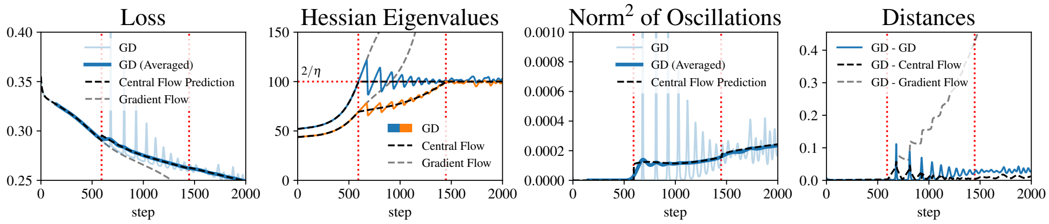


Figure 2: **Central flow for gradient descent.** A ViT is trained on CIFAR-10 using gradient descent with $\eta = 2/100$ (blue). Gradient descent enters EOS at step 600 and after step 1450 multiple eigenvalues are unstable (dotted red). The central flow (black) accurately models gradient descent even at EOS, whereas gradient flow (gray) follows a different path. The distance between the central flow and gradient descent (black) even remains smaller than the distance between consecutive iterates of gradient descent (blue) throughout the trajectory.

optimization problem. However, recent works have demonstrated that this explanation does not apply to deep learning. In deep learning, gradient descent converges not because the curvature is "already small," but rather because the optimizer *automatically avoids* high curvature regions of the loss landscape (Cohen et al., 2021; Damian et al., 2023). In this section, we will analyze these dynamics, with the goal of rendering *explicit* the path that is *implicitly* taken by gradient descent.

3.1 THE MECHANICS OF GRADIENT DESCENT

The dynamics of gradient descent in deep learning revolve around the *sharpness* $S(w) := \lambda_1(H(w))$, defined as the largest eigenvalue of the Hessian $H(w)$. There are three important properties of the sharpness: (1) the sharpness tends to rise during training, (2) gradient descent oscillates whenever the sharpness exceeds the *critical threshold* $2/\eta$, (3) such oscillations implicitly reduce the sharpness. The interactions between these three processes give rise to rich dynamics, termed *edge of stability*, which automatically keep the sharpness regulated around the critical threshold $2/\eta$.

1. Sharpness tends to rise A robust empirical observation of Cohen et al. (2021), dubbed *progressive sharpening*, is that the sharpness tends to rise during the training of neural networks.

2. High sharpness triggers oscillations A local quadratic Taylor expansion reveals that gradient descent oscillates whenever the sharpness exceeds the critical threshold $2/\eta$. For example, consider optimizing a one-dimensional quadratic objective $L(x) = \frac{1}{2}Sx^2$, which has global sharpness S. The gradient descent iterates $\{x_t\}$ evolve via $x_{t+1} = (1 - \eta S)x_t$. If $S > 2/\eta$, 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. More generally, for a quadratic objective in multiple dimensions, gradient descent oscillates with exponentially growing magnitude along all Hessian eigenvectors with eigenvalues exceeding $2/\eta$. While deep learning objectives are not globally quadratic, a *local* quadratic Taylor approximation suggests that in any region of weight space where the sharpness exceeds $2/\eta$, gradient descent will oscillate along the highest-curvature direction(s).

3. Oscillations reduce sharpness When high sharpness triggers oscillations, a local *cubic* Taylor approximation reveals that these oscillations in turn trigger a reduction in sharpness (Damian et al., 2023), a form of negative feedback which can prevent divergence. Suppose that gradient descent is oscillating around a reference point $\overline{w}$, along the top Hessian eigenvector u, with current magnitude x, so that $w = \overline{w} + xu$. Then Taylor-expanding $\nabla L(w)$ around $\overline{w}$ gives (see Lemma 1):

$$\nabla L(w) = \underbrace{\boxed{\nabla L(\overline{w})}}_{\text{(1) gradient at reference point}} + \underbrace{\boxed{xS(\overline{w})u}}_{\text{(2) oscillation}} + \underbrace{\boxed{\tfrac{1}{2}x^2\nabla S(\overline{w})}}_{\text{(3) sharpness reduction}} + \mathcal{O}(x^3) \qquad (1)$$

The third term, which arises from the cubic term in the Taylor expansion of the loss, reveals that a gradient step on the loss with step size η automatically includes a gradient step on the *sharpness* of the loss with step size $\frac{1}{2}\eta x^2$. Thus, oscillations automatically trigger reduction of sharpness. **This fact, not accounted for by classical theory, is necessary for understanding the dynamics of gradient descent in deep learning.** A similar argument extends to the case where gradient descent oscillates simultaneously along multiple top Hessian eigenvectors.

The interactions between these three processes give rise to rich dynamics, termed *edge of stability*, in which the algorithm oscillates along the highest-curvature direction(s) without diverging, and the sharpness stays automatically regulated around the critical threshold $2/\eta$ (Cohen et al., 2021). In the special case where only the largest eigenvalue crosses $2/\eta$ (e.g. steps 600 – 1450 in Figure 2), which was rigorously analyzed in Damian et al. (2023), the dynamics consist of repeated cycles in which: (a) progressive sharpening drives the sharpness above $2/\eta$; (b) this triggers growing oscillations along the top Hessian eigenvector; (c) such oscillations force the sharpness back below $2/\eta$; (d) the oscillations consequently shrink in magnitude. When multiple Hessian eigenvalues have reached $2/\eta$ (e.g. steps 1450 – 2000 in Figure 2), gradient descent oscillates simultaneously along all the corresponding eigenvectors, and all such eigenvalues are dynamically regulated around $2/\eta$.

While the *fine-grained* EOS dynamics are complex, we will now show that a simple *time-averaging* argument not only recovers the analysis of Damian et al. (2023) (albeit non-rigorously), but also generalizes to the more realistic and challenging setting of multiple oscillating directions.

3.2 DERIVING THE GD CENTRAL FLOW

So long as the sharpness $S(w_t)$ remains below the critical threshold $2/\eta$, gradient descent is said to be *stable*. While gradient descent is stable, it does not exhibit sustained oscillations, and its trajectory is empirically well-approximated[2] by that of gradient flow:[3] $\frac{dw}{dt} = -\eta \nabla L(w)$; however, once gradient descent enters the EOS regime, its trajectory rapidly departs from that of gradient flow (Cohen et al., 2021). We will now derive a more general ODE, which we call a *central flow*, which models the *time-averaged* trajectory of gradient descent in both the stable and EOS regimes. While our derivation is not rigorous, we will demonstrate that the central flow accurately predicts long-term gradient descent trajectories in a variety of neural network settings.

Throughout this paper, we abuse notation and use $\mathbb{E}$ to denote "local time-averages" of deterministic quantities — see Appendix C.2 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 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 $2/\eta$. In this setting, gradient descent oscillates along a single direction — the top Hessian eigenvector. We will therefore model the GD trajectory by $w_t = \overline{w}_t + x_t u_t$ where w_t is the true 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 direction. Note that by definition, $\mathbb{E}[x_t] = 0$, i.e. the time-averaged displacement is zero. To track the evolution of $\overline{w}_t$, we time-average both sides of the gradient descent update and Taylor expand the time-averaged gradient[4] using eq. (1):

$$\overline{w}_{t+1} \;=\; \overline{w}_t - \eta\,\mathbb{E}[\nabla L(w_t)] \;\approx\; \overline{w}_t - \eta\big[\nabla L(\overline{w}_t) + \cancel{S(\overline{w}_t)\,\mathbb{E}[x_t]u_t} + \tfrac{1}{2}\,\mathbb{E}[x_t^2]\nabla S(\overline{w}_t)\big].$$

Note that the second term in the time-averaged gradient is 0 because $\mathbb{E}[x_t] = 0$. Therefore, we model the time-averaged iterates $\overline{w}_t$ by the sharpness-penalized gradient flow $w(t)$ defined by:

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

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 empirically observed dynamics. In particular, so long as gradient descent is at the edge of stability, the sharpness stays dynamically regulated around $2/\eta$. Therefore, we will enforce that the central flow keeps the sharpness *fixed* at $S(w(t)) = 2/\eta$. This implies that along the central flow, $\frac{dS(w)}{dt} = 0$.

The time derivative of the sharpness under eq. (2) can be easily computed using the chain rule:

$$\frac{dS(w)}{dt} = \left\langle \nabla S(w), \frac{dw}{dt} \right\rangle = \underbrace{\eta\,\langle \nabla S(w), -\nabla L(w)\rangle}_{\text{progressive sharpening}} - \underbrace{\tfrac{1}{2}\eta\sigma^2(t)\|\nabla S(w)\|^2}_{\text{sharpness reduction from oscillations}} \tag{3}$$

Solving for $\frac{dS(w)}{dt} = 0$ yields the unique $\sigma^2(t)$ that keeps the sharpness fixed in place:

$$\sigma^2(t) = \frac{2\left\langle \nabla S(w), -\nabla L(w) \right\rangle}{\left\| \nabla S(w) \right\|^2}. \tag{4}$$

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 is locked at $2/\eta$. The central flow for a single unstable eigenvalue is given by substituting this $\sigma^2(t)$ into eq. (2).

3.2.2 THE GENERAL CASE (MULTIPLE UNSTABLE EIGENVALUES)

We now generalize this analysis to the setting where multiple eigenvalues have reached the critical threshold $2/\eta$, and gradient descent oscillates in the span of the corresponding eigenvectors. We assume the displacement $\delta_t := w_t - \overline{w}_t$ between the true process and the time-averaged process lies in the span of these eigenvectors. For example, when only one eigenvalue has reached $2/\eta$, taking $\delta_t = x_t u_t$ recovers the analysis in Section 3.2.1. The iterate w_t can be decomposed as $w_t = \overline{w}_t + \delta_t$ where, by definition, $\mathbb{E}[\delta_t] = 0$. As above, we Taylor-expand the gradient around $\overline{w}_t$:

$$\nabla L(w_t) \approx \nabla L(\overline{w}_t) + H(\overline{w}_t)\delta_t + \tfrac{1}{2}\nabla_{\overline{w}_t} \left\langle H(\overline{w}_t), \delta_t \delta_t^T \right\rangle. \tag{5}$$

The third term in this Taylor expansion can be interpreted as the gradient of the curvature in the δ_t direction.[5] To track the evolution of the time-averaged iterate $\overline{w}_t$, we time-average both sides of the gradient descent update and substitute in the Taylor-expansion in eq. (5):

$$\overline{w}_{t+1} \;=\; \overline{w}_t - \eta\, \mathbb{E}[\nabla L(w_t)] \;\approx\; \overline{w}_t - \eta \Big[\nabla L(\overline{w}_t) + \cancel{H(\overline{w}_t)\mathbb{E}[\delta_t]} + \nabla_{\overline{w}_t} \left\langle H(\overline{w}_t), \mathbb{E}[\delta_t \delta_t^T] \right\rangle \Big].$$

As above, the second term in the time-averaged gradient is 0 because $\mathbb{E}[\delta_t] = 0$. Therefore we model the time-averaged iterates $\overline{w}_t$ by the curvature-penalized gradient flow $w(t)$ defined by:

$$\frac{dw}{dt} = -\eta \Big[\nabla L(w) + \underbrace{\tfrac{1}{2}\nabla_w \left\langle H(w), \Sigma(t) \right\rangle}_{\text{implicit curvature penalty}} \Big]. \tag{6}$$

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. As above, we will argue that only one value of $\Sigma(t)$ is consistent with the empirically observed dynamics. In experiments, Hessian eigenvalues which have reached the critical threshold $2/\eta$ do not continue to rise further; rather, these eigenvalues either remain dynamically regulated around $2/\eta$, staying at the edge of stability; or they drop below $2/\eta$, leaving the edge of stability (Cohen et al., 2021). Thus, we impose the desideratum that the flow eq. (6) should not increase any Hessian eigenvalues beyond $2/\eta$. To this, we add the two natural desidirata that: (1) as a covariance matrix, $\Sigma(t)$ should be PSD, and (2) gradient descent should not be modeled as oscillating along Hessian eigenvectors with eigenvalues less than $2/\eta$. These three desiderata turn out to imply a unique value of $\Sigma(t)$. In particular, we detail in Appendix C.3 that these three desiderata imply that $\Sigma(t)$ is the solution to a type of convex program known as a *cone complementarity problem* (CCP). The central flow for gradient descent is defined as eq. (6) with this value of $\Sigma(t)$.

Experimental validation Although this derivation employs informal mathematical reasoning, our experiments demonstrate that this central flow can successfully predict long-term optimization trajectories of neural networks with a high degree of numerical accuracy. For example, in Figure 2, we run gradient descent side by side with the central flow, and monitor the distance between the two trajectories in weight space. Observe that the weight space distance between these two trajectories stays very close to zero (it is even less than the distance between successive gradient descent iterates). By contrast, the distance between the gradient descent and gradient *flow* trajectories grows large. Many of our experiments attain a similarly high level of numerical accuracy. Even in cases where weight-space distance grows non-negligibly during training, the central flow's predictions for derived quantities such as the train loss, or individual network outputs, often still closely match those of the real optimizer trajectory. **We are unaware of any other theoretical framework for reasoning about optimization in deep learning with a comparable level of predictive power**.

Appendix B shows our full set of experiments, and discusses the factors that affect the quality of the central flow approximation. A promising direction for future research is to rigorously identify conditions under which the central flow accurately approximates the real optimization trajectory.

3.3 INTERPRETING GRADIENT DESCENT VIA ITS CENTRAL FLOW

Being a smooth flow, the central flow is an simpler object to reason about than the oscillatory gradient descent trajectory. For example, the training loss $L(w_t)$ along the gradient descent trajectory behaves non-monotonically, as shown in Figure 3. This led Cohen et al. (2021) to observe that there was no explanation for why the loss still decreases over longer timescales. The central flow provides a simple explanation for this phenomenon.[6] Because the central flow is a smooth curve, the chain rule quantifies the rate of loss decrease: $\frac{dL(w)}{dt} = \langle \nabla L(w), \frac{dw}{dt} \rangle$. Using this, we prove in Proposition 1 that $\frac{dL(w)}{dt} \leq 0$, i.e. that the central flow loss $L(w(t))$ is monotonically decreasing. Thus, the central flow loss is a **hidden progress metric** for the optimization process.

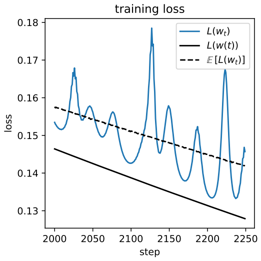


Figure 3: Train loss curve

As visible in Figure 3, the gradient descent train loss $L(w_t)$ is generally higher than the central flow train loss $L(w(t))$, due to the oscillations. However, since the central flow also models the oscillation covariance $\Sigma(t)$, it can render predictions for the *time-averaged* train loss of the algorithm:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \tfrac{1}{2} \langle H(w(t)), \Sigma(t) \rangle = \underbrace{L(w(t))}_{\text{central flow loss}} + \underbrace{\tfrac{1}{2} S(w(t)) \operatorname{tr}(\Sigma(t))}_{\text{effect of oscillations}}. \tag{7}$$

Thus, the central flow allows us to decompose the time-averaged training loss curve into the loss along the central flow, which decreases monotonically, plus a contribution from the oscillations.

4 RMSPROP-NORM

We now study "RMSProp-Norm", a simplification of RMSProp which uses one global adaptive step size rather than separate adaptive step sizes for each coordinate:

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

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*.[7][8] The EMA hyperparameter β_2 is a knob that interpolates the algorithm between gradient descent when $\beta_2 = 1$ and normalized gradient descent (NGD) when $\beta_2 = 0$.[9] Because RMSProp-Norm adapts only the step size, it is a simpler analogue of more complex adaptive optimizers which adapt the update *direction* as well. As such, its analysis will be a stepping stone to that of more complex methods.

We will explicitly characterize how RMSProp-Norm adapts its effective step size to the local loss landscape, revealing how oscillatory dynamics enable a *gradient*-based optimizer to adapt to the local *curvature*. We also will show that oscillations implicitly regularize curvature, giving rise to a mechanism we call *acceleration via regularization* (Section 4.3.3), whereby implicitly regularizing curvature drives the trajectory towards low-curvature regions where the algorithm can take larger steps. This mechanism is key to the efficacy of RMSProp-Norm and the function of its hyperparameters.

4.1 THE MECHANICS OF RMSPROP-NORM

The dynamics of RMSProp-Norm revolve around the *effective sharpness*, defined as $S^{\text{eff}} := S(w)/\sqrt{\nu}$. First, the effective sharpness controls the oscillations: when $S^{\text{eff}} > 2/\eta$, RMSProp-Norm 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 Equation (1), thereby decreasing the effective sharpness via its *numerator*. The other mechanism, new to RMSProp-Norm, 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 $2/\eta$, as depicted in the bottom left plot in Figure 4. 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.

4.2 DERIVING THE CENTRAL FLOW

Recall that while gradient descent trains stably, it is well-approximated by gradient flow. One can derive an analogous "stable flow" for RMSProp-Norm (Ma et al., 2022, cf.):[10]

$$\tfrac{d}{dt} w(t) = -\tfrac{\eta}{\sqrt{\nu(t)}} [\nabla L(w(t))], \quad \tfrac{d}{dt} \nu(t) = \tfrac{1-\beta_2}{\beta_2} \left[\|\nabla L(w(t))\|^2 - \nu(t) \right]. \tag{8}$$

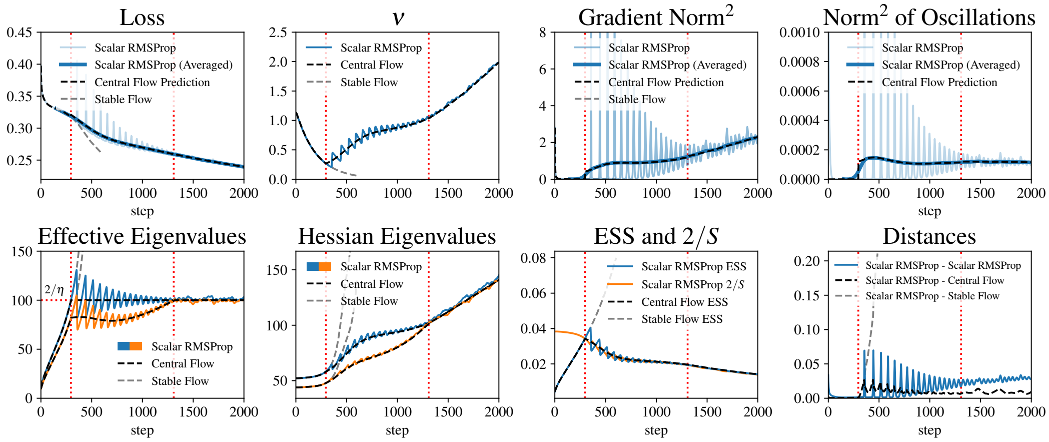


Figure 4: **Central flow for RMSProp-Norm.** A ViT is trained on a subset of CIFAR-10 using RMSProp-Norm with $\eta = 2/100$ and $\beta_2 = 0.995$ (blue). It enters EOS around step 300, and around step 1300 multiple eigenvalues go unstable (dotted red). The central flow (black) accurately models the time-averaged trajectory of RMSProp-Norm even at the edge of stability, whereas the naive stable flow (gray) follows a different path. The effective learning rate (ELR) plot shows that after reaching EOS, the ELR equilibrates around the *maximal locally stable learning rate* $2/S$.

However, at the edge of stability, the trajectory of RMSProp-Norm deviates from eq. (8). 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 unstable.

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. (1), we can also derive a time-averaged approximation for $\mathbb{E}[\|\nabla L(w)\|^2]$:

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

where we again used $\mathbb{E}[x] = 0$ to ignore the middle term. 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:

$$\begin{aligned} \tfrac{d}{dt} w(t) &= -\frac{\eta}{\sqrt{\nu(t)}}\Big[\underbrace{\nabla L(w(t)) + \tfrac{1}{2}\sigma^2(t)\nabla S(w(t))}_{\mathbb{E}[\nabla L(w_t)]}\Big] \\ \tfrac{d}{dt}\nu(t) &= \tfrac{1-\beta_2}{\beta_2}\Big[\overbrace{\|\nabla L(w(t))\|^2 + S(w(t))^2\sigma^2(t)}^{\mathbb{E}[\|\nabla L(w_t)\|^2]} - \nu(t)\Big] \end{aligned} \tag{9}$$

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 $S^{\text{eff}}(w, \nu) = 2/\eta$. 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. (9) 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:

$$\sigma^2(w; \eta, \beta_2) = \frac{\beta_2 \overbrace{\langle -\nabla L(w), \nabla S(w)\rangle}^{\text{progressive sharpening}} + (1-\beta_2)\overbrace{\left[S(w)^2/4 - \|\nabla L(w)\|^2/\eta^2\right]}^{\text{effect of mean reversion on }\nu}}{\beta_2 \underbrace{\tfrac{1}{2}\|\nabla S(w)\|^2}_{\text{sharpness reduction}} + (1-\beta_2)\underbrace{S(w)^2/\eta^2}_{\text{effect of oscillation on }\nu}}. \tag{10}$$

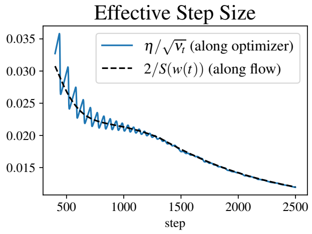
The central flow for RMSProp-Norm with a single unstable eigenvalue is given by eq. (9) with this value of σ^2. In Appendix C.4 we extend this flow to the case of multiple oscillating directions. In Figure 4 and Appendix D we validate this flow empirically.

4.3 INTERPRETING RMSPROP-NORM VIA ITS CENTRAL FLOW

We now interpret the RMSProp-Norm 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.[11]

4.3.1 IMPLICIT STEP SIZE SELECTION

The central flow renders *explicit* the step size strategy that is *implicit* in the oscillatory dynamics of RMSProp-Norm. Recall that while the central flow is at EOS, the effective sharpness $S^{\text{eff}} := S(w)/\sqrt{\nu}$ is fixed at $2/\eta$. This condition can be rearranged into a statement about the effective learning rate: $\eta/\sqrt{\nu} = 2/S(w)$. Notably, the value $2/S(w)$ is the *maximal locally stable step size* for the current location w in weight space. In other words, while the algorithm is at EOS, the oscillatory dynamics continually adapt the effective step size to the current maximal stable step size. This is the precise sense in which RMSProp-Norm "adapts" to the local loss landscape.



4.3.2 IMPLICIT CURVATURE REDUCTION

Understanding the implicit step size strategy employed by RMSProp-Norm 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 RMSProp-Norm 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:[12]

$$\frac{dw}{dt} = -\underbrace{\frac{2}{S(w)}}_{\text{effective step size}}\Big[\nabla L(w) + \underbrace{\tfrac{1}{2}\sigma^2(w;\eta,\beta_2)\nabla S(w)}_{\text{implicit sharpness penalty}}\Big] \tag{11}$$

where $\sigma^2(w;\eta,\beta_2)$ is given by eq. (10). In other words, the time-averaged trajectory of RMSProp-Norm 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. (10).

Notably, the hyperparameters η, β_2 are not used to determine the effective step size. 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. (10) is increasing in η while the denominator is decreasing in η, which implies the overall expression for σ^2 is increasing in η.[13] Meanwhile, the effect of the hyperparameter β_2 is to monotonically interpolate σ^2 between that of NGD when $\beta_2 = 0$ and that of gradient descent when $\beta_2 = 1$.[14] The interpretations of η, β_2 generalize to the setting of multiple oscillating directions, as detailed in Lemma 3.

4.3.3 ACCELERATION VIA REGULARIZATION

To fully grasp the *modus operandi* of RMSProp-Norm, it is necessary to consider the link between step size adaptation and curvature regularization. By regularizing sharpness $S(w)$, RMSProp-Norm is able to steer itself towards regions where the maximal locally stable step size of $2/S(w)$ is larger. In such regions, RMSProp-Norm can and does take larger steps. Thus, by regularizing sharpness, RMSProp-Norm enables 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.

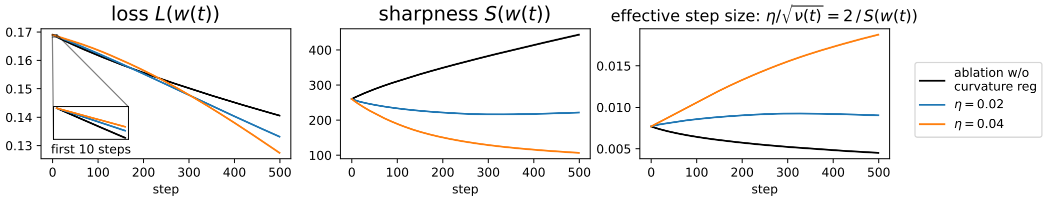


Figure 5: **"Acceleration via regularization" for RMSProp-Norm**. Starting from the same initialization, we run the RMSProp-Norm central flow at two different learning rates (in blue and orange), as well as an ablated flow $\frac{dw}{dt} = -\frac{2}{S(w)}\nabla L(w)$ (in black) with curvature regularization removed. These three flows all use the same step size strategy but differ 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 (CNN / CIFAR-10 / MSE).

In Figures 5 and 7, we compare the RMSProp-Norm 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 RMSProp-Norm central flow, because it traverses very sharp regions of weight space in which it is forced to take small steps.

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 5 and 7, over longer timescales, higher η steers the trajectory into lower-sharpness regions, in which RMSProp-Norm's effective step size will be larger, thereby tending to *speed up* optimization.

5 RMSPROP

We now study RMSProp (Tieleman & Hinton, 2012), which maintains an EMA ν of the elementwise squared gradients $\nabla L(w)^{\odot 2}$, and uses *per-coordinate* effective step sizes of $\eta/\sqrt{\nu}$:[15]

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2)\nabla L(w_t)^{\odot 2}, \quad w_{t+1} = w_t - \tfrac{\eta}{\sqrt{\nu_t}} \odot \nabla L(w_t), \qquad \text{(RMSProp)}$$

where $\odot$ represents the entrywise product. It is useful to view RMSProp as preconditioned gradient descent $w_{t+1} = w_t - P_t^{-1}\nabla L(w_t)$ with the dynamic preconditioner $P_t := \operatorname{diag}(\sqrt{\nu_t}/\eta)$.

In this section, we will show that RMSProp's preconditioner is *implicitly* determined by the algorithm's oscillatory dynamics, and we will make this preconditioner *explicit* for the first time. Interpreting this preconditioner sheds light on its efficacy, while also revealing a potential direction for future improvement. Finally, we will show that adaptive preconditioning is not the full story: RMSProp also implicitly regularizes curvature, and this behavior is crucial for its success.

5.1 THE MECHANICS OF RMSPROP; DERIVING A CENTRAL FLOW

The dynamics of RMSProp revolve around the *effective sharpness* $S^{\text{eff}}(w;\nu)$, defined as the largest eigenvalue of the preconditioned Hessian $P^{-1}H(w)$ where $P = \operatorname{diag}(\sqrt{\nu}/\eta)$.[16] When $S^{\text{eff}} > 2$, the iterates oscillate along the top right eigenvector of the preconditioned Hessian. In turn, such oscillations reduce S^{eff}, via a combination of two mechanisms: (1) they implicitly reduce curvature, and (2) they increase the gradient, growing ν and therefore P. The net effect is that the effective sharpness S^{eff} stays regulated around 2 throughout training (Cohen et al., 2022).[17]

In Appendix C.5 we derive a central flow $(w(t), \nu(t))$, in the same way as above, which models the time-averaged trajectory of RMSProp. We verify its accuracy in Figure 6 and Appendix D.

5.2 INTERPRETING RMSPROP VIA ITS CENTRAL FLOW

We now interpret the RMSProp central flow to understand the behavior of RMSProp. Because the dynamics usually transition from stable to EOS early in training, we focus on the EOS regime.

5.2.1 THE STATIONARY PRECONDITIONER

The central flow for RMSProp is harder to interpret than that for RMSProp-Norm, because even at EOS, ν cannot be expressed as a closed-form function of w, and instead remains an independent

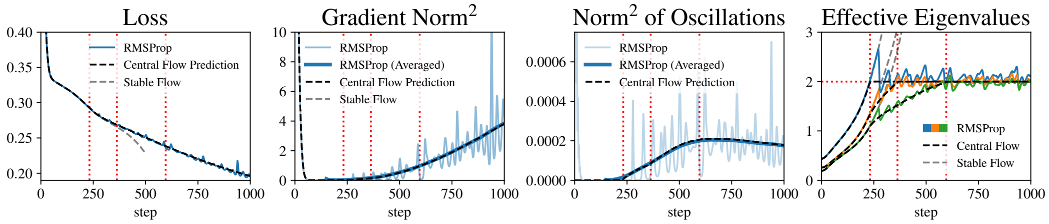


Figure 6: **Central Flow for RMSProp.** A ViT is trained on a subset of CIFAR-10 using RMSProp with $\eta = 4\times10^{-5}$ and $\beta_2 = 0.995$ (blue). It enters EOS at step 230 and multiple eigenvalues become unstable at step 360 (dotted red). The central flow (black) accurately models the time-averaged trajectory of RMSProp even at EOS, whereas the naive stable flow (gray) follows a different path.

variable.[18] Nevertheless, it turns out that, in some circumstances, ν implicitly converges under the dynamics of RMSProp to a value that depends on the current w alone. In particular, imagine holding the weights fixed at some value w, and letting ν "catch up." What value would ν converge to? We show in Proposition 3 that for any w, there is 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 converges to the stationary value $\bar{\nu}(w(t))$ during training (Figure 8).

In Proposition 2, we show that the corresponding *stationary preconditioner* $\overline{P}(w) = \operatorname{diag}(\sqrt{\overline{\nu}(w)}/\eta)$ is, remarkably, the optimal solution to a convex optimization problem over preconditioners:

$$\overline{P}(w) \quad := \underset{P \text{ diagonal},\, P \succeq 0}{\arg\min} \quad \operatorname{tr}(P) + \tfrac{1}{\eta^2} \underbrace{\|\nabla L(w)\|^2_{P^{-1}}}_{\text{optimization speed}} \quad \text{such that} \quad \underbrace{H(w) \preceq 2P}_{\text{local stability}}. \tag{12}$$

That is, **RMSProp implicitly solves the convex program eq. (12) to compute its preconditioner**. This is the precise sense in which RMSProp "adapts" to the local loss landscape.

We can now understand RMSProp's preconditioning strategy by interpreting the optimization problem eq. (12). 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, $\operatorname{tr}(P)$, 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.

However, matters are complicated by the presence of the second term in the eq. (12) objective. The quantity $\|\nabla L(w)\|^2_{P^{-1}}$ is the instantaneous rate of loss decrease under preconditioned gradient flow with preconditioner P. Minimizing this term actually *slows down* optimization (Figure 9). Therefore, the presence of this term implies that RMSProp's preconditioning strategy is not optimal.

5.2.2 IMPLICIT CURVATURE REDUCTION AND ACCELERATION VIA REGULARIZATION

Understanding the preconditioning strategy implicitly employed by 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 implicit curvature reduction induced by the oscillations. Substituting $\bar{P}$ into the RMSProp central flow, we can obtain a *stationary flow* in terms of w alone, which assumes that the preconditioner P is always fixed at its stationary value $\overline{P}$ (eq. 12):

$$\frac{dw}{dt} = \underbrace{\overline{P}(w)^{-1}}_{\substack{\text{stationary}\\ \text{preconditioner}}} \Big[\nabla L(w) + \underbrace{\tfrac{1}{2}\nabla_w \langle \Sigma, H(w) \rangle}_{\text{implicit curvature penalty}} \Big]. \tag{13}$$

where $\Sigma = \Sigma(w; \eta; \beta_2)$ is defined as the solution to a cone complementarity problem (eq. 25). Figure 10 shows that this stationary flow can accurately predict the instantaneous optimization speed of the central flow, given only access to $w(t)$ and not $\nu(t)$.

Empirically, we find that the implicit curvature reduction effect is crucial for the efficacy of the optimizer. In Figure 11, we compare the stationary flow against an ablated flow which uses the same preconditioning strategy, but leaves out the curvature penalty. We find that in the long run, this ablated flow navigates into sharper regions in which it takes smaller steps, and optimizes slower.

6 CONCLUSION

In this paper, we have developed a methodology for analyzing deep learning optimizers. To analyze an optimization algorithm, we derive a *central flow* which models the oscillatory optimizer's time-averaged trajectory, rendering implicit behaviors explicit. We have empirically demonstrated that these central flows can accurately predict long-term optimization trajectories of neural networks, and by interpreting these flows we have obtained new insights about optimizers' behavior.

These advances are made possible by the fact that we adopt different goals from most works in optimization. Rather than try to characterize global convergence rates, we set ourselves the more modest goal of characterizing the *local* optimization dynamics throughout training. The local dynamics are important, they are more interesting than may have been assumed (even vanilla gradient descent gives rise to rich, complex dynamics), and they are empirically consistent across different deep learning settings, which suggests that a general theory is feasible. We believe that similar analyses can be fruitfully conducted for other optimizers, and we hope to inspire work in that direction.

Reproducibility statement We have submitted the source code for reproducing the Appendix D experiments in the supplementary material. The precise mathematical definition of all central flows can be found in Appendix C.

Ethics statement This paper presents work whose goal is to advance the field of machine learning. As such, there are many potential societal consequences of our work.

REFERENCES

Atish Agarwala, Fabian Pedregosa, and Jeffrey 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.

Kwangjun Ahn, Sébastien Bubeck, Sinho Chewi, Yin Tat Lee, Felipe Suarez, and Yi Zhang. Learning threshold neurons via edge of stability. *Advances in Neural Information Processing Systems*, 36, 2024.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pp. 948–1024. PMLR, 2022.

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

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul 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`.

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

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

Lei Chen and Joan 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.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi 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`.

Zaiyi Chen, Zhuoning Yuan, Jinfeng Yi, Bowen Zhou, Enhong Chen, and Tianbao 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`.

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

Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability, 2022.

Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun Zhuang. Robustness to unbounded smoothness of generalized signsgd. *Advances in neural information processing systems*, 35:9955–9968, 2022.

Alex Damian, Eshaan Nichani, and Jason 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`.

Yann Dauphin, Atish Agarwala, and Hossein Mobahi. How hessian structure explains mysteries in sharpness regularization. *Advances in neural information processing systems*, 37, 2024.

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

Alexandre Défossez, Leon Bottou, Francis Bach, and Nicolas 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`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil 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`.

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

Mathieu Even, Scott Pesme, Suriya Gunasekar, and Nicolas 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.

Zhishuai Guo, Yi Xu, Wotao Yin, Rong Jin, and Tianbao Yang. A novel convergence analysis for algorithms of the adam family. *arXiv preprint arXiv:2112.03459*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yusu Hong and Junhong Lin. On convergence of adam for stochastic optimization under relaxed assumptions. *arXiv preprint arXiv:2402.03982*, 2024.

Florian Hübler, Junchi Yang, Xiang Li, and Niao He. Parameter-agnostic optimization under relaxed smoothness. In *International Conference on Artificial Intelligence and Statistics*, pp. 4861–4869. PMLR, 2024.

Stanislaw Jastrzębski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof 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`.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost 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`.

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

Itai Kreisler, Mor Shpigel Nacson, Daniel Soudry, and Yair Carmon. Gradient descent monotonically decreases the sharpness of gradient flow solutions in scalar networks and beyond, 2023.

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

Huan Li and Zhouchen 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.

Zhouzi Li, Zixuan Wang, and Jian Li. Analyzing sharpness along gd trajectory: Progressive sharpening and edge of stability. In *Neural Information Processing Systems*, 2022.

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

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

Aaron Mishkin, Ahmed Khaled, Yuanhao Wang, Aaron Defazio, and Robert M Gower. Directional smoothness and gradient methods: Convergence and adaptivity. *Advances in neural information processing systems*, 2024.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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

Vincent Roulet, Atish Agarwala, Jean-Bastien Grill, Grzegorz Swirszcz, Mathieu Blondel, and Fabian Pedregosa. Stepping on the edge: Curvature aware learning rate tuners. *arXiv preprint arXiv:2407.06183*, 2024.

Naichen Shi, Dawei Li, Mingyi Hong, and Ruoyu Sun. RMSprop converges with proper hyperparameter. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=3UDSdyIcBDA`.

Minhak Song and Chulhee 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`.

Tijmen Tieleman and Geoffrey 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.

Bohan Wang, Jingwen Fu, Huishuai Zhang, Nanning Zheng, and Wei Chen. Closing the gap between the upper bound and lower bound of adam's iteration complexity. *Advances in Neural Information Processing Systems*, 36, 2024a.

Bohan Wang, Huishuai Zhang, Qi Meng, Ruoyu Sun, Zhi-Ming Ma, and Wei Chen. On the convergence of adam under non-uniform smoothness: Separability from sgdm and beyond. *arXiv preprint arXiv:2403.15146*, 2024b.

Bohan Wang, Yushun Zhang, Huishuai Zhang, Qi Meng, Ruoyu Sun, Zhi-Ming Ma, Tie-Yan Liu, Zhi-Quan Luo, and Wei 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.

Mingze Wang, Jinbo Wang, Haotian He, Zilin Wang, Guanhua Huang, Feiyu Xiong, Zhiyu Li, Weinan E, and Lei Wu. Improving generalization and convergence by enhancing implicit regularization, 2024d. URL `https://arxiv.org/abs/2405.20763`.

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

Lei Wu, Chao Ma, and Weinan 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 (eds.), *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`.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd, 2018.

Junchi Yang, Xiang Li, Ilyas Fatkhullin, and Niao 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.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

Qi Zhang, Yi Zhou, and Shaofeng Zou. Convergence guarantees for rmsprop and adam in generalized-smooth non-convex optimization with affine noise variance. *arXiv preprint arXiv:2404.01436*, 2024.

Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge without any modification on update rules. *Advances in neural information processing systems*, 35: 28386–28399, 2022.

Xingyu Zhu, Zixuan Wang, Xiang Wang, Mo Zhou, and Rong 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`.

Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11127–11135, 2019.

NOTES

1. As is common in the optimization literature, we use curvature synonymously with the Hessian of the loss.

2. Barrett & Dherin (2021) argued that the accuracy of the gradient flow approximation can be improved by adding a penalty on the squared gradient norm to account for discretization error. However, their modified flow does not work (and is not intended to work) in the EOS regime, and in the stable regime, we found that the improvement in accuracy it brings is small. Therefore, for simplicity, we leave out any such term from our flows.

3. 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 learning rate is changing.

4. In taking the time-average of eq. (1), we assume that the eigenvector u changes slowly relative to the displacement x so that $\mathbb{E}[x_t u_t] \approx \mathbb{E}[x_t] u_t$.

5. Indeed, notice that $\left\langle H(\overline{w}_t), \delta_t \delta_t^T \right\rangle = \delta_t^T H(\overline{w}_t) \delta_t$, the curvature in the δ_t direction.

6. Our explanation generalizes the explanation in (Damian et al., 2023), which applied only to the special case when only the top Hessian eigenvalue has reached $2/\eta$.

7. 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.

8. 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 analysis.

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

10. 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. NGD), $\nu_t = \|\nabla L(w_t)\|^2$ so in the continuous time ODE, $\nu(t)$ needs to adapt “instantly” to $\|\nabla L(w(t))\|^2$. See Lemma 4 for additional justification for this correction term.

11. In the stable regime ($S^{\text{eff}} < 2/\eta$), the central flow is given by the stable flow eq. (8). Note that the value of $\frac{dw}{dt}$ for this flow is directly proportional to that of gradient flow, implying that this flow traverses the gradient flow trajectory, just at a different speed (i.e. with a nonlinear time-rescaling).

12. 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. (10).

13. The formula for σ^2 is somewhat opaque for general β_2. However, it is relatively simple in the special case of NGD ($\beta_2 = 0$). In this case, eq. (10) reduces to $\sigma^2(w;\eta) \approx \frac{\eta^2}{4}$.

14. 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 $\sigma^2(w;\eta,0) \approx \eta^2/4$ and $\sigma^2(w;\eta,1)$ 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.

15. 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 C.7). However, to simplify exposition, the main text focuses on this simpler version of RMSProp.

16. Note that $P^{-1}H(w)$ is similar to $P^{-1/2}H(w)P^{-1/2}$ which is symmetric and therefore diagonalizable with real eigenvalues. Therefore $P^{-1}H(w)$ also has real eigenvalues.

17. Note that in this section we have absorbed the learning rate η into the definition of P so the critical threshold is 2 rather than $2/\eta$. This simplifies the analysis.

18. 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 historical trajectory.

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 Gradient Descent **2**
3.1 The Mechanics of Gradient Descent 3
3.2 Deriving the GD Central Flow 4
3.2.1 The Special Case of One Unstable Eigenvalue 4
3.2.2 The General Case (Multiple Unstable Eigenvalues) 5
3.3 Interpreting Gradient Descent via its Central Flow 5

4 RMSProp-Norm **6**
4.1 The Mechanics of RMSProp-Norm 6
4.2 Deriving the Central Flow 6
4.3 Interpreting RMSProp-Norm via its Central Flow 7
4.3.1 Implicit step size selection 8
4.3.2 Implicit curvature reduction 8
4.3.3 Acceleration via regularization 8

5 RMSProp **9**
5.1 The Mechanics of RMSProp; Deriving a Central Flow 9
5.2 Interpreting RMSProp via its Central Flow 9
5.2.1 The stationary preconditioner 9
5.2.2 Implicit Curvature Reduction and Acceleration via Regularization 10

6 Conclusion **10**

A Additional Figures **17**

B Additional Experimental Details **21**

C Flow Derivations **21**
- C.1 Tensor Notation . . . 21
- C.2 On Local Time Averaging . . . 22
- C.3 Gradient Descent . . . 22
  - C.3.1 The Differential Varational Inequality Formulation . . . 22
  - C.3.2 The Cone Complementarity Problem Formulation . . . 23
  - C.3.3 The Projection Formulation . . . 24
  - C.3.4 The Rate of Loss Decrease . . . 24
- C.4 RMSProp-Norm . . . 25
  - C.4.1 The effect of the hyperparameters η, β_2 . . . 26
  - C.4.2 The small β_2 correction . . . 27
- C.5 RMSProp . . . 28
- C.6 Solving for the stationary ν in RMSProp . . . 30
- C.7 Arbitrary Preconditioned Methods . . . 31
- C.8 Cone Complementarity Problems . . . 32

D Full Experiments **33**
- D.1 Gradient Descent . . . 33
  - D.1.1 CIFAR 10 . . . 34
  - D.1.2 CIFAR 10 (4 class) . . . 37
  - D.1.3 Sorting . . . 43
- D.2 RMSProp-Norm . . . 44
  - D.2.1 CIFAR 10 (4 class) . . . 44
- D.3 RMSProp . . . 50
  - D.3.1 CIFAR 10 (4 class) . . . 50
  - D.3.2 Sorting . . . 56
  - D.3.3 The effect of β_2 . . . 58

E Extra figures **59**

A ADDITIONAL FIGURES

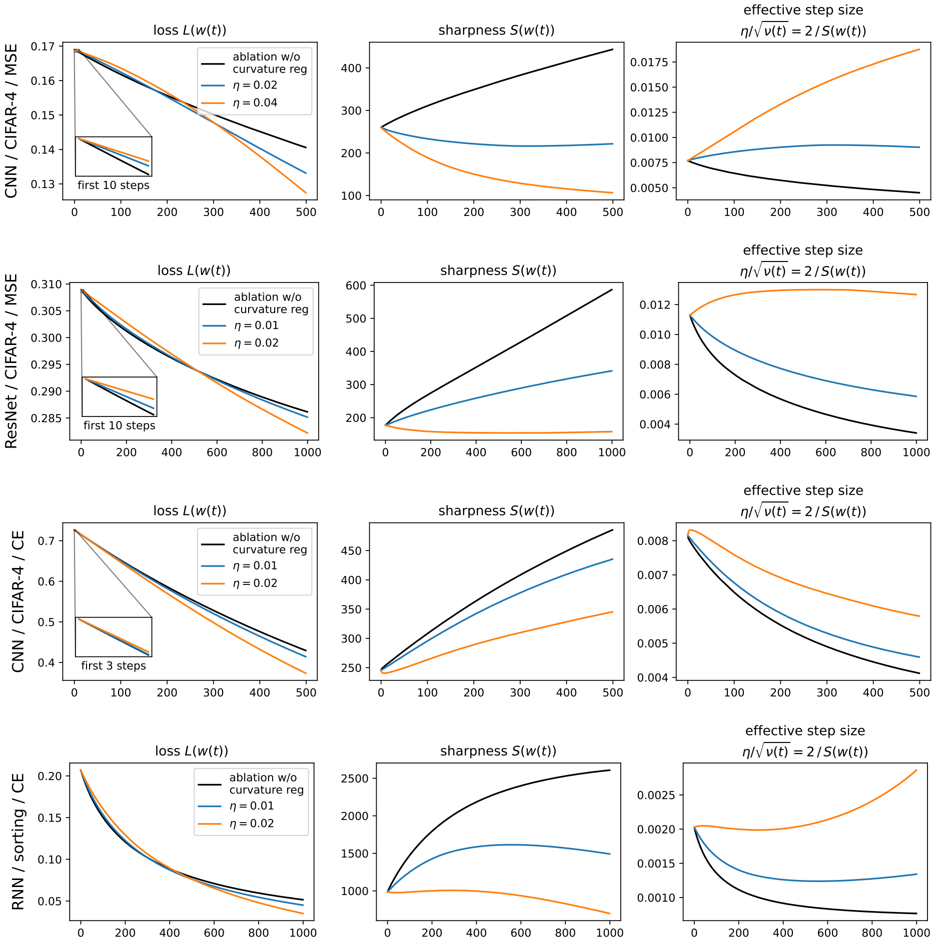


Figure 7: **"Acceleration via regularization" for RMSProp-Norm**. Starting from the same initialization, we run the RMSProp-Norm central flow at various learning rates, as well as an ablated flow $\frac{dw}{dt} = -\frac{2}{S(w)} \nabla L(w)$ with curvature regularization removed. These three flows all use the same step size strategy but differ in the strength of implicit curvature regularization. Initially (see inset), the flows with higher curvature regularization often optimize slower; however, over the longer run, they are able to take larger steps and optimize faster.

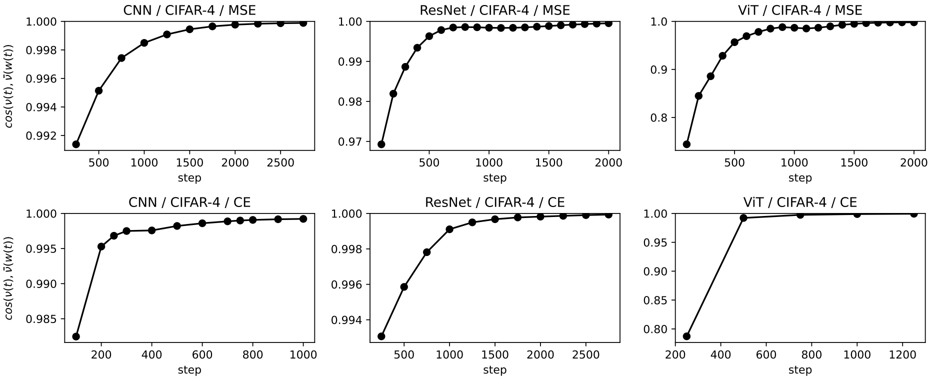


Figure 8: **The EMA ν reaches stationarity during training.** While running the RMSProp central flow, we monitor the cosine similarity between $\nu(t)$, the real EMA, and $\bar{\nu}(w(t))$, the stationary EMA. This cosine similarity rises to high values during training, implying that $\nu(t)$ reaches stationarity. Thus, we can reason about the stationary ν (and in particular, the corresponding stationary preconditioner eq. (12)) in order to reason about RMSProp's preconditioning strategy. Note that we compute $\bar{\nu}(w)$ using the Burer-Monteiro factorization, as described in appendix B.

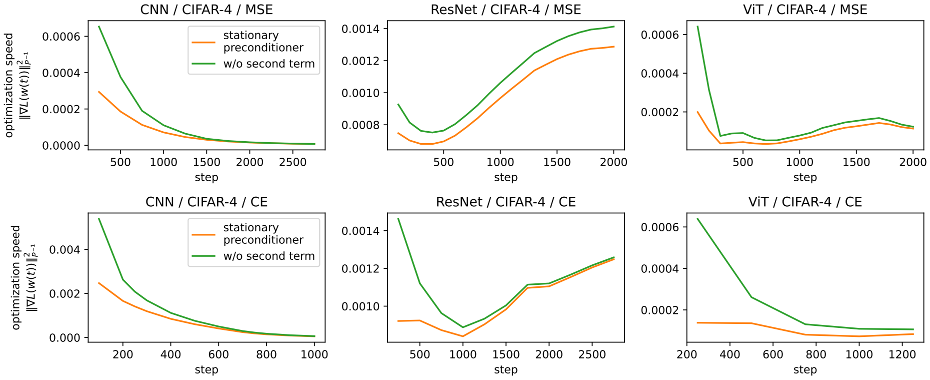


Figure 9: **RMSProp's implicit preconditioner is suboptimal**. We compare RMSProp's stationary preconditioner, defined as the solution to the optimization problem eq. (12), to an alternative preconditioner defined as the solution to an analogous optimization problem without the second term in the objective. We assess the efficacy of each preconditioner P by computing $\|\nabla L(w(t))\|^2_{P^{-1}}$, the instantaneous rate of loss decrease under the preconditioned gradient flow with preconditioner P. We observe that the rate of loss decrease for the alternative preconditioner (in green) is higher than the rate of loss decrease for the RMSProp stationary preconditioner (in orange), indicating that the alternative preconditioner would be better. Both preconditioners are computed using the Burer-Monteiro factorization, as described in appendix B.

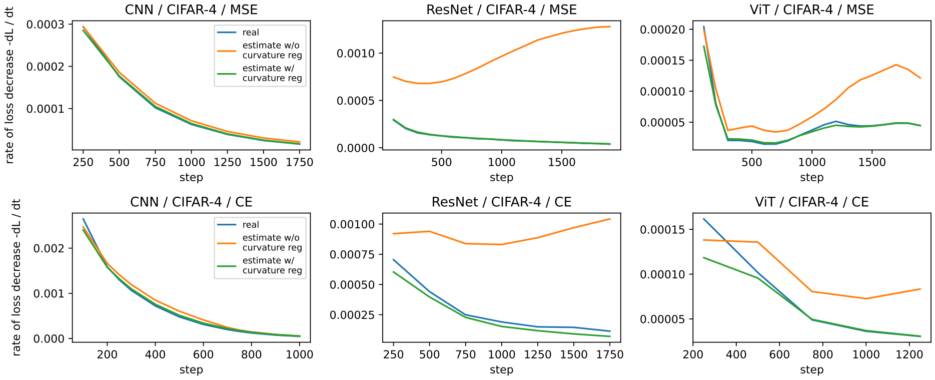


Figure 10: **Stationary flow predicts the rate of loss decrease**. The stationary flow eq. (13), which incorporates implicit curvature regularization, predicts (green) the rate of loss decrease (blue) more accurately than a naive estimate (orange) which uses the stationary preconditioner but does not incorporate curvature regularization. Note that all the estimates are off early in training, as the preconditioner has not yet reached stationarity (see Figure 8).

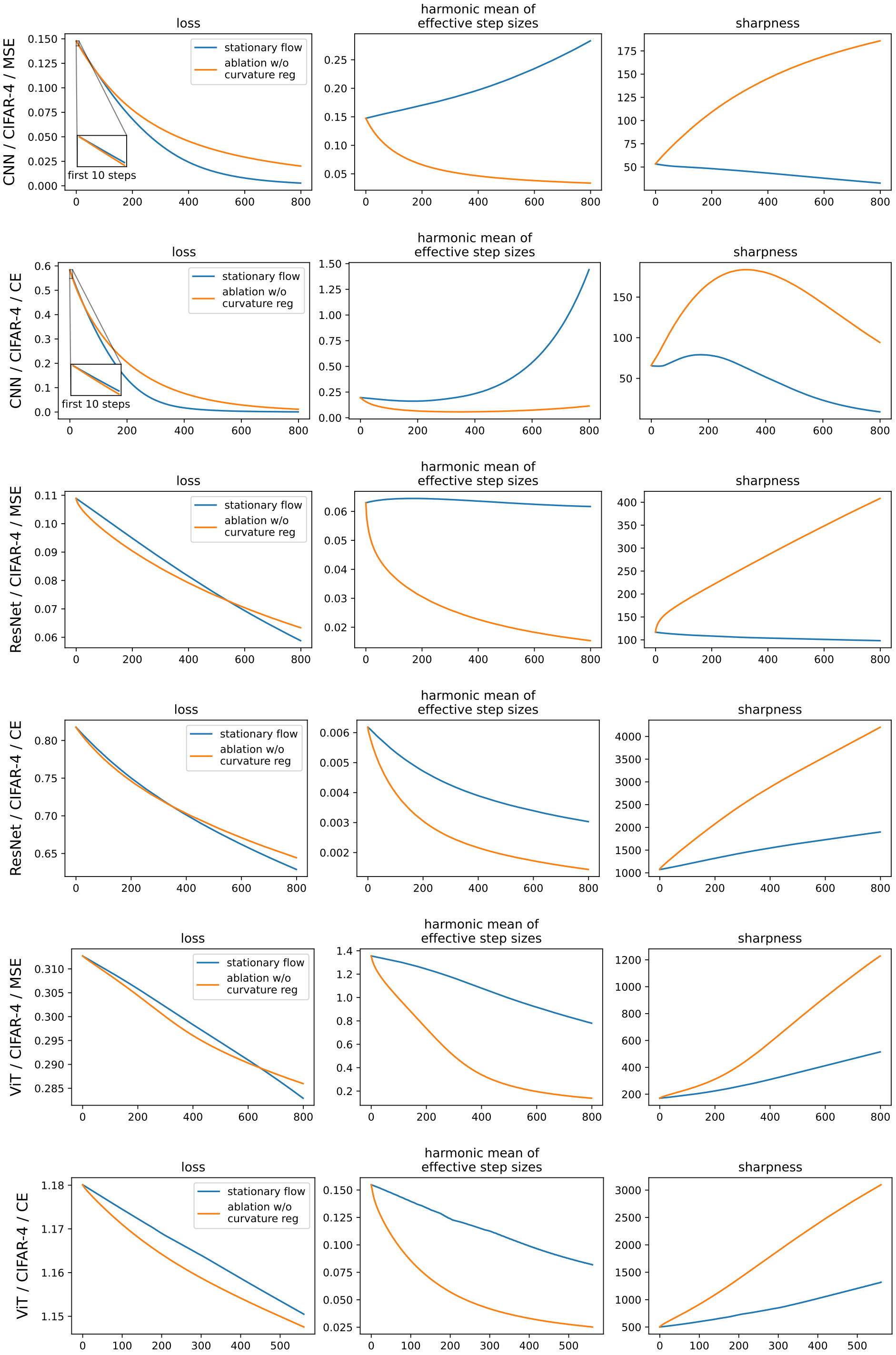


Figure 11: **"Acceleration via regularization" for RMSProp stationary flow**. We compare the RMSProp stationary flow (blue) to an ablated version (orange) which uses the same preconditioning strategy but leaves out the implicit curvature regularization. Initially, the stationary flow optimizes slower (left, initial), due to the presence of implicit curvature regularization. But over time, it navigates to lower-curvature regions (right), where it takes larger steps (middle), and optimizes faster (left). Each row is a different DL setting. The left column plots the train loss, the middle column plots the harmonic mean of the effective learning rates, the right column plots the sharpness.

B ADDITIONAL EXPERIMENTAL DETAILS

Architectures Our CNN is comprised of 4 layers with GeLU activations and uses average pooling before a linear readout layer. We use a modified ResNet without BatchNorm which has 20 layers and uses GeLU activations. Our vision transformer (ViT, (Dosovitskiy et al., 2021)) is based on the default PyTorch (Paszke et al., 2017) vision transformer and uses the GeLU activation as well. Our recurrent neural network (RNN) has a single layer and uses $\tanh$ activations. Our GPT2-style (Radford et al., 2019) transformer is based on the "nano" version of minGPT (`github.com/karpathy/minGPT`). See the attached code for additional architectural details.

Discretizing the flows We discretize each flow using Euler steps. We found that Euler steps of size $1/4$ sufficed to approximate the central flows, we use Euler steps of size $1/10$ to approximate the stable flows. For each trajectory: the true trajectory, the central flow, and the stable flow, we recompute the top eigenvectors of the Hessian from scratch every 5 steps. Eigenvalues and eigenvectors are computed using a sparse eigenvalue solver and hessian-vector products. For the intermediate steps, we reuse the eigenvectors and approximate the eigenvalues by the quadratic forms $u^T \nabla^2 L(w) u$. We compute the gradients of the eigenvalues by computing the contraction of the third derivative $\nabla^3 L(w)$ with the eigenvectors, i.e. $\nabla^3 L(w)[u, u]$ which can be implemented in standard PyTorch.

Computational Resources In total, our 115 experiments required about 2000 GPU-hours on a mix of $A6000$ and $A100$ GPUs.

Computing the RMSProp Stationary Preconditioner To verify that the RMSProp preconditioner indeed solves the convex program eq. (12), we explicitly solve this convex program throughout the training trajectory and compare it with the true RMSProp preconditioner. To solve eq. (12), we apply a fixed point iteration based on the fixed point equations for Σ, P. To derive our fixed point iteration, we factorize $\Sigma = DD^T$ where $D \in \mathbb{R}^{d \times r}$ (as in the Burer-Monteiro factorization (Burer & Monteiro, 2005)) and $r \ll d$ is intended to upper bound the number of unstable eigenvalues. Then the formula for $\frac{d\nu}{dt}$ reduces to:

$$\nu = \nabla L(w)^{\odot 2} + \operatorname{diag}[H \Sigma H] = \nabla L(w)^{\odot 2} + (HD)^{\odot 2} \mathbf{1}.$$

In addition, we will use that the span of Σ lies in the critical subspace, so

$$\operatorname{diag}[\nu^{-1/2}] HD = \frac{\eta}{2} D.$$

We begin with a random initial guess for D and iteratively update D, ν by:

$$\nu \leftarrow \nabla L(w)^{\odot 2} + (HD)^{\odot 2} \mathbf{1}$$
$$D \leftarrow \frac{\eta}{2} \operatorname{diag}[\nu^{-1/2}] HD.$$

If this fixed point iteration converges, we have

$$\nu = \nabla L(w)^{\odot 2} + (HD)^{\odot 2} \mathbf{1} \quad \text{and} \quad \operatorname{diag}[\nu^{-1/2}] HD = \frac{2}{\eta} HD$$

so that $\frac{d\nu}{dt} = 0$ and the span of $\Sigma = DD^T$ is in the critical subspace.

To verify the results of this fixed point iteration, we compare the objective values of the primal and dual programs. The dual program to eq. (12) is:

$$\max_{\Sigma \succeq 0} \langle \Sigma, H(w) \rangle + 2 |\nabla L(w)| \cdot \sqrt{1 - 2 \operatorname{diag} \Sigma} \quad \text{such that} \quad \operatorname{diag} \Sigma \preceq 1/2.$$

C FLOW DERIVATIONS

We will now derive the general central flows for gradient descent, RMSProp-Norm, and RMSProp. We begin by defining the necessary tensor notation.

C.1 TENSOR NOTATION

For a k-tensor T and a s-tensor A with $s < k$ we define $T[A]$ to be the contraction of T with A along its last s indices, i.e.

$$(T[A])_{i_1, \dots, i_{k-s}} = \sum_{j_1, \dots, j_s} T_{i_1, \dots, i_{k-s}, j_1, \dots, j_s} A_{j_1, \dots, j_s}. \tag{14}$$

We also define $\langle T, T' \rangle$ for two k-tensors T, T' of the same shape by:

$$\langle T, T' \rangle = \sum_{i_1,\ldots,i_k} T_{i_1,\ldots,i_k} T'_{i_1,\ldots,i_k}. \tag{15}$$

We will also use $T[u_1, \ldots, u_j]$ to denote $T[u_1 \otimes \cdots \otimes u_j]$.

C.2 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 oscillates around $2/\eta$, the time-average is coarse enough to smooth out these oscillations so that $S(\mathbb{E}[w_t]) = 2/\eta$

When we empirically verify the accuracy of our central flows, we use Gaussian smoothing with a standard deviation of 50 steps, i.e. we define

$$\mathbb{E}[f_t] := \frac{\sum_s f_s c_{t-s}}{\sum_s c_{t-s}} \quad \text{where} \quad c_j := \exp\left(\frac{-j^2}{2 \times 50^2}\right).$$

C.3 GRADIENT DESCENT

We begin by proving the Taylor expansion used in Section 3.1:

Lemma 1. *Assume that $\overline{w}$ is such that the top eigenvalue of $H(\overline{w})$ has multiplicity 1 and let $w = \overline{w} + xu$ where u is the top eigenvector of $H(\overline{w})$. Furthermore, assume that $\sup_{w' \in U} \left\| \nabla^4 L(w') \right\| < \infty$ for some open neighborhood U of $\overline{w}$. Then,*

$$\nabla L(w) = \nabla L(\overline{w}) + S(\overline{w}) x u + \tfrac{x^2}{2} \nabla S(\overline{w}) + \mathcal{O}(x^3).$$

Proof. By Taylor's theorem,

$$\nabla L(w) = \nabla L(\overline{w}) + H(\overline{w}) x u + \tfrac{x^2}{2} \nabla^3 L(\overline{w})[u, u] + \mathcal{O}(x^3).$$

Because u is an eigenvector of $H(\overline{w})$ with eigenvalue $S(\overline{w})$, the second term can be simplified to $S(\overline{w}) x u$. Finally, by Danskin's theorem (or equivalently the standard formula for the derivative of an eigenvalue):

$$\nabla_{\overline{w}} S(\overline{w}) = \nabla_{\overline{w}} \left[\max_{\|u\|=1} u^T H(\overline{w}) u \right] = \nabla_{\overline{w}} \left[u^T H(\overline{w}) u \right] = \nabla^3 L(\overline{w})[u, u]$$

where u is the argmax of the second expression, i.e. the top eigenvector of the Hessian at $\overline{w}$. $\square$

We now define some notation that will be necessary for the remainder of the section. First, let $U(w) := \ker[\eta H(w) - 2]$ denote the *critical subspace*, i.e. the directions in weight space that are at the edge of stability. Let $\mathrm{Sym}(U(w)) \subseteq \mathbb{R}^{d \times d}$ denote the subspace of symmetric matrices whose span is contained in $U(w)$.

C.3.1 THE DIFFERENTIAL VARATIONAL INEQUALITY FORMULATION

As in the main text, we will assume that GD oscillates around the central flow $w(t)$ with covariance $\Sigma(t) \in \mathrm{Sym}(U(w))$ so that it follows:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \tfrac{1}{2} \nabla^3 L(w)[\Sigma(t)] \right]. \tag{16}$$

To determine Σ, we impose three conditions for all times t:

- **non-negativity:** As a covariance matrix, $\Sigma(t)$ is positive semidefinite, i.e. $\Sigma(t) \succeq 0$
- **stability:** The sharpness remains bounded by $2/\eta$, i.e. $H(w(t)) \preceq (2/\eta) I$
- **complementarity:** The span of $\Sigma(t)$ (i.e. the span of the oscillations) is contained within the critical subspace $U(w(t))$. Equivalently, $\Sigma(t) \in \mathrm{Sym}(U(w(t)))$.

We say that $(w(t), \Sigma(t))$ follow the GD central flow if they follow eq. (16) along with these conditions:

Definition 1 (GD Central Flow, DVI Formulation)**.** We say that $\{w(t), \Sigma(t)\}_{t \geq 0}$ follow the GD central flow if for almost all t they satisfy eq. (16) along with the conditions: $\Sigma(t) \succeq 0$, $H(w(t)) \preceq (2/\eta)I$, and $\Sigma(t) \in \mathrm{Sym}(U(w))$.

Definition 1 is an example of a *differential variational inequality* (DVI). It provides conditions that the central flow must satisfy but it is not a-priori clear that such a flow exists or is unique. To prove this, and to turn Definition 1 into a form that we can simulate numerically, we will show that for almost all times t, $\Sigma(t)$ solves a low dimensional convex *cone complementarity program* (CCP) This CCP will be guaranteed to have a unique solution, which can be easily computed numerically.

C.3.2 THE CONE COMPLEMENTARITY PROBLEM FORMULATION

We will start with the DVI formulation, Definition 1, and prove two additional conditions on $\Sigma(t)$ which will allow us to solve for $\Sigma(t)$. For simplicity, we will look for a choice of $\Sigma(t)$ is right-continuous so that we can reason about the immediate future using the chain rule. First, we consider the change in the Hessian along the central flow ansatz eq. (16), restricted to the critical subspace. We will denote this quantity by $\dot{H}(t) \in \mathrm{Sym}(U(w(t)))$. This generalizes the computation of $\frac{dS}{dt}$ in Section 3. To compute $\dot{H}$, we define the linear operator $\mathcal{T}_w : \mathrm{Sym}(U(w)) \to \mathbb{R}^d$ by

$$\mathcal{T}_w[\Sigma] := \nabla_w \langle H(w), \Sigma \rangle \quad \forall\, \Sigma \in \mathrm{Sym}(U(w)).$$

Intuitively, $\mathcal{T}_w$ will play a role analogous to that of $\nabla S(w)$ in Section 3. The operator $\mathcal{T}_w : \mathrm{Sym}(U(w)) \to \mathbb{R}^d$ takes as input a $d \times d$ matrix $\Sigma \in \mathrm{Sym}(U(w))$ and returns the gradient of $\langle H(w), \Sigma \rangle$, the Σ-weighted Hessian. Meanwhile, its transpose $\mathcal{T}_w^\top : \mathbb{R}^d \to \mathrm{Sym}(U(w))$ takes as input a direction, and returns the directional derivative of the Hessian, restricted to the critical subspace. Therefore, $\frac{dw}{dt} = -\eta\big[\nabla L(w) + \frac{1}{2}\mathcal{T}_w[\Sigma]\big]$, and $\dot{H}$ is given by:

$$\dot{H}(t) := \frac{dH}{dt}\bigg|_{U(w)} = \mathcal{T}_w^T\left[\frac{dw}{dt}\right] = \underbrace{\mathcal{T}_w^T[-\nabla L(w)]}_{=:\alpha(w)} - \underbrace{\tfrac{1}{2}\mathcal{T}_w^T\mathcal{T}_w}_{=:\beta(w)}[\Sigma].$$

To simplify notation, we define $\alpha(w), \beta(w)$ by:

$$\alpha(w) := \mathcal{T}_w^T[-\nabla L(w)] \in \mathrm{Sym}(U(w)), \quad \beta(w) := \tfrac{1}{2}\mathcal{T}_w^T\mathcal{T}_w \in \mathrm{Sym}(U(w)) \otimes \mathrm{Sym}(U(w)),$$

so that $\dot{H} = \alpha - \beta[\Sigma]$. In order to maintain stability, i.e. $H \preceq (2/\eta)I$, we need to ensure $\dot{H} \preceq 0$, i.e. $\alpha \preceq \beta[\Sigma]$. However, this condition only lower bounds Σ and does not yet uniquely determine it. To do so, we will need to differentiate the complementarity condition. Recall that $\dot{H}$ measures the change in the Hessian, restricted to the critical subspace. The directions in the kernel of $\dot{H}$ will remain at EOS. However, the curvature in the directions where $\dot{H} \prec 0$ will drop below $2/\eta$, so these directions will leave the critical subspace. In other words, the critical subspace at time $t + \epsilon$ will be contained in $\ker[\dot{H}]$, which is a subspace of the critical subspace. Because we assumed Σ is right-continuous, this implies that $\Sigma(t)$ must lie in this subspace, i.e. $\mathrm{span}[\Sigma] \in \ker[\dot{H}]$. Equivalently, because $\Sigma \succeq 0$ and $\dot{H} \preceq 0$, we can rewrite this condition as $\langle \Sigma, \dot{H} \rangle = 0$. Together, these constraints:

$$\Sigma \succeq 0, \quad \dot{H} \preceq 0, \quad \left\langle \Sigma, \dot{H} \right\rangle = 0, \quad \dot{H} = \alpha - \beta[\Sigma]$$

define a *cone complementarity problem* for Σ (Definition 8). We show in Lemma 5 that if β is symmetric and has full rank, the solution to this CCP is unique, and we denote it by $\mathrm{CCP}(\alpha, \beta)$. Otherwise it denotes the set of all solutions. We can now define the central flow in terms of this CCP:

Definition 2 (GD Central Flow, CCP Formulation)**.** We say $\{w(t)\}_{t \geq 0}$ follows the GD central flow if for almost all $t \geq 0$, w satisfies eq. (16) with $\Sigma \in \mathrm{CCP}(\alpha(w), \beta(w))$.

Definition 2 is the simplest version of the central flow to simulate numerically. By picking a basis for $\mathrm{Sym}(U(w))$, which has dimension $\frac{k(k+1)}{2}$, we can materialize the linear operator $\mathcal{T}_w$ as a $\frac{k(k+1)}{2} \times d$ dimensional matrix, compute $\alpha(w), \beta(w)$, solve the low-dimensional CCP to compute Σ, and take a small Euler step on the flow using this Σ.

In the next section, we will see that this CCP formulation can be reinterpreted as a projected gradient flow, which will help to interpret the behavior of the gradient descent central flow.

C.3.3 THE PROJECTION FORMULATION

Let $\mathbb{S}_\eta := \{w \ : \ S(w) \leq 2/\eta\}$ denote the *stable set*. In this section we will show that the gradient descent central flow can be reinterpreted as projected gradient flow constrained to this set.

Definition 3 (GD Central Flow, Projection Formulation)**.** We say that $\{w(t)\}_{t \geq 0}$ follows the gradient descent central flow if for almost all t,

$$\frac{dw}{dt} = \mathrm{proj}_{T_{\mathbb{S}_\eta(w)}}[-\eta \nabla L(w)] \quad \text{where} \quad \mathbb{S}_\eta := \{w \ : \ S(w) \leq 2/\eta\}, \tag{17}$$

where $\mathrm{proj}_{T_{\mathbb{S}_\eta(w)}}$ denotes the orthogonal projection onto the *tangent cone* of $\mathbb{S}_\eta$ at iterate w, i.e. the set of directions that, if followed, would not immediately increase the curvature in the critical subspace: $T_{\mathbb{S}_\eta(w)} = \{v \in \mathbb{R}^d : \mathcal{T}_w^T[v] \preceq 0\}$.

The equivalence between Definition 3 and the CCP formulation Definition 2 ia a consequence of this simple lemma:

Lemma 2. *Let $w \in \mathbb{S}_\eta$. Then*

$$\mathrm{proj}_{T_{\mathbb{S}_\eta(w)}}[v] = v - \tfrac{1}{2}\mathcal{T}_w[\Sigma] \quad \textit{where} \quad \Sigma \in CCP(\mathcal{T}_w^T[v], \tfrac{1}{2}\mathcal{T}_w^T \mathcal{T}_w).$$

Proof. Note that the tangent space of $\mathbb{S}_\eta$ is given by the set: $\{v \ : \ \mathcal{T}_w^T[v] \preceq 0\}$. Therefore the projection is given by solving the quadratic program:

$$\min_\delta \|\delta\|^2 \quad \text{such that} \quad \mathcal{T}_w^T[v + \delta] \preceq 0.$$

The KKT conditions imply that there exists a Σ such that:

$$\delta = -\tfrac{1}{2}\mathcal{T}_w[\Sigma], \quad \left\langle \Sigma, \mathcal{T}_w^T[v + \delta] \right\rangle = 0, \quad \Sigma \succeq 0,$$

which implies that $\Sigma \in CCP(\mathcal{T}_w^T[v], \frac{1}{2}\mathcal{T}_w^T \mathcal{T}_w)$. □

We now intuitively reconcile eq. (17) with the CCP definition:

- When $S(w) < 2/\eta$, w is in the interior of $\mathbb{S}_\eta$ so $U(w) = \emptyset$, the tangent cone is the entire space, and the projection is the identity map. Therefore eq. (17) 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: $-\eta \Pi^\perp_{\nabla S(w)} \nabla L(w)$. Otherwise, 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.
- In general, computing the orthogonal projection onto $T_{\mathbb{S}_\eta(w)} = \{v : \mathcal{T}_w^\top[v] \preceq 0\}$ requires solving a semidefinite quadratic program for which Σ is the Lagrangian dual variable. The KKT conditions of this quadratic program are equivalent to the CCP that defines Σ above.

C.3.4 THE RATE OF LOSS DECREASE

Proposition 1. *Under the GD central flow (definition 3), for almost all t we have*

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

This implies $\frac{dL}{dt} \leq 0$ (i.e. the loss monotonically decreases) and $\frac{dL}{dt} \geq -\eta \|\nabla L(w)\|^2$ (i.e. the loss decreases at a slower rate than under the gradient flow).

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 we used that for any orthogonal projection $\mathrm{proj}[cv] = \mathrm{proj}[c]$ when $c \geq 0$ to pull out the η, and that $\langle v, \mathrm{proj}[v] \rangle = \|\mathrm{proj}[v]\|^2$ to get the last equality. Finally, the comparison with gradient flow follows from the inequality $\|\mathrm{proj}[v]\| \leq \|v\|$ for any orthogonal projection. $\square$

C.4 RMSPROP-NORM

We follow a similar derivation to gradient descent. We define $U(w, \nu) := \ker[\nu^{-1/2} H(w) - (2/\eta) I]$, and let $\mathrm{Sym}(U(w, \nu))$ denote the subspace of symmetric matrices with span in $U(w, \nu)$. We now proceed with the time-averaging derivation.

If $w = \overline{w} + \delta$ with $\mathbb{E}[\delta] = 0$, we have that

$$\mathbb{E}\,\|\nabla L(w)\|^2 \approx \mathbb{E}\,\|\nabla L(\overline{w})\|^2 + \mathbb{E}\,\left\|\nabla^2 L(\overline{w})\delta\right\|^2 = \|\nabla L(\overline{w})\|^2 + \left\langle \nabla^2 L(\overline{w})^2, \Sigma \right\rangle \tag{18}$$

where $\Sigma = \mathbb{E}[\delta\delta^T]$. Because we assume that $\Sigma \in \mathcal{S}_{\text{crit}}$, $\nabla^2 L(\overline{w})\Sigma = S(\overline{w})\Sigma$, so this expression is equal to

$$\mathbb{E}\,\|\nabla L(w)\|^2 \approx \|\nabla L(\overline{w})\|^2 + S(\overline{w})^2 \,\mathrm{tr}(\Sigma).$$

This suggests the central flow ansatz:

$$\begin{aligned} \frac{dw}{dt} &= -\frac{\eta}{\sqrt{\nu}}\left[\nabla L(w) + \tfrac{1}{2}\nabla^3 L(w)[\Sigma(t)]\right] \\ \frac{d\nu}{dt} &= \tfrac{1-\beta_2}{\beta_2}\Big[\|\nabla L(w)\|^2 + S(w)^2 \mathrm{tr}(\Sigma(t)) - \nu\Big]. \end{aligned} \tag{19}$$

We can now give the differential variational inequality definition for the RMSProp-Norm central flow:

Definition 4 (RMSNorm Central Flow, Differential Variational Inequality Formulation)**.** We say that $\{(w(t), \nu(t), \Sigma(t))\}_{t \geq 0}$ satisfy the RMSProp-Norm central flow if they satisfy eq. (19) along with the conditions: $\Sigma(t) \succeq 0$, $\nu(t)^{-1/2} H(w(t)) \preceq 2/\eta$, and $\Sigma(t) \in \mathcal{S}_{\text{crit}}(w(t), \nu(t))$ for almost all t.

As for gradient descent, this definition is fairly opaque and does not give a way of actually computing Σ. To do so, we will again derive a cone-complementarity problem formulation of the RMSProp-Norm central flow. The first step in this derivation is to differentiate the stability constraint: $\nu^{-1/2} H(w) \preceq 2/\eta$. We will fix a time t and use U to denote $U(w, \nu)$ to simplify notation. We will also define the linear operator $\mathcal{T} : \mathrm{Sym}(U(w, \nu)) \to \mathbb{R}^d$ by

$$\mathcal{T}[\Sigma] = \nabla_w \langle H(w), \Sigma \rangle \quad \text{for} \quad \Sigma \in \mathrm{Sym}(U(w, nu)).$$

Let $\dot{H} := \frac{d}{dt}\nu^{-1/2} H(w)\big|_U$ be the time-derivative of the preconditioned Hessian restricted to the critical subspace. In order to avoid violating the stability condition, we need to enforce $\dot{H} \preceq 0$. We can compute $\dot{H}$ using the chain rule.

$$\begin{aligned} \dot{H} &:= \frac{d}{dt}\nu^{-1/2} H(w)\bigg|_U \\ &= \nu^{-1/2}\left(\frac{d}{dt}H(w)\right)\bigg|_U + \left(\frac{d}{dt}\nu^{-1/2}\right) H(w)\bigg|_U \\ &= \nu^{-1/2}\mathcal{T}^T\left[\frac{dw}{dt}\right] - \frac{1}{2\nu^{3/2}}\frac{d\nu}{dt}\frac{2\sqrt{\nu}}{\eta} I \\ &= \frac{\eta}{\nu}\mathcal{T}^T\left[-\nabla L(w) - \tfrac{1}{2}\mathcal{T}[\Sigma]\right] + \frac{1-\beta_2}{\beta_2}\frac{1}{\eta\nu}\Big[\nu - \|\nabla L(w)\|^2 - S(w)^2\,\mathrm{tr}(\Sigma)\Big] I. \end{aligned}$$

We can now assume that (w, ν) are at EOS so that $\nu = \frac{\eta^2 S(w)^2}{4}$. Otherwise, complementarity forces $\Sigma = 0$. Substituting this gives:

$$\begin{aligned} \dot{H} &= \frac{4}{\eta S(w)^2}\mathcal{T}^T\left[-\nabla L(w) - \tfrac{1}{2}\mathcal{T}[\Sigma]\right] \\ &\quad + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^3 S(w)^2}\left[\frac{\eta^2 S(w)^2}{4} - \|\nabla L(w)\|^2 - S(w)^2\,\mathrm{tr}(\Sigma)\right] I. \end{aligned}$$

We will now group the constant terms and the terms linear in Σ. Define:

$$\alpha(w) = \frac{4}{\eta S(w)^2}\mathcal{T}^T[-\nabla L(w)] + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^3 S(w)^2}\left[\frac{\eta^2 S(w)^2}{4} - \|\nabla L(w)\|^2\right]I$$

$$\beta(w) = \frac{2}{\eta S(w)^2}\mathcal{T}^T\mathcal{T} + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^3}I \otimes I.$$

Then $\dot{H} = \alpha - \beta[\Sigma]$. As in the derivation for GD, if $\dot{H}$ is strictly negative definite in some direction, these directions will drop from the critical subspace so right continuity forces $\Sigma \in \ker[\dot{H}]$. This gives us the cone complementarity definition of the RMSProp-Norm central flow:

Definition 5 (RMSProp-Norm Central Flow, CCP Formulation)**.** We say that $\{(w(t), \nu(t)\}_{t \geq 0}$ follow the RMSProp-Norm central flow if they satisfy eq. (19) where $\Sigma \in CCP(\alpha(w), \beta(w))$.

We can now use this formulation to prove eq. (10) when $U(w, \nu)$ is one dimensional. In the one dimensional case, α, β are both scalars and assuming that $\alpha(w) > 0$ the solution to the CCP is simply $\Sigma = \alpha(w)/\beta(w)$. In addition, in this one dimensional case we simply have $\mathcal{T} = \nabla S(w)$. Therefore,

$$\sigma^2 = \frac{\frac{4}{\eta S(w)^2}\langle -\nabla L(w), \nabla S(w)\rangle + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^3 S(w)^2}\left[\frac{\eta^2 S(w)^2}{4} - \|\nabla L(w)\|^2\right]}{\frac{2}{\eta S(w)^2}\|\nabla S(w)\|^2 + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^3}} \tag{20}$$

$$= \frac{\beta_2\langle -\nabla L(w), \nabla S(w)\rangle + (1-\beta_2)\left[\frac{S(w)^2}{4} - \frac{\|\nabla L(w)\|^2}{\eta^2}\right]}{\beta_2\frac{1}{2}\|\nabla S(w)\|^2 + (1-\beta_2)S(w)^2/\eta^2}. \tag{21}$$

C.4.1 THE EFFECT OF THE HYPERPARAMETERS η, β_2

Lemma 3. *Fix an iterate $w = w(t)$ and define $\nu = \nu(w;\eta) := \frac{\eta^2 S(w)^2}{4}$ so that $S^{\text{eff}}(w,\nu) = 2/\eta$. Let U be the top eigenspace of $H(w)$. Then if $\Sigma(t) \succ 0$ under the RMSProp-Norm central flow (Definition 5), we have that*

$$\frac{\partial}{\partial\eta}\frac{dH}{dt}\bigg|_U = -cI_U \quad \textit{where} \quad C \geq 0$$

and where I_U denotes the identity matrix on the subspace U. In other words, larger learning rates η more aggressively decrease the curvature and they do so uniformly across all eigenvalues in the critical subspace U. In addition,

$$\frac{\partial}{\partial\beta_2}\frac{dH}{dt}\bigg|_U = -CI_U$$

where $C \geq 0$ when $\operatorname{tr}\Sigma\big|_{\beta_2=1} \leq \operatorname{tr}\Sigma\big|_{\beta_2=0}$ and $C \leq 0$ otherwise. Therefore β_2 can either monotonically increase or decrease the curvature regularization depending on whether gradient descent ($\beta_2 = 1$) or normalized gradient descent ($\beta_2 = 0$) would have a larger oscillation variance.

Proof. The condition $\Sigma(t) \succ 0$ implies that $\dot{H}(t) = 0$ and $\Sigma = \beta^{-1}\alpha$. We will rescale α, β by $\eta S(w)^2/4$:

$$\hat{\alpha} = \mathcal{T}^T[-\nabla L(w)] + \frac{1-\beta_2}{\beta_2}\left[\frac{S(w)^2}{4} - \frac{\|\nabla L(w)\|^2}{\eta^2}\right]I$$

$$\hat{\beta} = \tfrac{1}{2}\mathcal{T}^T\mathcal{T} + \frac{1-\beta_2}{\beta_2}\frac{S(w)^2}{\eta^2}I \otimes I.$$

We will also use $\hat{\alpha}_w, \hat{\beta}_w$ to refer to just the first terms in $\hat{\alpha}, \hat{\beta}$:

$$\hat{\alpha}_w = \mathcal{T}^T[-\nabla L(w)] \quad \text{and} \quad \hat{\beta}_w = \tfrac{1}{2}\mathcal{T}^T\mathcal{T}.$$

Then $\Sigma = \hat{\beta}^{-1}\hat{\alpha}$. Differentiating this with respect to η, β_2 gives:

$$\frac{\partial}{\partial\eta}\Sigma = \frac{1-\beta_2}{2\eta^3\beta_2}\hat{\beta}^{-1}[I]\left[\|\nabla L(w)\|^2 + S(w)^2\operatorname{tr}\Sigma\right]$$

$$\frac{\partial}{\partial\beta_2}\Sigma = -\frac{1}{\beta_2^2}\hat{\beta}^{-1}[I]\left[\frac{S(w)^2}{4} - \frac{\|\nabla L(w)\|^2}{\eta^2} - \frac{S(w)^2}{\eta^2}\operatorname{tr}\Sigma\right]$$

Then:

$$\frac{\partial}{\partial \eta}\frac{dH}{dt}\bigg|_U = -\hat{\beta}_w\left[\frac{\partial}{\partial \eta}\Sigma\right], \quad \frac{\partial}{\partial \beta_2}\frac{dH}{dt}\bigg|_U = -\hat{\beta}_w\left[\frac{\partial}{\partial \beta_2}\Sigma\right]$$

Therefore it suffices to compute $\hat{\beta}_w\hat{\beta}^{-1}[I]$. To compute this, we use the Sherman–Morrison formula. Let $c = \frac{1-\beta_2}{\beta_2}\frac{S(w)^2}{\eta^2}$. Then,

$$\hat{\beta}_w\hat{\beta}^{-1}[I] = I\left[1 - \frac{c\beta_w^{-1}[I,I]}{1 + c\beta_w^{-1}[I,I]}\right] = I\left[\frac{1}{1 + c\beta_w^{-1}[I,I]}\right].$$

Finally, note that $\left[\frac{1}{1+c\beta_w^{-1}[I,I]}\right] \geq 0$, which immediately implies the result for η. For β_2, we have shown that

$$\frac{\partial}{\partial \beta_2}\frac{dH}{dt}\bigg|_U = -CI_U$$

where $C \geq 0$ if and only if

$$\operatorname{tr}\Sigma \leq \frac{\eta^2}{4} - \frac{\|\nabla L(w)\|^2}{S(w)^2}.$$

To rewrite this in a form that is independent of Σ, note that

$$\begin{aligned}
\operatorname{tr}\Sigma &= \operatorname{tr}\left[\hat{\beta}^{-1}\hat{\alpha}\right] \\
&= \left\langle \hat{\beta}^{-1}[I], \alpha \right\rangle \\
&= \left\langle \alpha, \hat{\beta}_w^{-1}[I] \right\rangle \left[\frac{1}{1 + c\beta_w^{-1}[I,I]}\right] \\
&= \frac{\operatorname{tr}\left[\hat{\beta}_w^{-1}\hat{\alpha}_w\right] + c\beta_w^{-1}[I,I]\left[\frac{\eta^2}{4} - \frac{\nabla L(w)^2}{S(w)^2}\right]}{1 + c\beta_w^{-1}[I,I]}.
\end{aligned}$$

Therefore $\operatorname{tr}\Sigma$ is a weighted average between $\operatorname{tr}\left[\hat{\beta}_w^{-1}\hat{\alpha}_w\right]$ and $\frac{\eta^2}{4} - \frac{\nabla L(w)^2}{S(w)^2}$, so $\operatorname{tr}[\Sigma] \leq \frac{\eta^2}{4} - \frac{\nabla L(w)^2}{S(w)^2}$ if and only if $\operatorname{tr}\left[\hat{\beta}_w^{-1}\hat{\alpha}_w\right] \leq \frac{\eta^2}{4} - \frac{\nabla L(w)^2}{S(w)^2}$. This first expression is just $\operatorname{tr}\Sigma\big|_{\beta_2=1}$ and the second is just $\operatorname{tr}\Sigma\big|_{\beta_2=0}$. □

The condition $\Sigma(t) \succ 0$ is equivalent to requiring that we are not currently at a "breakpoint," i.e. a time t at which an eigenvalue drops from EOS. The set of such t constitute a measure zero set, so the above lemma holds for almost all t.

C.4.2 THE SMALL β_2 CORRECTION

The following lemma shows that the $\beta_2 \to \frac{1-\beta_2}{\beta_2}$ correction for the RMSProp-Norm and RMSProp central flows allows the continuous EMA to match the discrete EMA for linear targets $f(t)$:

Lemma 4. *Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous time process with $|f^{(2)}(t)| \leq \Delta$ for all t. Then if*

$$\begin{aligned}
\nu_t &= \beta_2\nu_{t-1} + (1-\beta_2)f(t) \\
\nu'(t) &= \frac{1-\beta_2}{\beta_2}[f(t) - \nu(t)]
\end{aligned}$$

we have that for all integers $t \geq \tilde{O}((1-\beta_2)^{-1})$,

$$|\nu(t) - \nu_t| \leq O\left(\frac{\beta_2\Delta}{(1-\beta_2)^2}\right).$$

Proof. First define

$$\delta_t := \nu_t - f(t) - \frac{\beta_2}{1-\beta_2} f'(t)$$

$$\delta(t) := \nu(t) - f(t) - \frac{\beta_2}{1-\beta_2} f'(t).$$

Then we have that:

$$\begin{aligned}
\delta_t &= \beta_2 \nu_{t-1} + (1-\beta_2) f(t) - f(t) - \frac{1-\beta_2}{\beta_2} f'(t) \\
&= \beta_2 \nu_{t-1} - \beta_2 f(t) - \frac{\beta_2}{1-\beta_2} f'(t) \\
&= \beta_2 \delta_{t-1} - \beta_2 [f(t) - f(t-1)] - \frac{\beta_2}{1-\beta_2} f'(t) + \frac{\beta_2^2}{1-\beta_2} f'(t-1) \\
&= \beta_2 \delta_{t-1} - \beta_2 [f(t) - f'(t) - f(t-1)] + O\left(\frac{\beta_2^2 \Delta}{1-\beta_2}\right) \\
&= \beta_2 \delta_{t-1} + O\left(\frac{\beta_2 \Delta}{1-\beta_2}\right).
\end{aligned}$$

Therefore,

$$\delta_t = \beta_2^t \delta_0 + O\left(\frac{\beta_2 \Delta}{(1-\beta_2)^2}\right).$$

Similarly,

$$\begin{aligned}
\delta'(t) &= \frac{1-\beta_2}{\beta_2} [f(t) - \nu(t)] - f'(t) - \frac{\beta_2}{1-\beta_2} f''(t) \\
&= -\frac{1-\beta_2}{\beta_2} \delta(t) + O\left(\frac{\beta_2 \Delta}{1-\beta_2}\right)
\end{aligned}$$

so

$$\delta(t) = e^{-\frac{1-\beta_2}{\beta_2} t} \delta(0) + O\left(\frac{\beta_2^2 \Delta}{(1-\beta_2)^2}\right).$$

Therefore subtracting the bounds on δ_t and $\delta(t)$ gives:

$$|\nu_t - \nu(t)| \lesssim \frac{\beta_2 \Delta}{(1-\beta_2)^2}$$

as desired. $\square$

C.5 RMSPROP

We first motivate the stability condition $\lambda_{max}(P^{-1}H) \le 2$ on a quadratic. First, note that $P^{-1}H$ is similar to $P^{-1/2} H P^{-1/2}$ which is symmetric so $P^{-1}H$ has real eigenvalues and real eigenvectors. Next, consider gradient descent on the quadratic $\frac{1}{2} w^T H w$ with preconditioner P. The update is:

$$w \leftarrow w - P^{-1} H w = (I - P^{-1} H) w.$$

Therefore, $w_t = (I - P^{-1}H)^t w_0$. If the eigenvalues of $P^{-1}H$ are in the range $(0, 2)$, then $w \to 0$. Otherwise, it diverges along the corresponding *right* eigenvectors of $P^{-1}H$. We can reinterpret this stability condition using the equivalent condition $H \preceq 2P$. Note that the top right eigenvectors of $P^{-1}H$ correspond to the top eigenvectors for the generalized eigenvalue problem $Hv = \lambda P v$.

We again follow a similar derivation to gradient descent. We define the critical subspace by $U(w, \nu) := \ker[H - 2P]$ where $P = \operatorname{diag}(\sqrt{\nu}/\eta)$, and use $\operatorname{Sym}(U(w, \nu))$ to denote the set of symmetric matrices on this subspace. We can now proceed with the time-averaging argument.

As before, we assume $w = \overline{w} + \delta$ with $\mathbb{E}[\delta] = 0$. Rather than expanding the squared gradient norm, we expand the element wise squared gradient:

$$\mathbb{E}[\nabla L(w)^{\odot 2}] \approx \nabla L(\overline{w})^{\odot 2} + \mathbb{E}[(H(\overline{w})\delta)^{\odot 2}] \tag{22}$$

$$= \nabla L(\overline{w})^{\odot 2} + \operatorname{diag}\left[H(\overline{w})\Sigma H(\overline{w})\right] \tag{23}$$

where $\Sigma := \mathbb{E}[\delta\delta^T]$. This can be further simplified using the fact that Σ is in the top eigenspace, so $H(\overline{w})\Sigma = 2P\Sigma$. Therefore,

$$\operatorname{diag}\left[H(\overline{w})\Sigma H(\overline{w})\right] = 4P \operatorname{diag}[\Sigma] P = \frac{4\nu}{\eta^2} \odot \operatorname{diag}[\Sigma].$$

This suggests the central flow ansatz:

$$\begin{aligned} \frac{dw}{dt} &= -\frac{\eta}{\sqrt{\nu}} \odot \left[\nabla L(w) + \tfrac{1}{2}\nabla^3 L(w)[\Sigma(t)]\right] \\ \frac{d\nu}{dt} &= \frac{1-\beta_2}{\beta_2}\left[\nabla L(w)^{\odot 2} + \frac{4\nu}{\eta^2} \odot \operatorname{diag}[\Sigma(t)] - \nu\right]. \end{aligned} \tag{24}$$

As for gradient descent and RMSProp-Norm, this immediately implies a differential variational inequality definition:

Definition 6 (RMSProp Central Flow, Differential Variational Inequality Formulation)**.** 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. (24) along with the conditions: $\Sigma(t) \succeq 0$, $H(w(t)) \preceq 2P(t)$, and $\langle H(w(t)) - 2P(t), \Sigma(t)\rangle = 0$ where $P(t) = \sqrt{\nu(t)}/\eta$.

We will now show that Σ can be computed as the solution to a cone complementarity problem. Fix some time t, let $U = U(w, \nu)$, and define the linear operator $\mathcal{T} : \operatorname{Sym}(U) \to \mathbb{R}^d$ by $\mathcal{T}[\Sigma] = \nabla_w \langle H(w), \Sigma\rangle$. We can differentiate the stability condition in the critical subspace as:

$$\begin{aligned} \dot{H} &:= \frac{d}{dt}(H(w) - 2P)\bigg|_U \\ &= \mathcal{T}^T\left[\frac{dw}{dt}\right] - 2\frac{\partial P}{\partial \nu}\frac{d\nu}{dt}\bigg|_U \\ &= \mathcal{T}^T P^{-1}\left[-\nabla L(w) - \tfrac{1}{2}\mathcal{T}[\Sigma]\right] \\ &\quad + \frac{1-\beta_2}{\beta_2}\operatorname{diag}\left[\frac{1}{\eta\nu^{1/2}} \odot \left[\nu - \nabla L(w)^{\odot 2} - \frac{4\nu}{\eta^2} \odot \operatorname{diag}[\Sigma]\right]\right]\bigg|_U. \end{aligned}$$

We again group the constant terms and the terms linear in Σ. Define:

$$\begin{aligned} \alpha(w, P) &:= \mathcal{T}^T[-P^{-1}\nabla L(w)] + \frac{1-\beta_2}{\beta_2}P^{-1}\left[P^2 - \frac{\operatorname{diag}[\nabla L(w)^{\odot 2}]}{\eta^2}\right]\bigg|_U \\ \beta(w, P)[\Sigma] &:= \tfrac{1}{2}\mathcal{T}^T P^{-1}\mathcal{T}[\Sigma] + \frac{1-\beta_2}{\beta_2}\frac{4}{\eta^2}P \operatorname{diag}[\Sigma]\bigg|_U. \end{aligned}$$

Note that we chose to define β through its action on Σ to avoid unnecessarily complicated tensor notation. Then $\dot{H} = \alpha - \beta[\Sigma]$. As in the derivations of gradient descent and RMSProp-Norm, we require that Σ solves the CCP:

$$\Sigma \succeq 0, \quad \dot{H} \preceq 0, \quad \left\langle \Sigma, \dot{H} \right\rangle = 0. \tag{25}$$

Together these define the CCP formulation, which can be efficiently simulated:

Definition 7 (RMSProp Central Flow, CCP Formulation)**.** 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. (24) with $\Sigma(t) \in CCP(\alpha(w(t), P(t)), \beta(w(t), P(t))$ where $P(t) = \operatorname{diag}[\sqrt{\nu(t)}/\eta]$.

C.6 SOLVING FOR THE STATIONARY ν IN RMSPROP

Setting $\frac{d\nu}{dt} = 0$ in eq. (24) gives:

$$\nu = \nabla L(w)^{\odot 2} + (4/\eta^2)\nu \odot \operatorname{diag}[\Sigma]. \tag{26}$$

In addition, by complementarity we know that

$$\Sigma\big(H(w) - 2\operatorname{diag}[\sqrt{\nu}/\eta]\big) = 0. \tag{27}$$

Proposition 2. *If w, ν satisfy eqs.* (26) *and* (27)*, then $P := \operatorname{diag}(\sqrt{\nu}/\eta)$ minimizes the convex program eq.* (12)*.*

Proof. Let $p = \sqrt{\nu}/\eta$. The convex program eq. (12) can be written as:

$$\min_p \sum_i p_i + \frac{\nabla L(w)_i^2}{p_i} \quad \text{such that} \quad H(w) \preceq 2\operatorname{diag}(p). \tag{28}$$

If $\hat{\Sigma}$ is the dual variable for the semidefinite constraint, the KKT conditions for this program are:

$$1 - \frac{\nabla L(w)_i^2}{p_i^2} - 2\hat{\Sigma}_{ii} = 0 \quad \forall i, \quad \hat{\Sigma}(H(w) - 2\operatorname{diag}(p)) = 0. \tag{29}$$

We will prove that $(p, \hat{\Sigma}) = (\frac{\sqrt{\nu}}{\eta}, \frac{2}{\eta^2}\Sigma)$ solve the KKT conditions eq. (29). Note that the complementary slackness KKT condition is equivalent to the complementarity condition on Σ. In addition, dividing the stationarity condition for ν (eq. (26)) by ν gives

$$1 - \frac{\nabla L(w)^{\odot 2}}{\nu_i} - (4/\eta^2)\Sigma_{ii} = 0 \quad \forall i \tag{30}$$

which is equivalent to the first condition in eq. (29) after substituting $\hat{\Sigma} = \frac{2}{\eta^2}\Sigma(t)$. □

Proposition 3. *For any g, H, the solution to eq.* (12) *is unique.*

Proof. Assume there are two minimizers P, P' and let $p := \operatorname{diag}(P), \delta := \operatorname{diag}(P' - P)$. Then by convexity, $\operatorname{diag}[p + \epsilon\delta]$ also minimizes eq. (12) for any $\epsilon \le 1$. Therefore, differentiating the objective function in this direction gives:

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

Taking another derivative implies that:

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

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 denote the vector p restricted to the indices in I. Define the linear map g by

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

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

$$\mathcal{A}^T[v_I] = \operatorname{diag}[g[v_I]] \oplus \operatorname{diag}[v_I].$$

Then both p_I, p'_I minimize the following reduced SDP:

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

Now we apply (de Carli Silva & 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 & Tunçel, 2018, Corollary 2) Therefore $p_I = p'_I$, and as we have already shown equality on I^c, we must have $p = p'$. □

C.7 ARBITRARY PRECONDITIONED METHODS

In this section, we derive a central flow for an abstract preconditioned method. This general central flow will reduce to our central flows for gradient descent, RMSProp-Norm, and RMSProp. However, we emphasize that this paper does not claim that the central flow derived in this section will be correct for any preconditioned method. We include this section both because it allows us 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), and because we hope it can be a starting point for others to derive central flows for other optimizers.

We will let ν be the "state" of the preconditioner, and let P be a function that maps the state ν to a symmetric, positive definite matrix. Specifically, we consider the discrete update:

$$\nu_t = \nu_{t-1} + f(\nu_{t-1}, w_t), \quad w_{t+1} = w_t - P(\nu_t)^{-1}\nabla L(w_t).$$

This formulation is very general and includes a wide variety of updates including:

- *GD with a learning rate schedule $\eta(t)$:* Set $f(\nu, w) = 1$ so that $\nu(t) = t$, and $P(t) = \eta(t)^{-1} I$
- *Vanilla RMSProp:* Set $f(\nu, w) = (1-\beta_2)[\nabla L(w)^{\odot 2} - \nu]$ and $P(\nu) = \operatorname{diag}[\sqrt{\nu}/\eta]$
- *RMSProp with ϵ, bias correction, and learning rate schedule $\eta(t)$:* Set $\nu = [v, t]$, $f([v,t], w) = [(1-\beta_2)[\nabla L(w)^{\odot 2} - v], 1]$ and define
$$P([v,t]) = \operatorname{diag}\left[\frac{1}{\eta(t)} \cdot \sqrt{\frac{v}{1-\beta_2^t} + \epsilon}\right].$$

Note that this trick of embedding t into the state variable ν allows us to automatically derive central flows for any smooth hyperparameter schedules (e.g. $\eta(t), \beta_2(t), \epsilon(t)$) as a simple corollary.

As for RMSProp, the stability of this algorithm requires $\lambda_{max}(P^{-1}H) \le 2$ or equivalently $H \preceq 2P$. To derive the central flow, we assume that $w = \overline{w} + \delta$ with $\mathbb{E}[\delta] = 0$ and $\mathbb{E}[\delta\delta^T] = \Sigma$. Taylor expanding and time-averaging the update for ν gives:

$$\mathbb{E}[f(\nu, w)] = \mathbb{E}[f(\nu, \nabla L(\overline{w} + \delta))] \approx f(\nu, \nabla L(\overline{w})) + \tfrac{1}{2}\nabla_w^2 f(\nu, \overline{w})[\Sigma]$$

These motivate the central flow ansatz:

$$\begin{aligned} \frac{dw}{dt} &= -P(\nu)^{-1}\left[\nabla L(w) + \tfrac{1}{2}\nabla^3 L(w)[\Sigma]\right] \\ \frac{d\nu}{dt} &= f(\nu, w) + \tfrac{1}{2}\nabla_w^2 f(\nu, w)[\Sigma]. \end{aligned} \tag{31}$$

We say that $\{w(t), \nu(t), \Sigma(t)\}_{t\ge 0}$ satisfy the DVI formulation of the central flow if for almost all $t \ge 0$ they satisfy eq. (31), $\Sigma(t) \succeq 0$, $H(w(t)) \preceq 2P(\nu(t))$, and $\langle \Sigma(t), H(w(t)) - 2P(\nu(t))\rangle$.

To derive the CCP formulation which is efficiently computable, we differentiate the stability condition. Fix an iterate t, let $U := \ker[H - 2P(\nu(t))]$ be the critical subspace at time t, and define $\dot{H} := \frac{dH}{dt}\big|_U$ under eq. (31). Then for the stability condition to remain true, we need $\dot{H} \preceq 0$. To compute it, define $\mathcal{T} : \mathrm{Sym}(U) \to \mathbb{R}^d$ by $\mathcal{T}[\Sigma] := \nabla_w \langle H(w), \Sigma\rangle$. Then,

$$\begin{aligned} \dot{H} &:= \frac{dH}{dt}\bigg|_U \\ &= \mathcal{T}^T\left[\frac{dw}{dt}\right] - 2\nabla_\nu P(\nu)\left[\frac{d\nu}{dt}\right]\bigg|_U \\ &= \mathcal{T}^T P(\nu)^{-1}\left[-\nabla L(w) - \tfrac{1}{2}\mathcal{T}[\Sigma]\right] - 2\nabla_\nu P(\nu)\left[f(\nu, w) + \tfrac{1}{2}\nabla_w^2 f(\nu, w)[\Sigma]\right]\bigg|_U. \end{aligned}$$

Splitting up the constant terms and the terms linear in Σ, we define:

$$\begin{aligned} \alpha(w, \nu) &:= \mathcal{T}^T[-P(\nu)^{-1}\nabla L(w)] - 2\nabla_\nu P(\nu)|_U f(\nu, w) \in \mathrm{Sym}(U) \\ \beta(w, \nu) &:= \tfrac{1}{2}\mathcal{T}^T P(\nu)^{-1}\mathcal{T} + \nabla_\nu P(\nu)|_U \nabla_w^2 f(\nu, w) \in \mathrm{Sym}(U) \otimes \mathrm{Sym}(U). \end{aligned}$$

Then $\dot{H} = \alpha - \beta[\Sigma]$, so by the same arguments as for gradient descent,RMSProp-Norm,RMSProp, Σ needs to solve $CCP(\alpha(w,\nu), \beta(w,\nu))$, and plugging this Σ into eq. (31) gives the CCP formulation for the central flow for this preconditioned method.

C.8 CONE COMPLEMENTARITY PROBLEMS

In this section we will let $k \geq 1$ be a positive integer, and we will use $\mathrm{Sym}_k(\mathbb{R})$ to denote the set of $k \times k$ symmetric matrices.

Definition 8 (Cone Complementarity Problem)**.** Let $\alpha \in \mathrm{Sym}_k(\mathbb{R})$, and let $\beta \in \mathrm{Sym}_k(\mathbb{R}) \otimes \mathrm{Sym}_k(\mathbb{R})$ be a linear operator on symmetric matrices. We say that the matrix $X \in \mathrm{Sym}_k(\mathbb{R})$ solves the cone complementarity problem $\mathrm{CCP}(\alpha, \beta)$ if:

$$X \succeq 0, \quad \alpha - \beta[X] \preceq 0, \quad \langle X, \alpha - \beta[X] \rangle = 0.$$

Lemma 5. *Let $\beta \in \mathrm{Sym}_k(\mathbb{R}) \otimes \mathrm{Sym}_k(\mathbb{R})$ be a symmetric linear operator on symmetric matrices.*

1. *If $\beta \succeq 0$ as an operator on $\mathrm{Sym}_k(\mathbb{R})$, then $CCP(\alpha, \beta)$ has a solution for all α.*
2. *If $\beta \succ 0$, this solution is unique. Otherwise, all solutions X, X' differ by a matrix in the kernel of β.*

Proof. Consider the quadratic program

$$\min_X \tfrac{1}{2}\beta[X, X] - \langle X, \alpha \rangle \quad \text{such that} \quad X \succeq 0.$$

Because $\beta \succeq 0$, this program is convex. The KKT conditions for this program are $\beta[X] - \alpha \succeq 0$ and the complementary slackness condition $X(\beta[X] - \alpha) = 0$. Therefore any KKT point to the original quadratic program, including its global optimum, solve the conic complementarity problem which proves (1). To prove (2), let X, X' be solutions to the conic complementarity program. Then taking the trace of the complementarity relations we have:

$$\langle \alpha, X \rangle - \beta[X, X] = 0, \quad \langle \alpha, X' \rangle - \beta[X', X'] = 0.$$

In addition, because both $\beta[X] - \alpha$ and X' are PSD and vice-versa, we must have

$$\langle X', \beta[X] - \alpha \rangle \leq 0, \quad \langle X, \beta[X'] - \alpha \rangle \leq 0.$$

Adding these four conditions gives:

$$\beta[X, X] + \beta[X', X'] - \beta[X, X'] - \beta[X', X] \leq 0 \iff \beta[X - X', X - X'] \leq 0.$$

However, since $\beta \succeq 0$ this implies that $\beta[X - X', X - X'] = 0$. When $\beta \succ 0$ this implies that $X = X'$, and when $\beta \succeq 0$ this implies that $X - X'$ is in the kernel of β. □

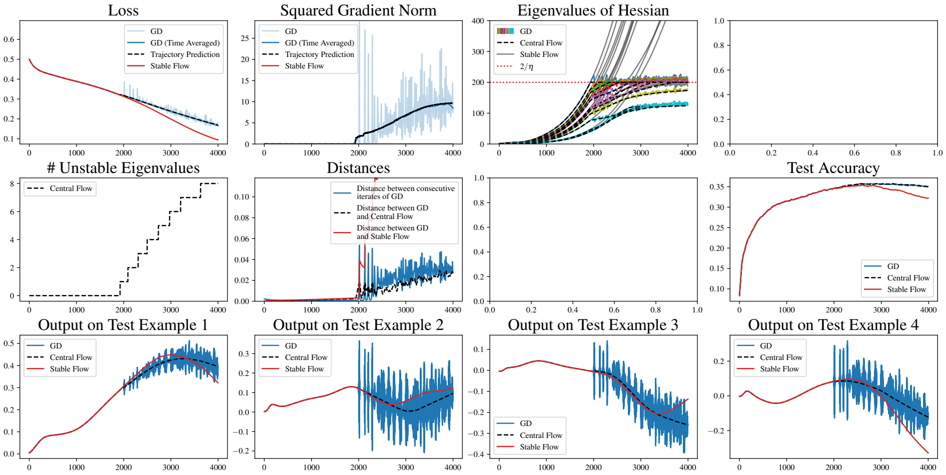
D FULL EXPERIMENTS

Our full set of experiments span a variety of neural network architectures: a CNN, a ResNet (He et al., 2016), a vision transformer (ViT) (Dosovitskiy et al., 2021), a GPT-2 (Radford et al., 2019) style transformer, and a recurrent neural network (RNN). We evaluate the CNN, ResNet, and ViT on an image classification task, and we evaluate the GPT-style transformer and RNN on a sequence prediction task, using both mean-squared-error (MSE) and cross entropy losses. Since discretizing the flows is computationally expensive, we are restricted to small scale datasets: our image dataset is a subset of CIFAR-10 with 1,000 examples and either 4 or 10 classes, while our sequence dataset is a sorting task with 1,000 sequences of length 8 and alphabet size 4. Even at these modest scales, our full set of 115 experiments required 2000 GPU-hours. We further describe the architectural details, and the procedure for discretizing the flows in Appendix B.

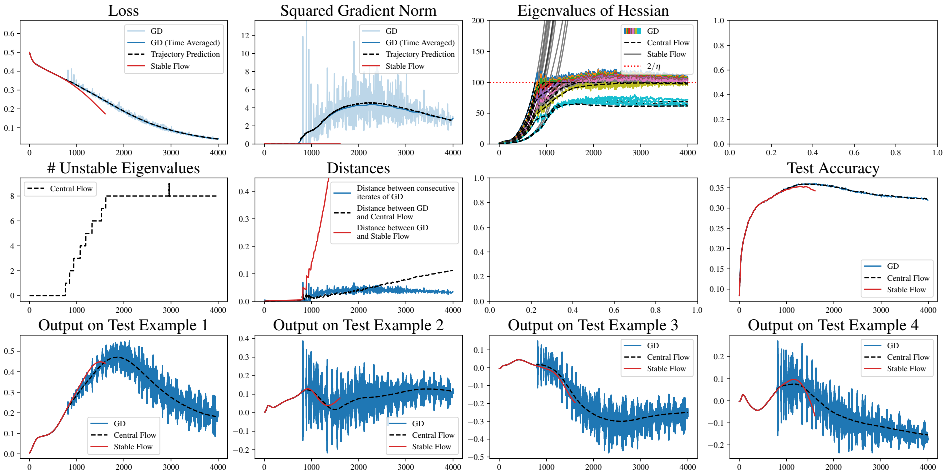
Empirically, the quality of the central flow approximation is usually better for smaller learning rates than large ones, and for MSE loss rather than cross-entropy loss. We believe that some important underlying factors are: (1) the magnitude of the oscillations in sharpness around $2/\eta$; and (2) whether higher-order terms cause the sharpness equilibrium point to differ slightly from $2/\eta$. A promising direction for future research is to rigorously identify conditions under which the central flow accurately approximates the real optimization trajectory.

D.1 GRADIENT DESCENT

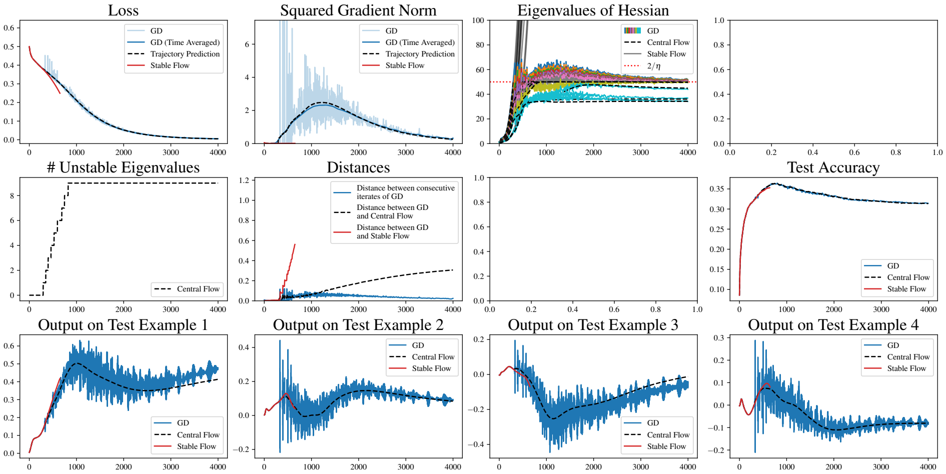
D.1.1 CIFAR 10



(a) $\eta = 0.01$

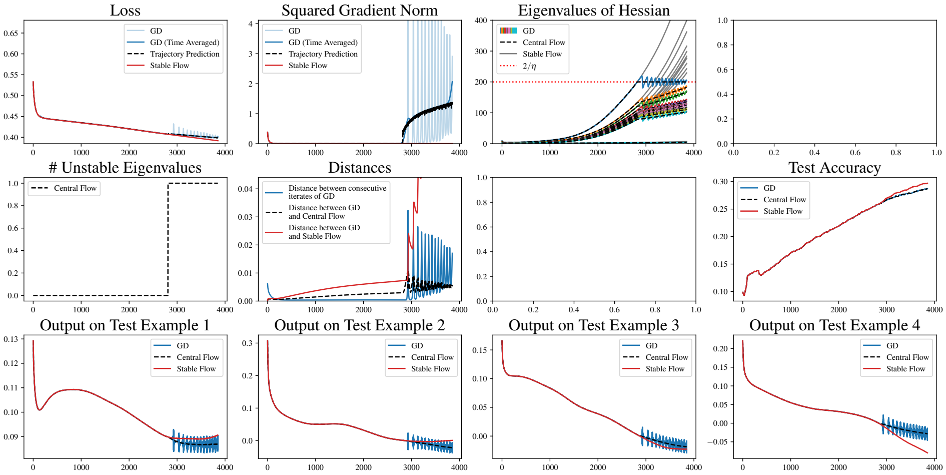


(b) $\eta = 0.02$

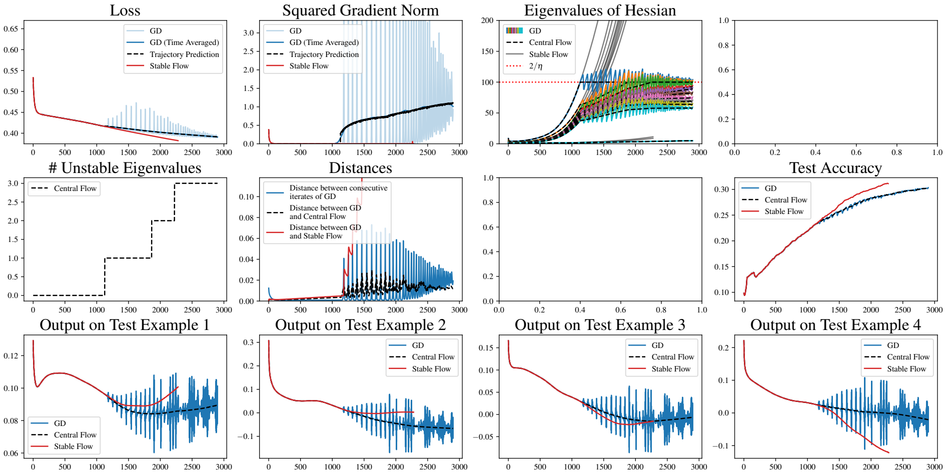


(c) $\eta = 0.04$

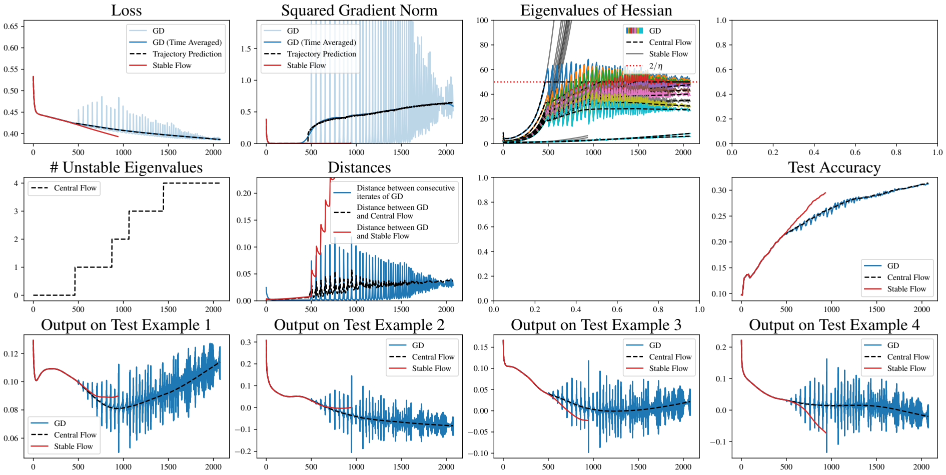
Figure 12: gradient descent on a CNN on a 1000 example subset of CIFAR10 with MSE loss



(a) $\eta = 0.01$

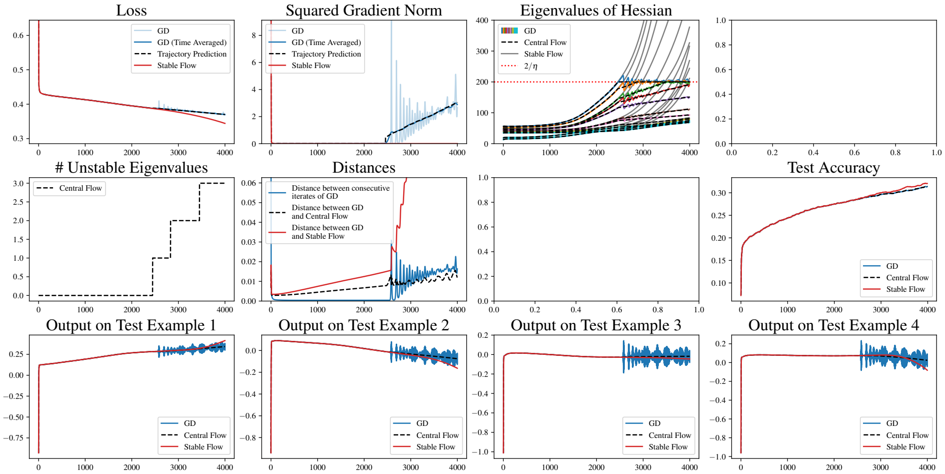


(b) $\eta = 0.02$

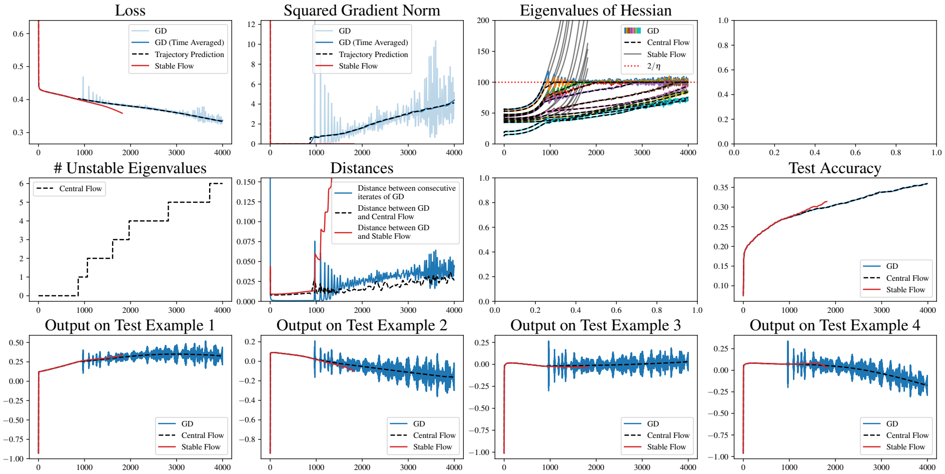


(c) $\eta = 0.04$

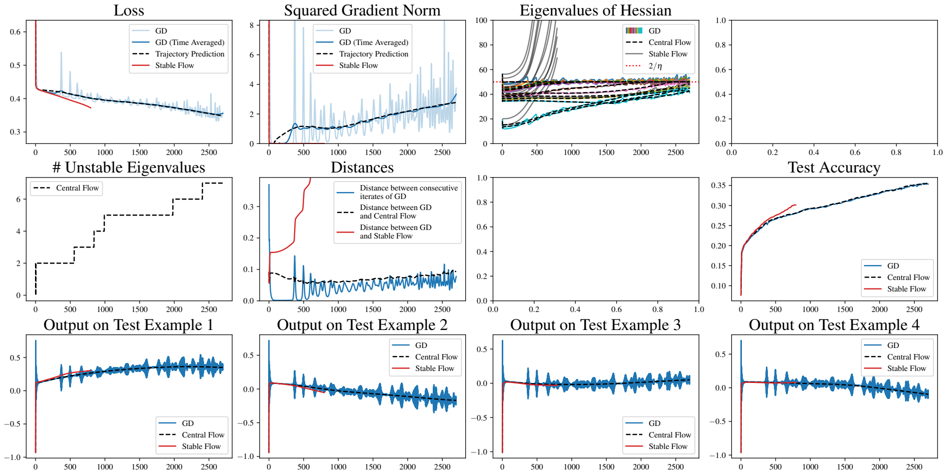
Figure 13: gradient descent on a ResNet on a 1000 example subset of CIFAR10 with MSE loss



(a) $\eta = 0.01$



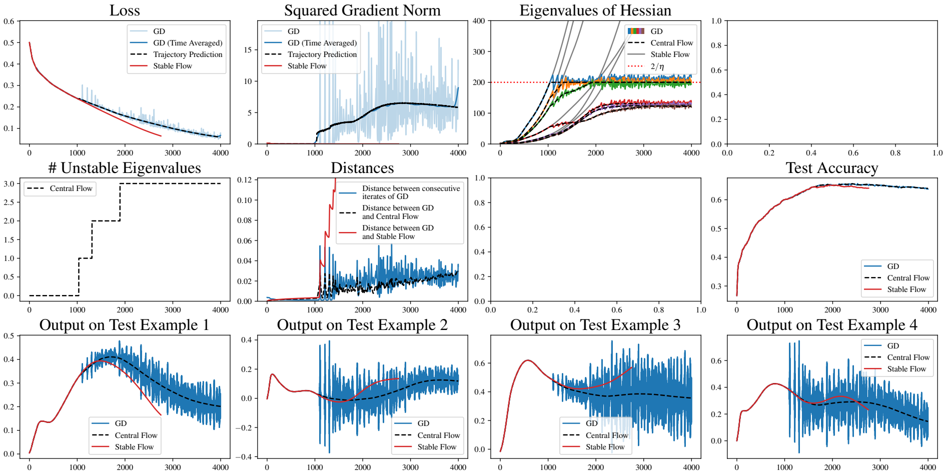
(b) $\eta = 0.02$



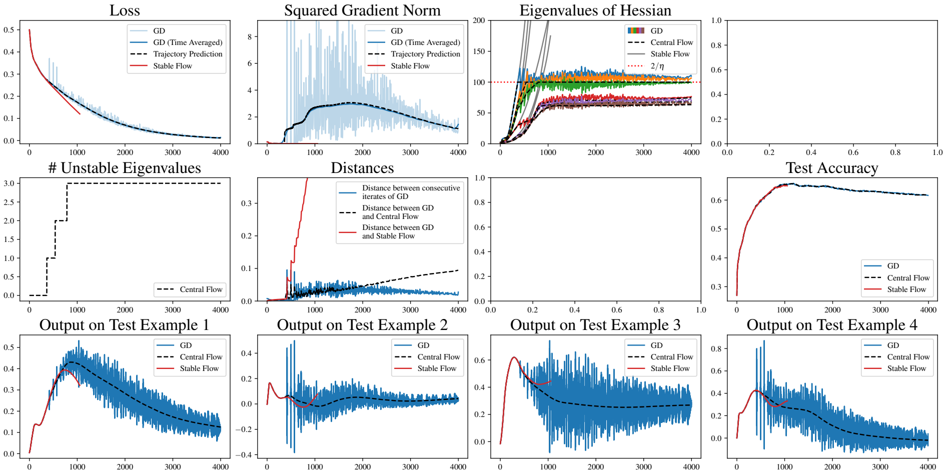
(c) $\eta = 0.04$

Figure 14: gradient descent on a ViT on a 1000 example subset of CIFAR10 with MSE loss

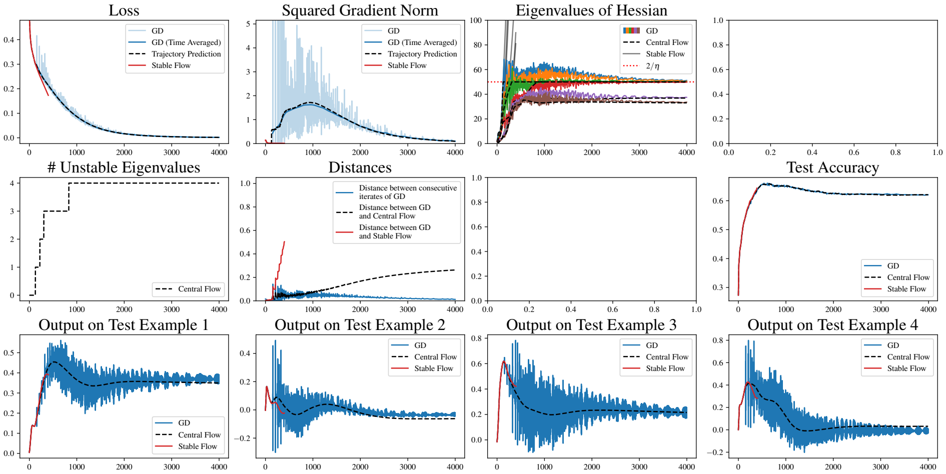
D.1.2 CIFAR 10 (4 CLASS)



(a) $\eta = 0.01$

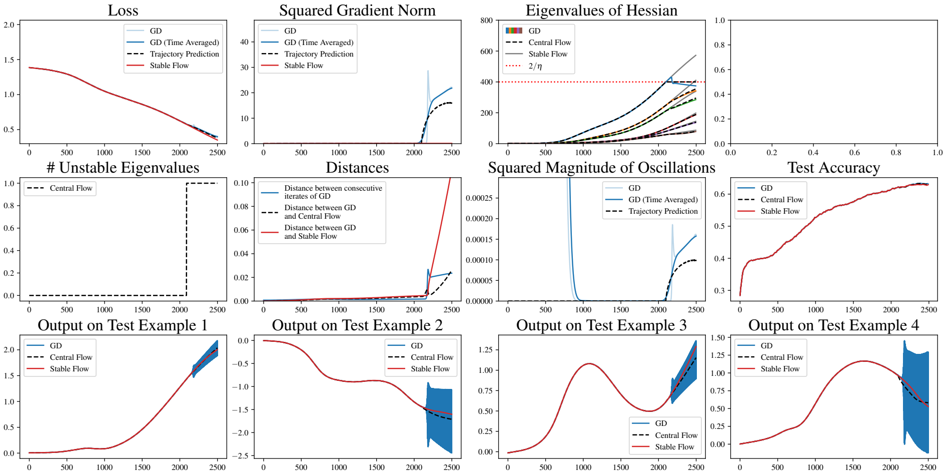


(b) $\eta = 0.02$

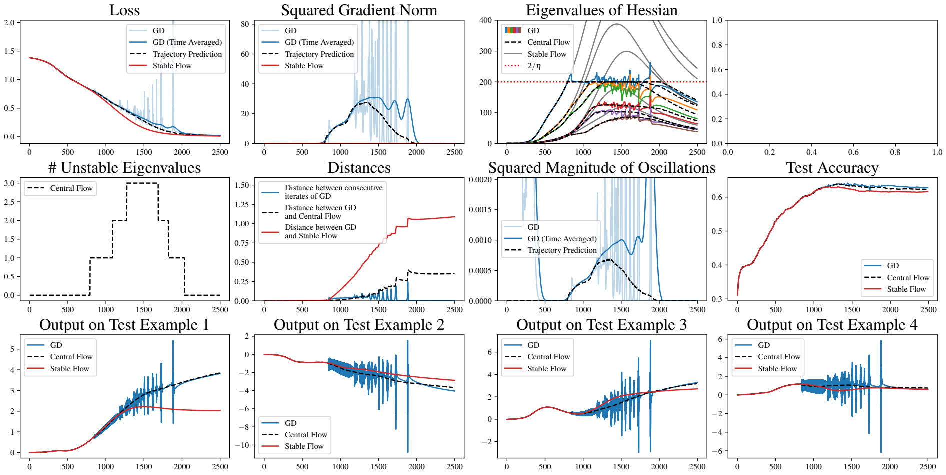


(c) $\eta = 0.04$

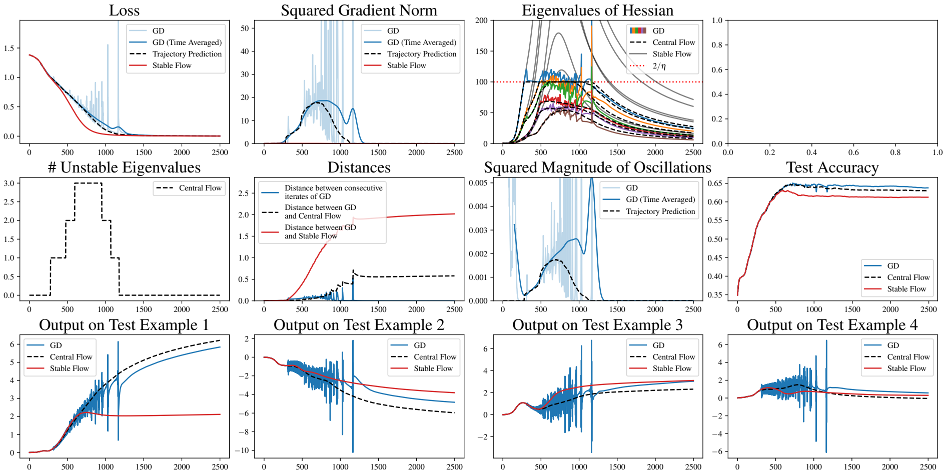
Figure 15: gradient descent on a CNN on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 0.005$

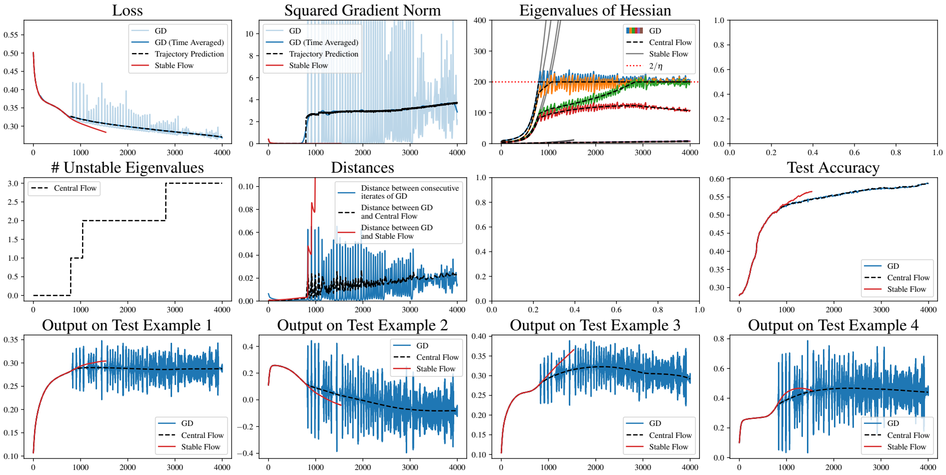


(b) $\eta = 0.01$

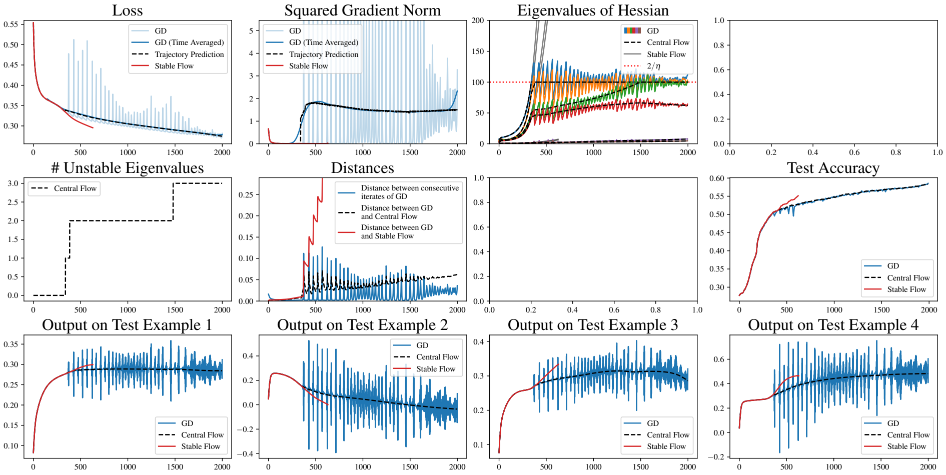


(c) $\eta = 0.02$

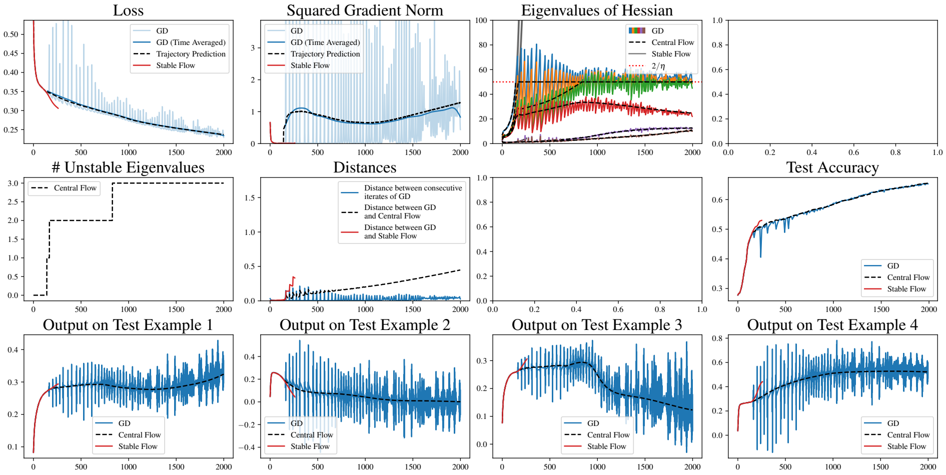
Figure 16: gradient descent on a CNN on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss



(a) $\eta = 0.01$

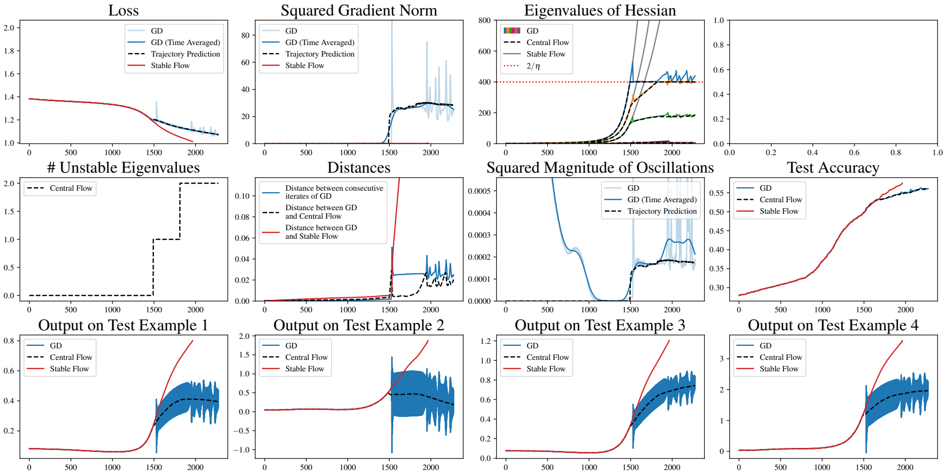


(b) $\eta = 0.02$

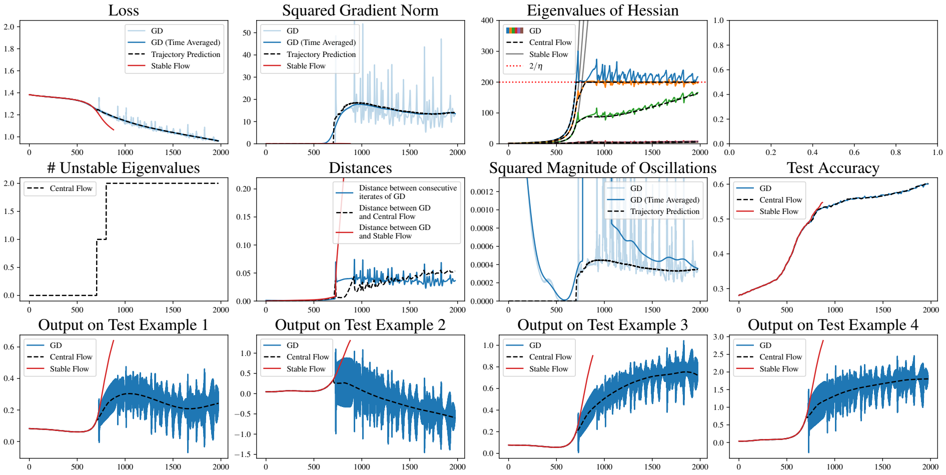


(c) $\eta = 0.04$

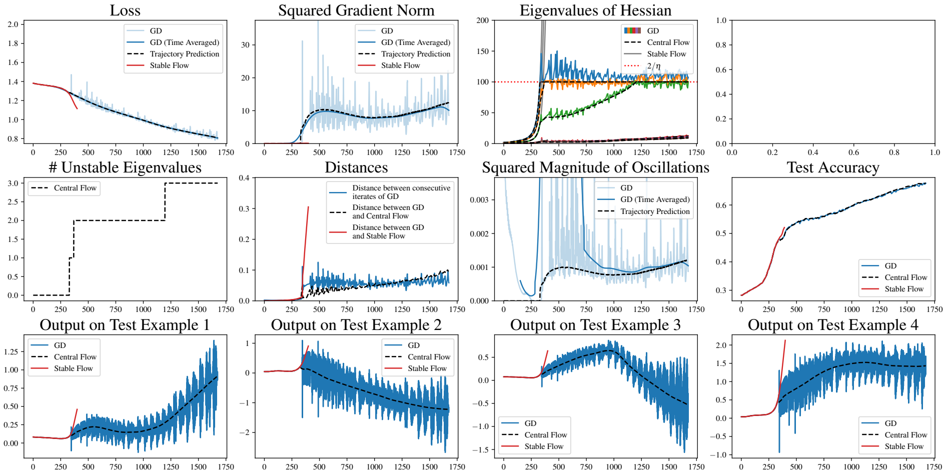
Figure 17: gradient descent on a ResNet on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 0.005$

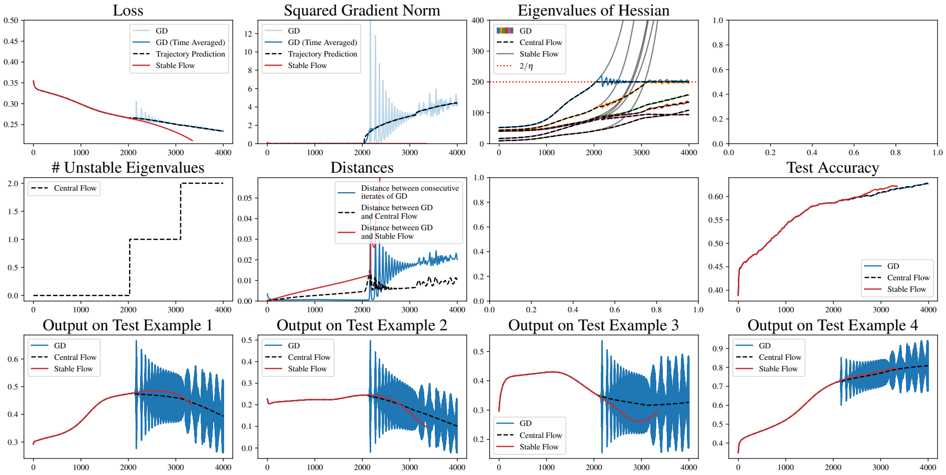


(b) $\eta = 0.01$

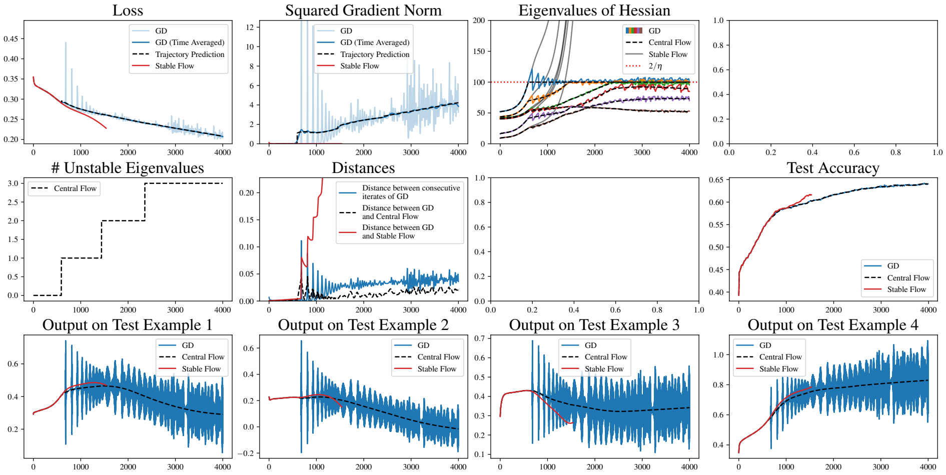


(c) $\eta = 0.02$

Figure 18: gradient descent on a ResNet on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss

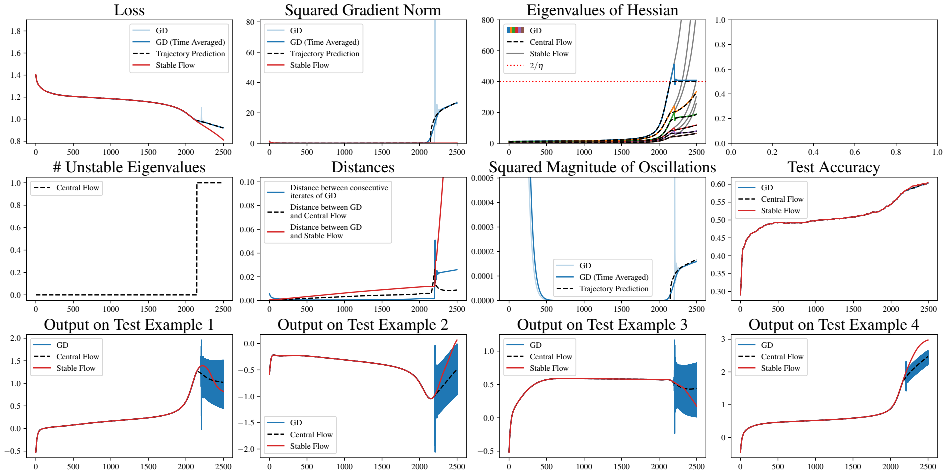


(a) $\eta = 0.01$

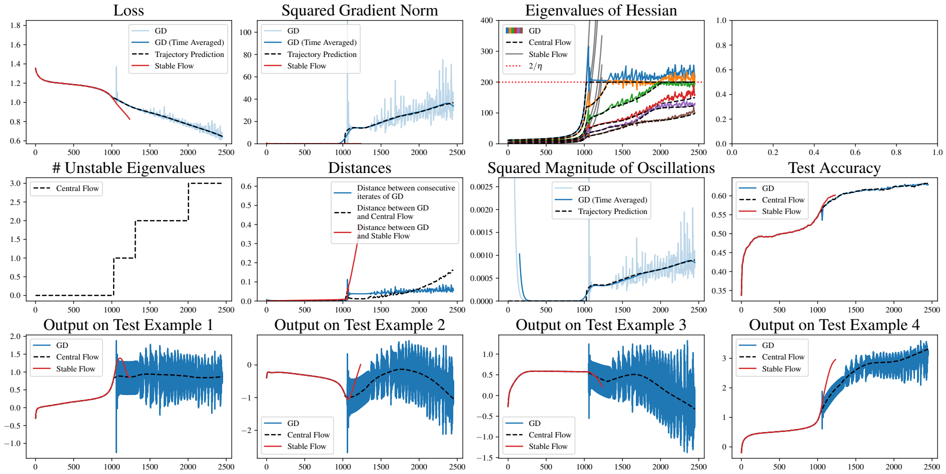


(b) $\eta = 0.02$

Figure 19: gradient descent on a ViT on a 1000 example, 4 class subset of CIFAR10 with MSE loss



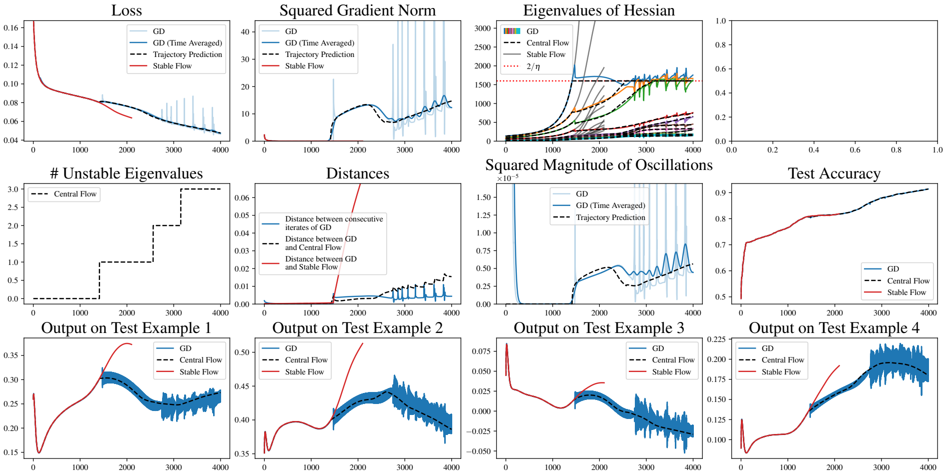
(a) $\eta = 0.005$



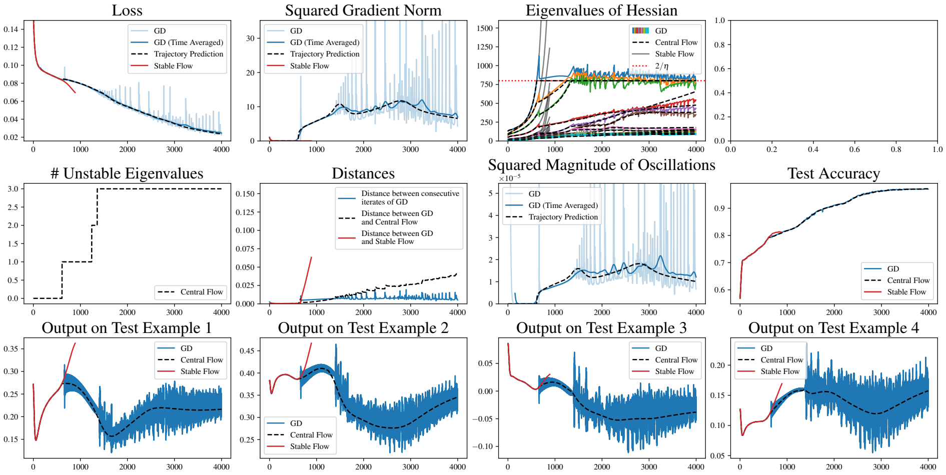
(b) $\eta = 0.01$

Figure 20: gradient descent on a ViT on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss

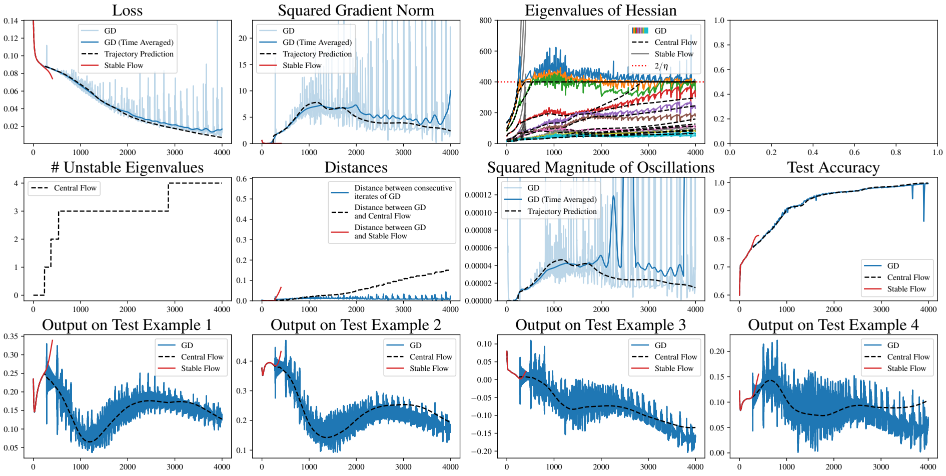
D.1.3 SORTING



(a) $\eta = 0.00125$



(b) $\eta = 0.0025$

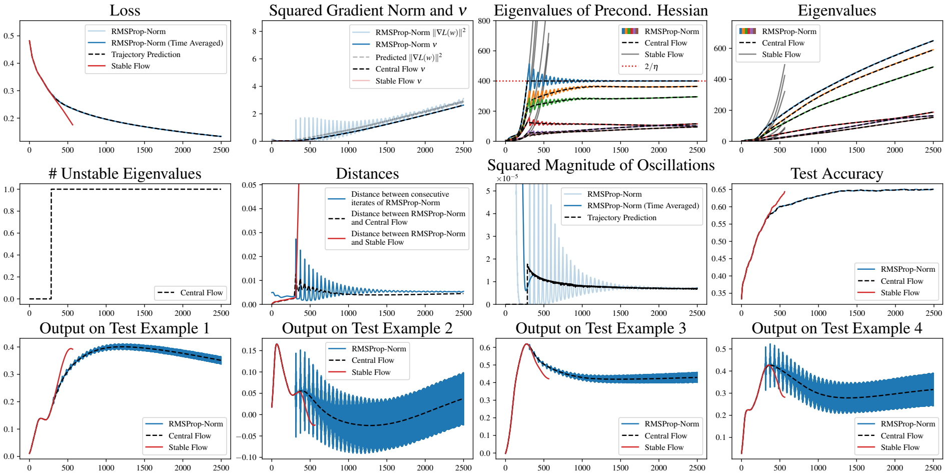


(c) $\eta = 0.005$

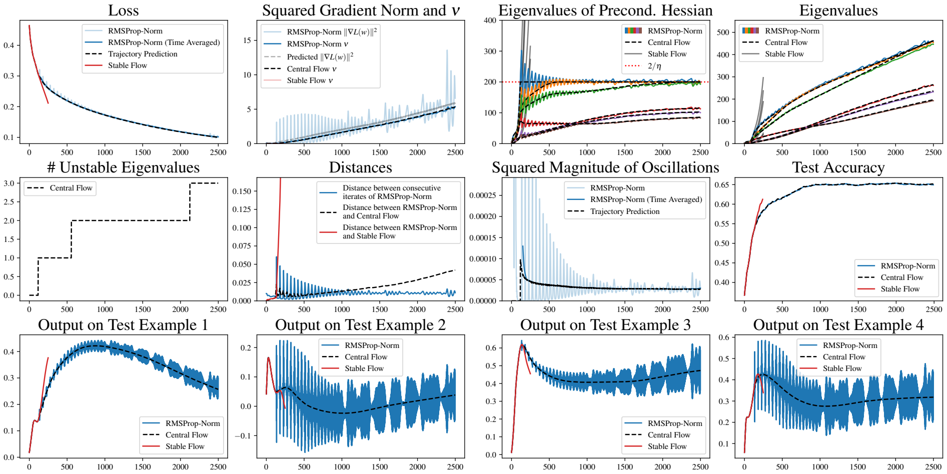
Figure 21: gradient descent on a GPT-style transformer on a synthetic sorting task with MSE loss

D.2 RMSPROP-NORM

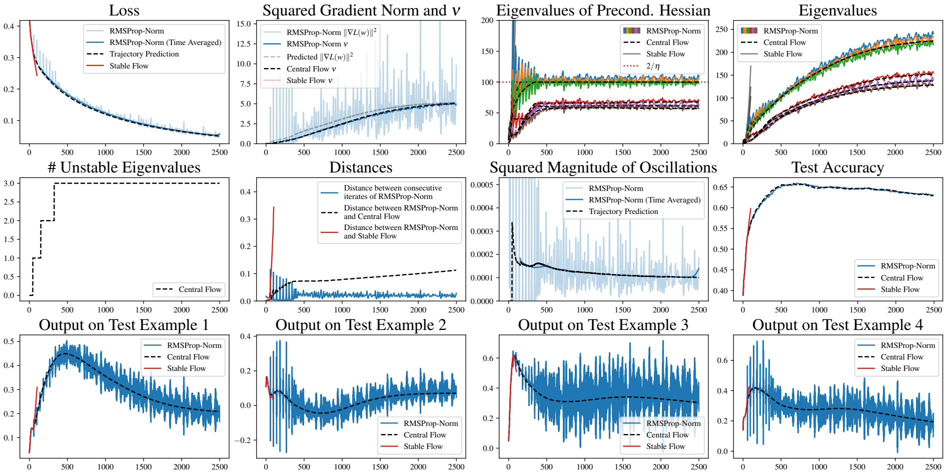
D.2.1 CIFAR 10 (4 CLASS)



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

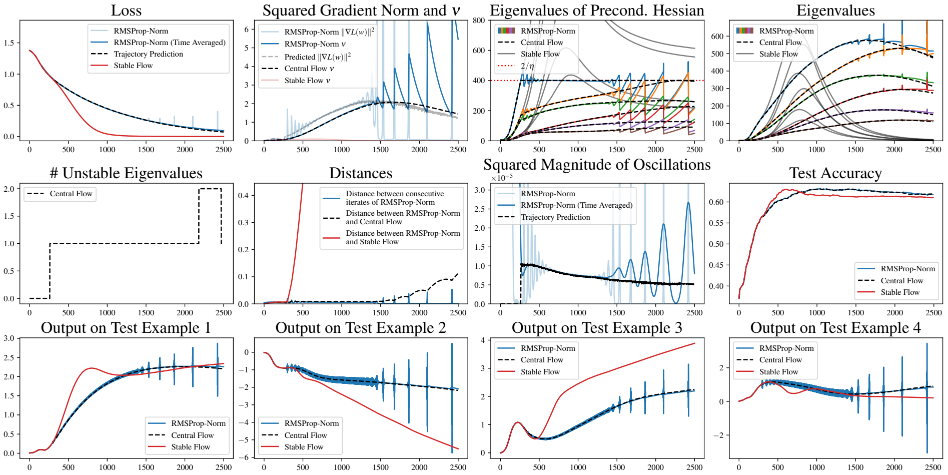


(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

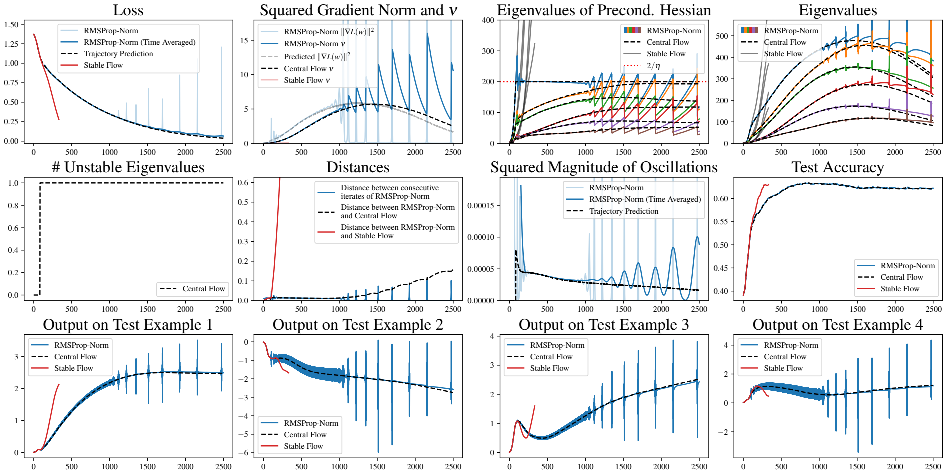


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

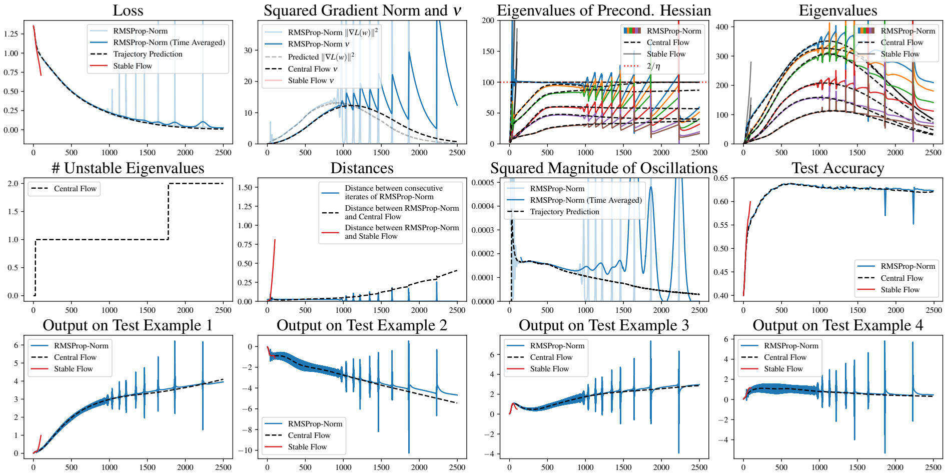
Figure 22: RMSProp-Norm on a CNN on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

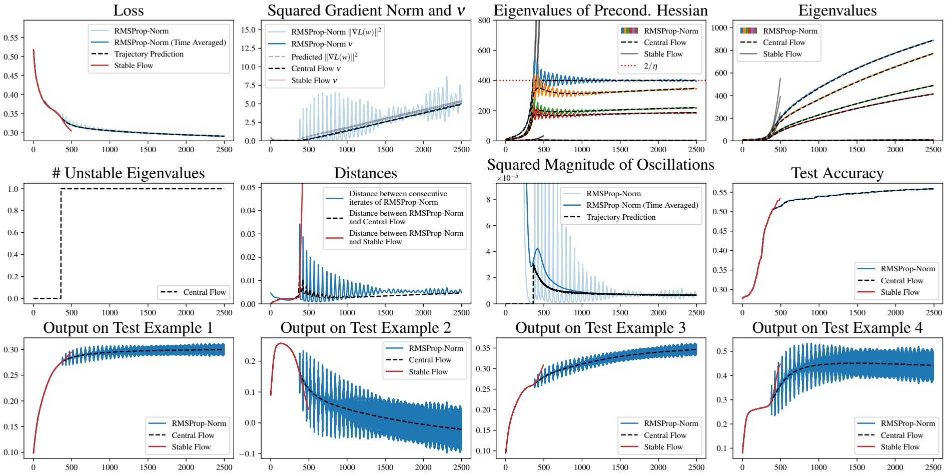


(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

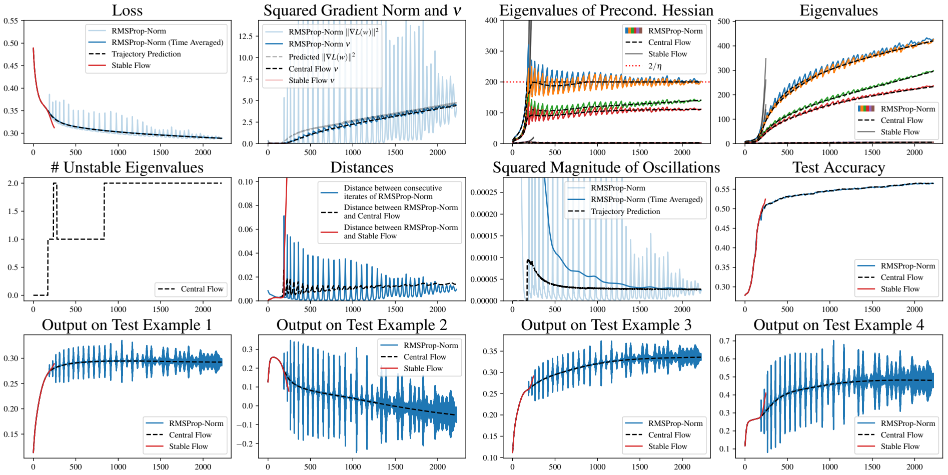


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

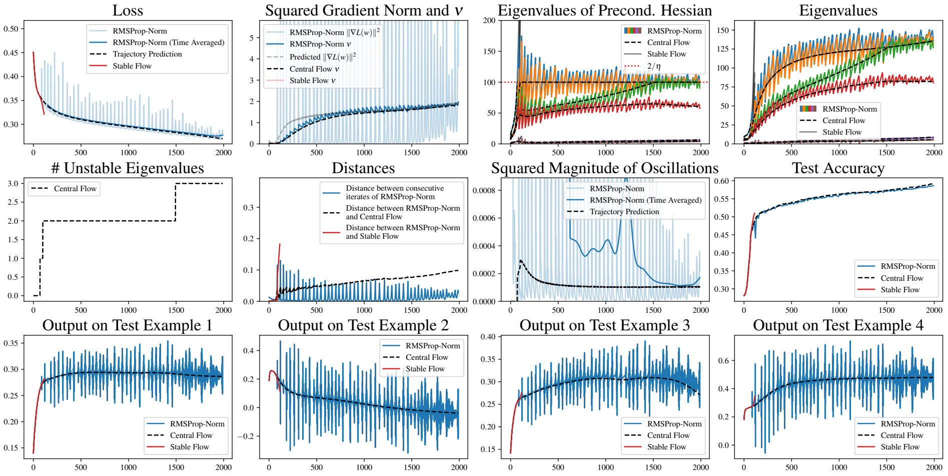
Figure 23: RMSProp-Norm on a CNN on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

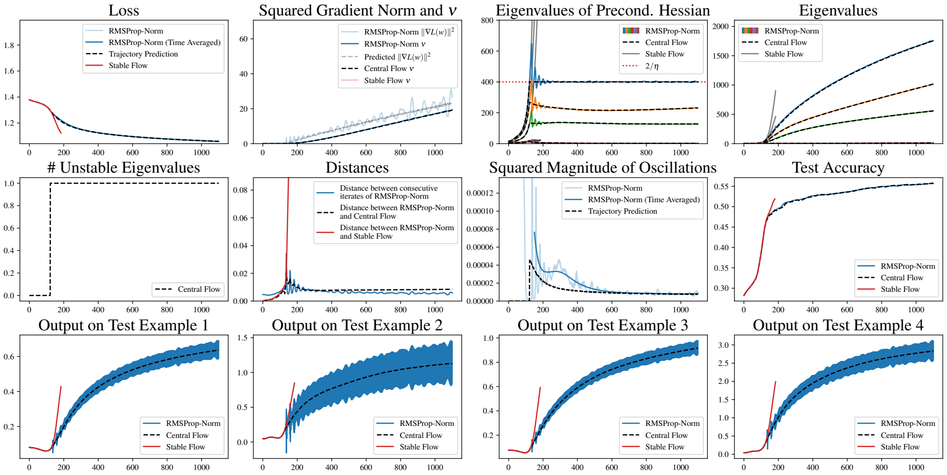


(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

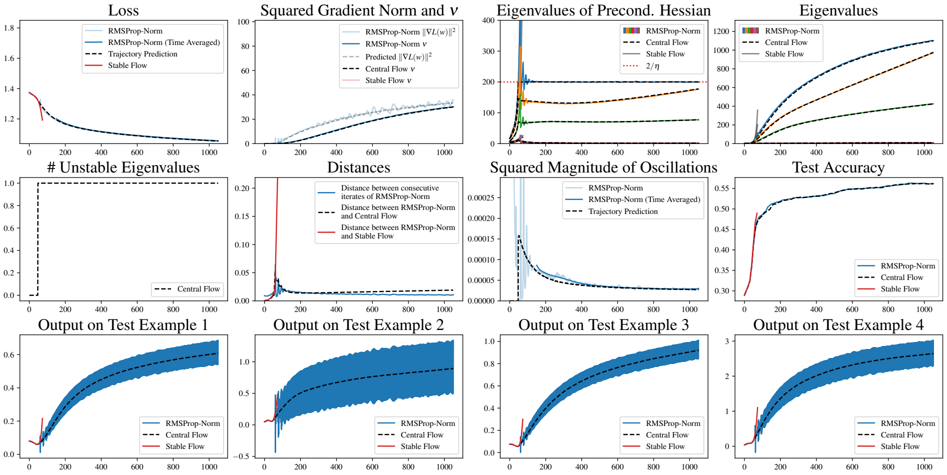


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

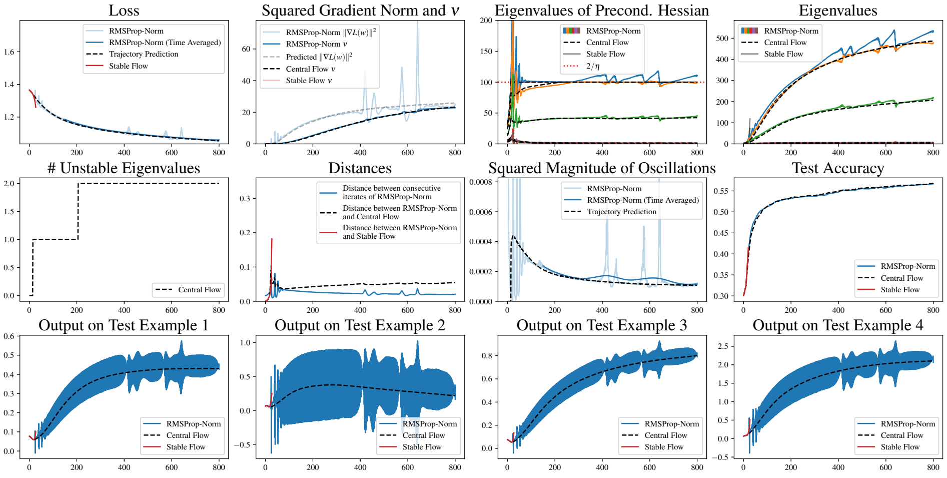
Figure 24: RMSProp-Norm on a ResNet on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

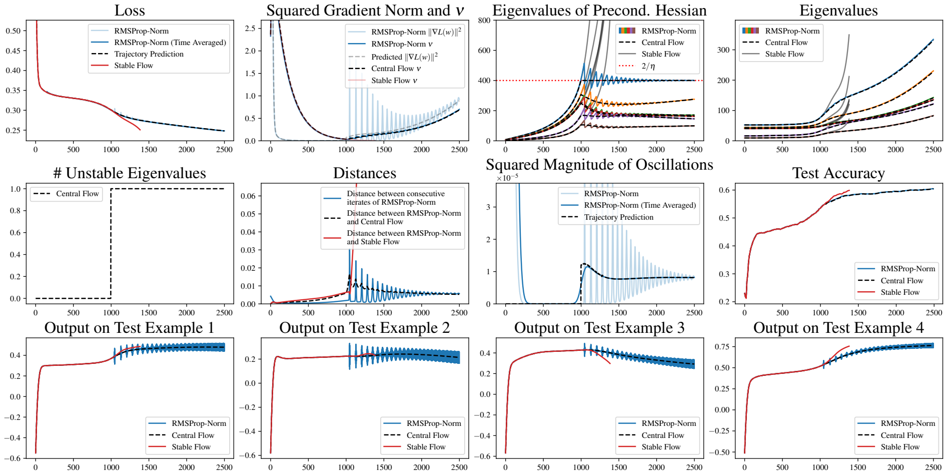


(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

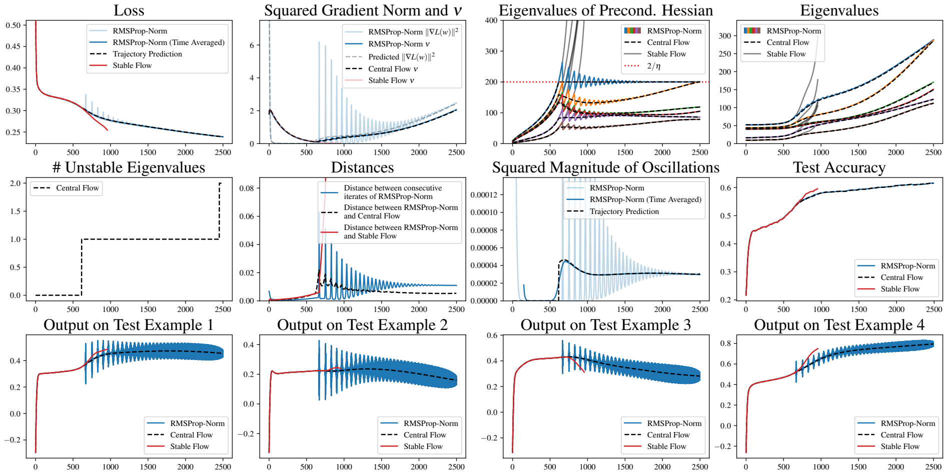


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

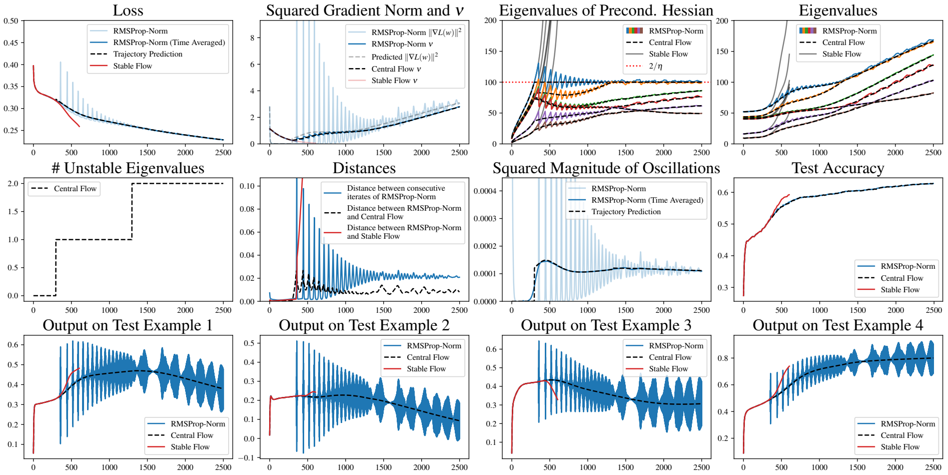
Figure 25: RMSProp-Norm on a ResNet on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

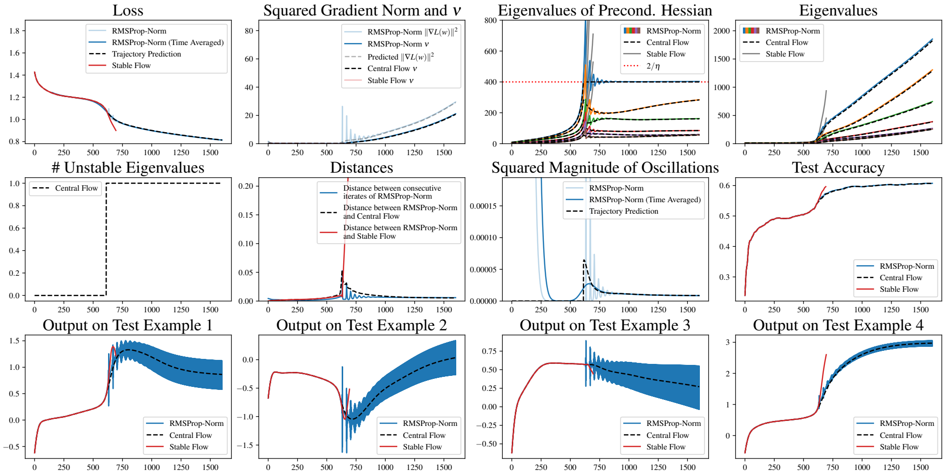


(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

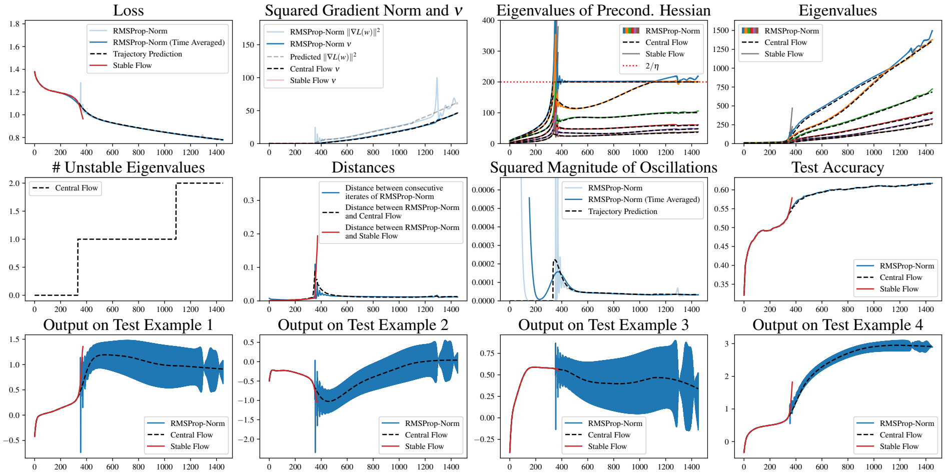


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

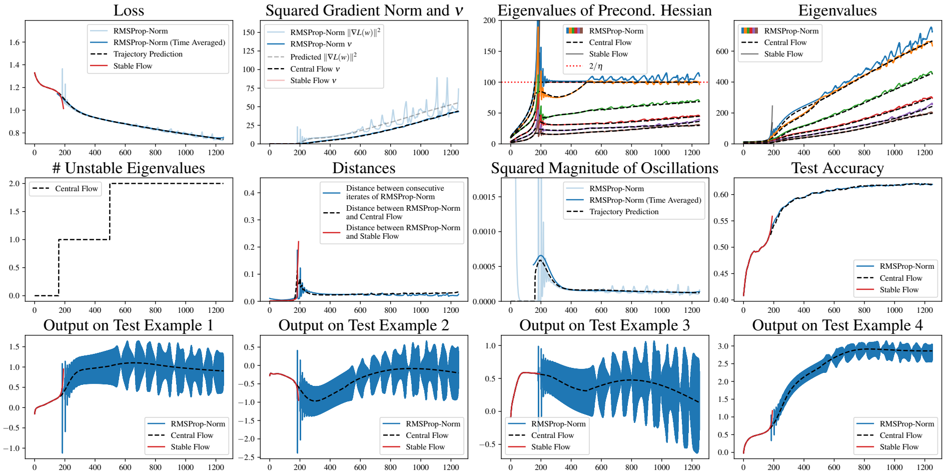
Figure 26: RMSProp-Norm on a ViT on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 0.005$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction



(b) $\eta = 0.01$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

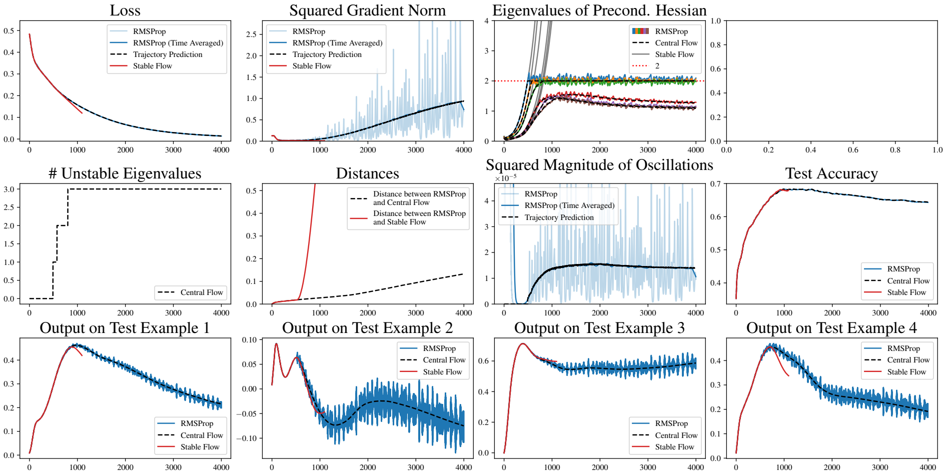


(c) $\eta = 0.02$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

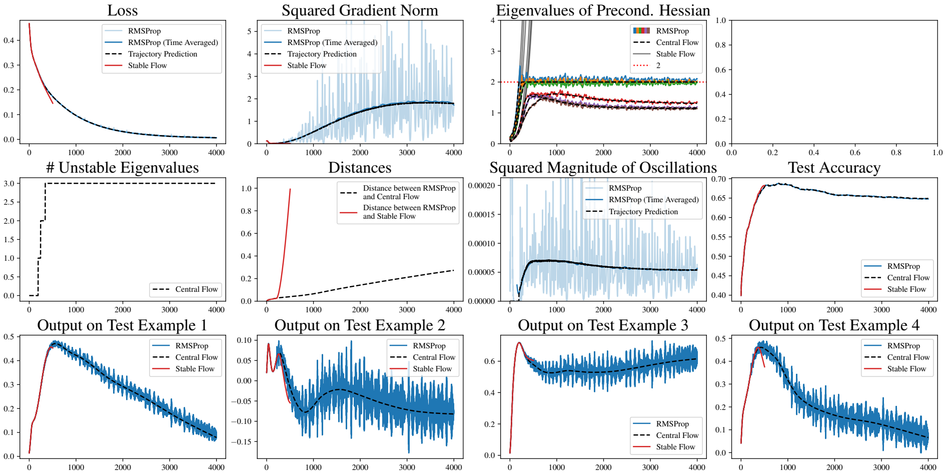
Figure 27: RMSProp-Norm on a ViT on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss

D.3 RMSProp

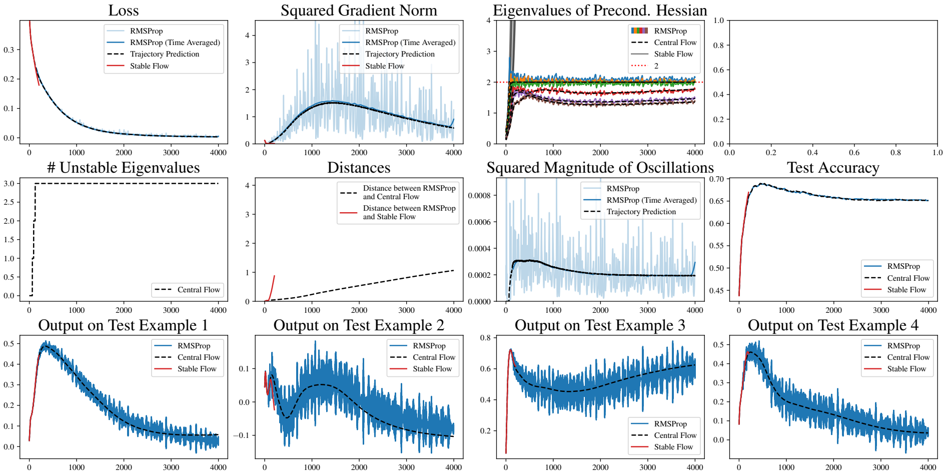
D.3.1 CIFAR 10 (4 class)



(a) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

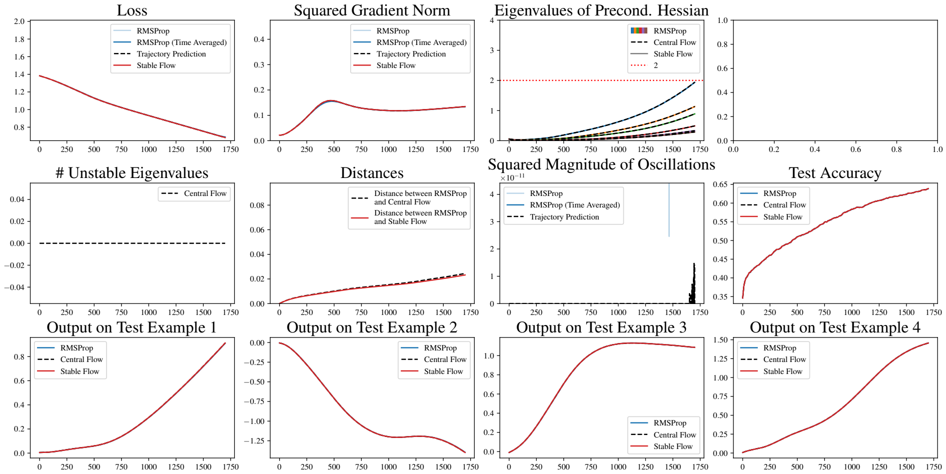


(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

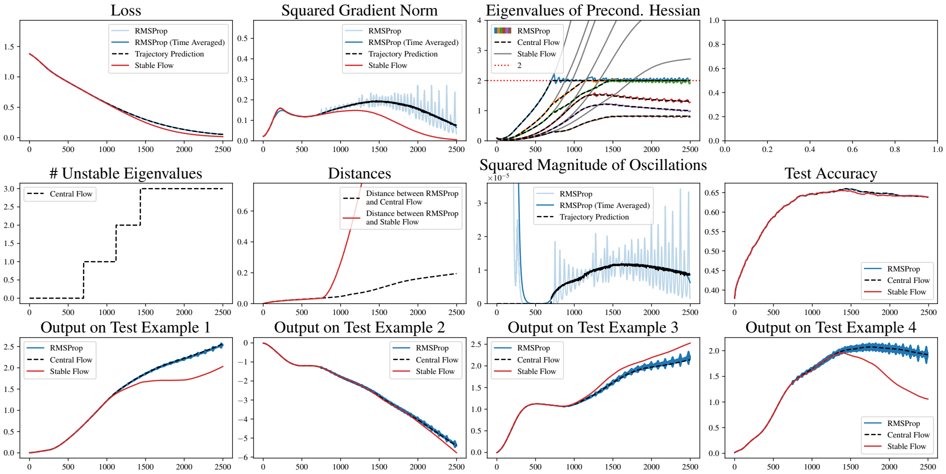


(c) $\eta = 4 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

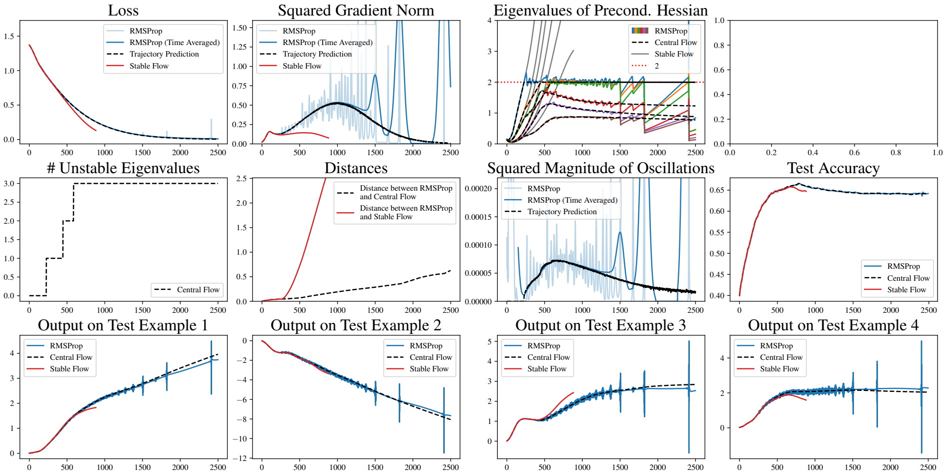
Figure 28: RMSProp on a CNN on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 5 \times 10^{-6}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

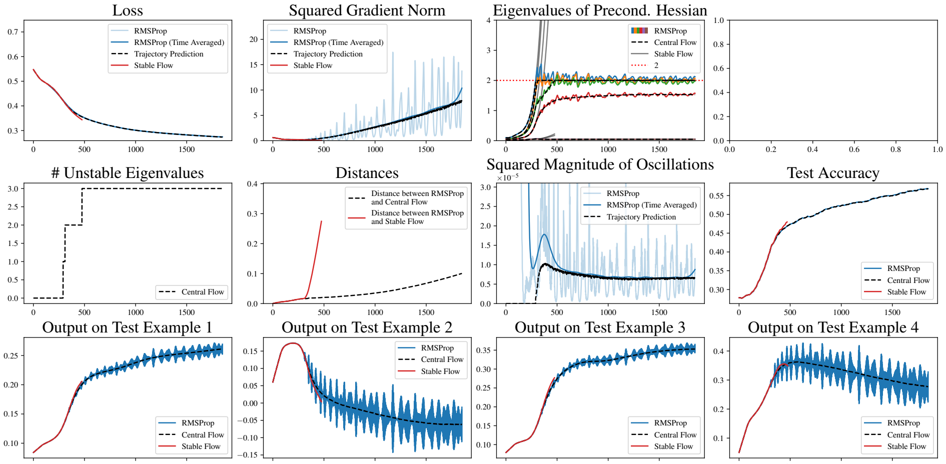


(b) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

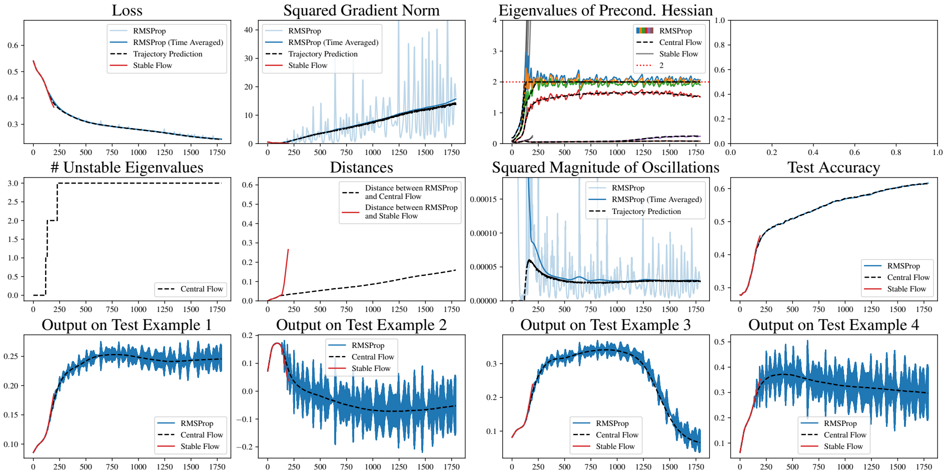


(c) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

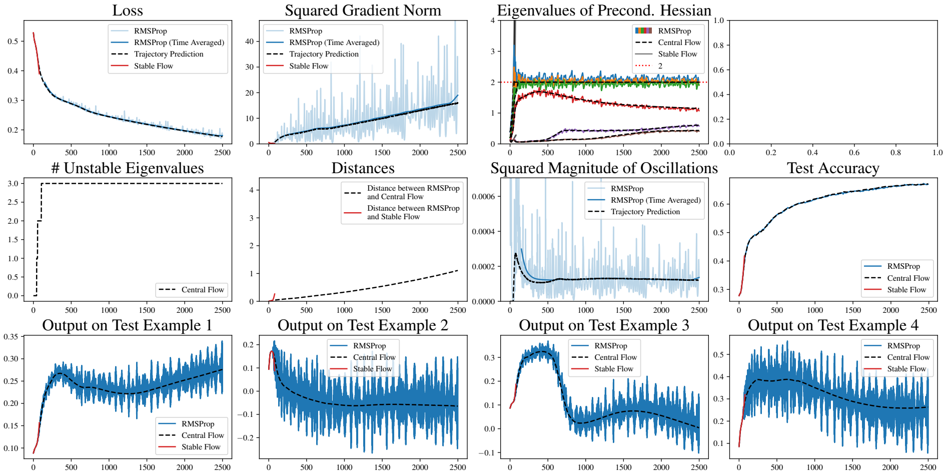
Figure 29: RMSProp on a CNN on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss



(a) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

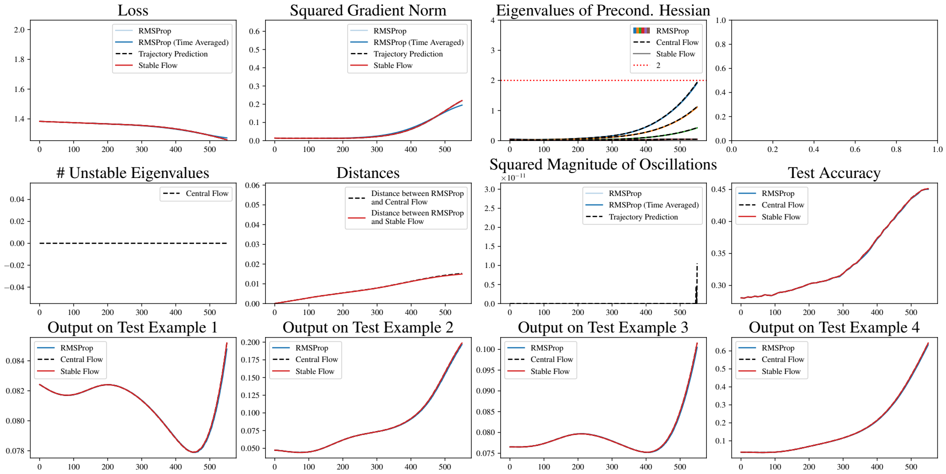


(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

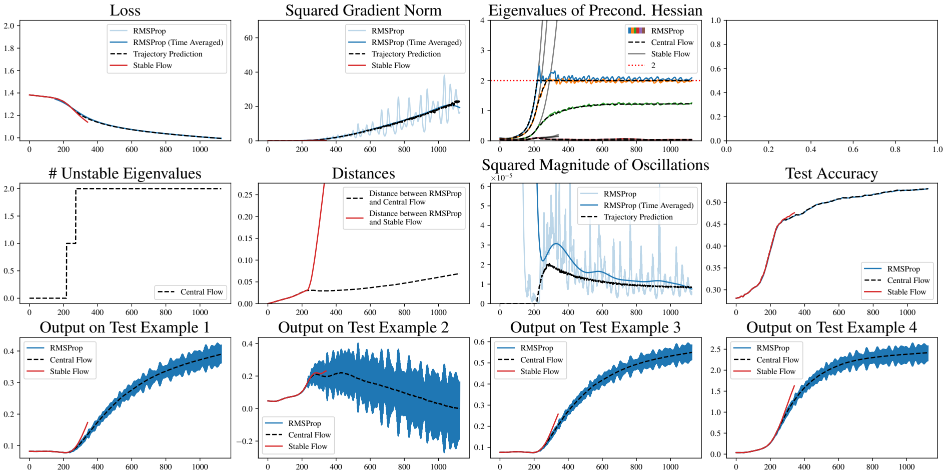


(c) $\eta = 4 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

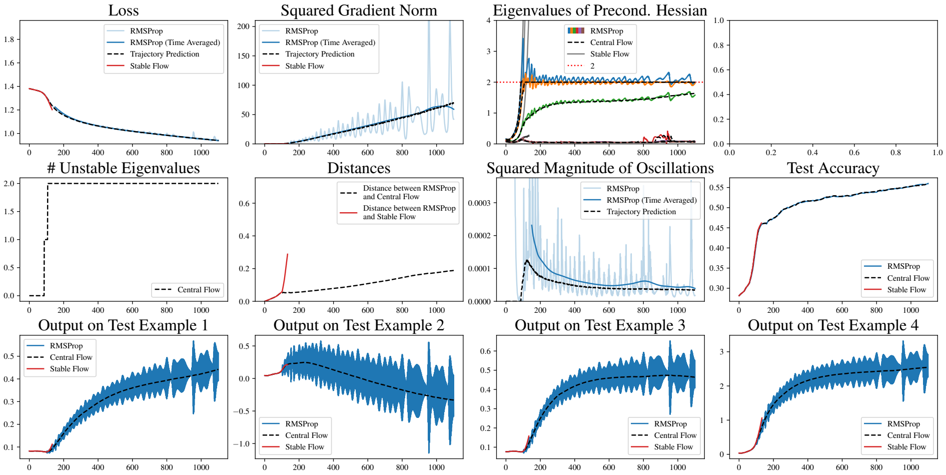
Figure 30: RMSProp on a ResNet on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 5 \times 10^{-6}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

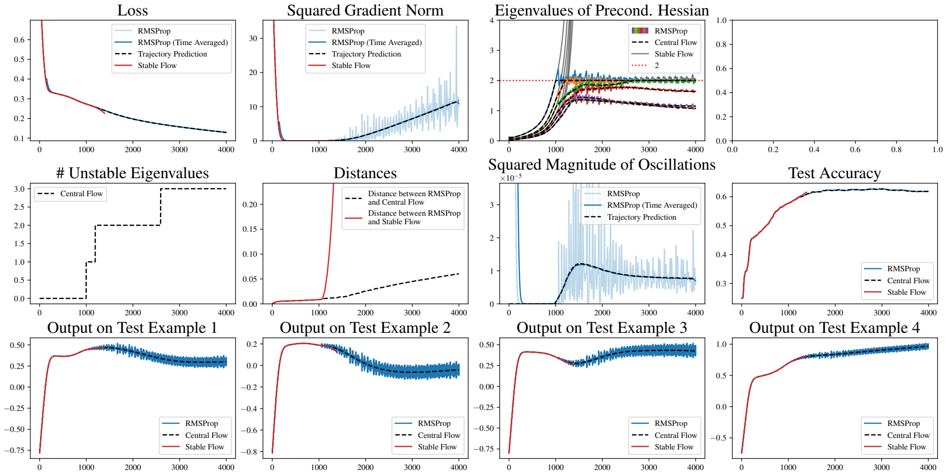


(b) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

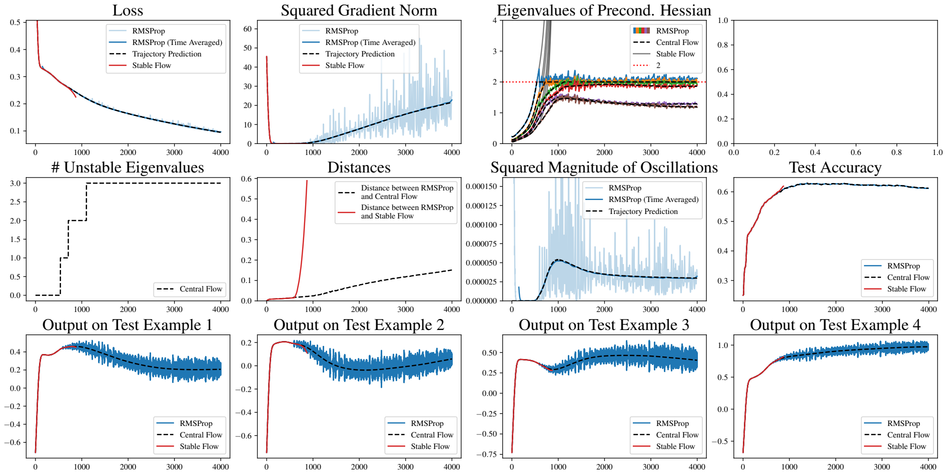


(c) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

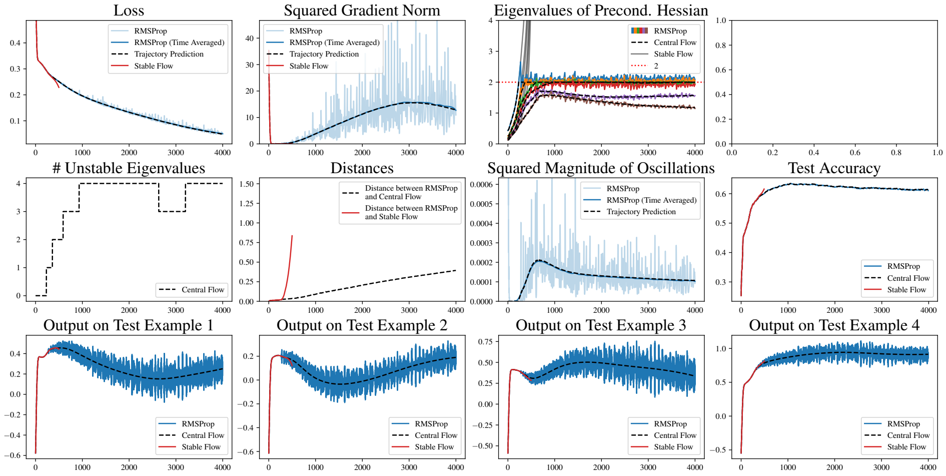
Figure 31: RMSProp on a ResNet on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss



(a) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

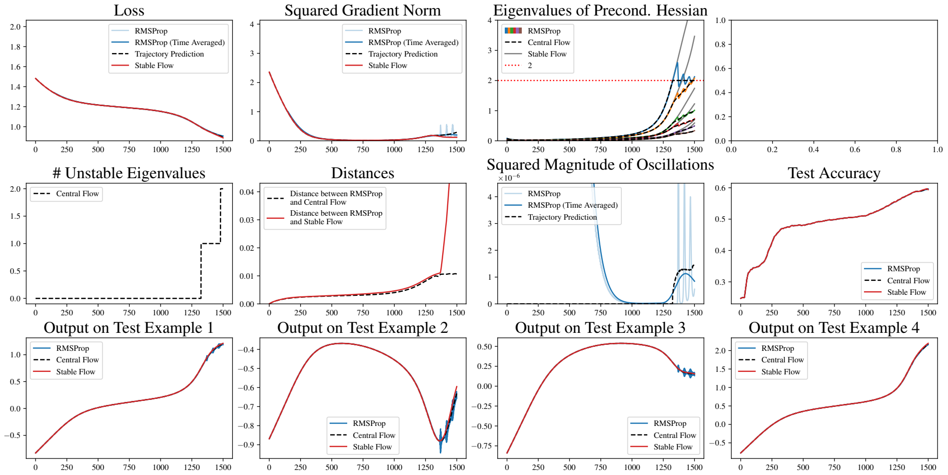


(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

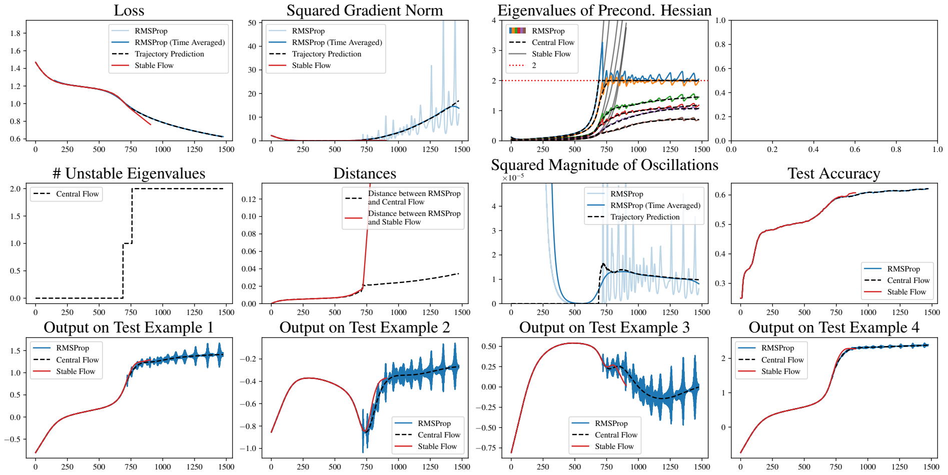


(c) $\eta = 4 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

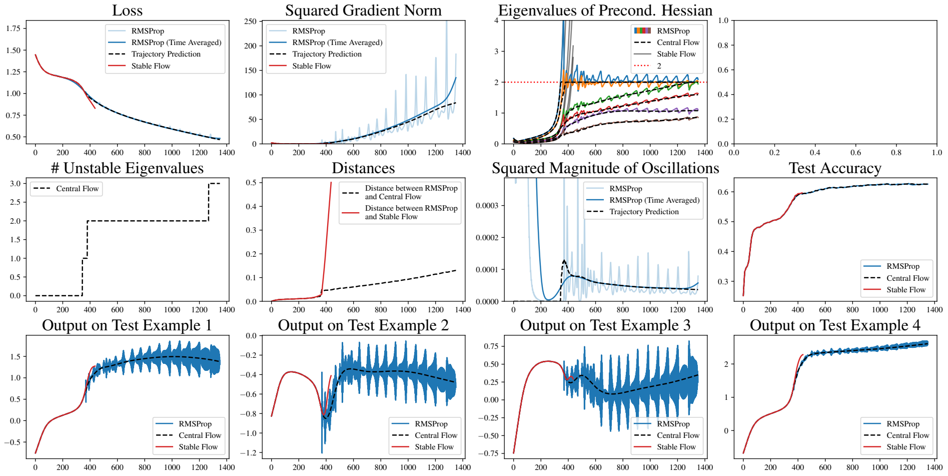
Figure 32: RMSProp on a ViT on a 1000 example, 4 class subset of CIFAR10 with MSE loss



(a) $\eta = 5 \times 10^{-6}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction



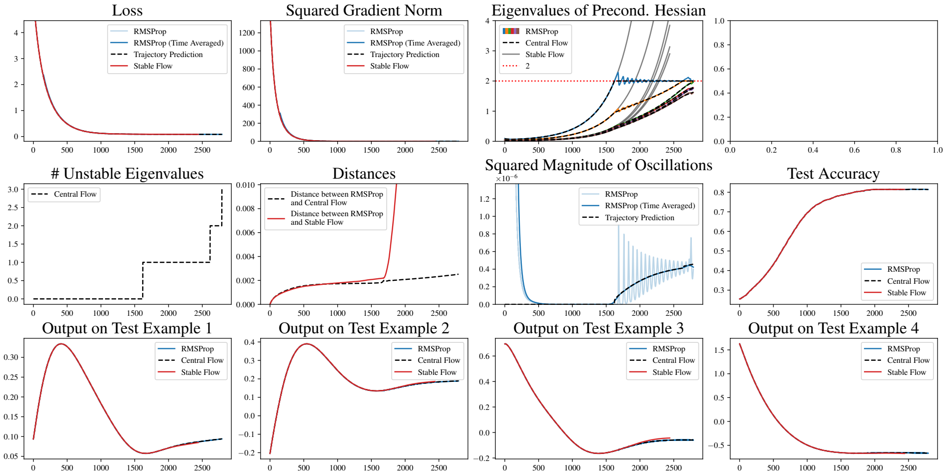
(b) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction



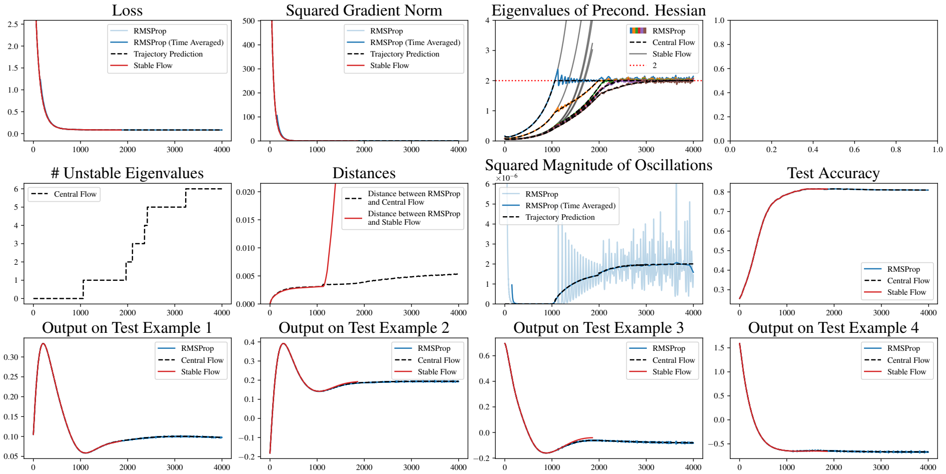
(c) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

Figure 33: RMSProp on a ViT on a 1000 example, 4 class subset of CIFAR10 with cross entropy loss

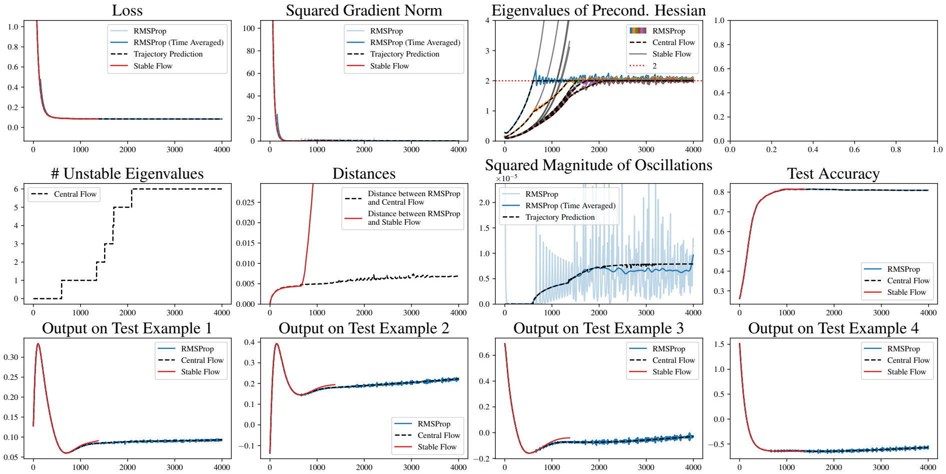
D.3.2 SORTING



(a) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

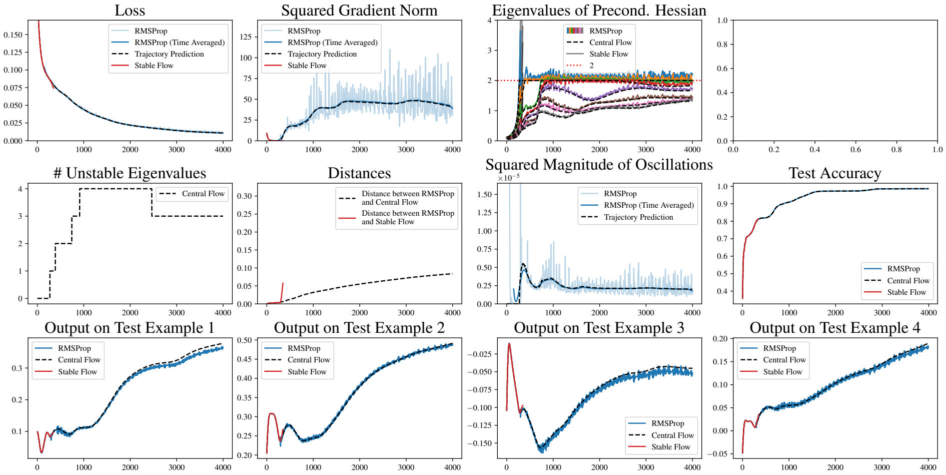


(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

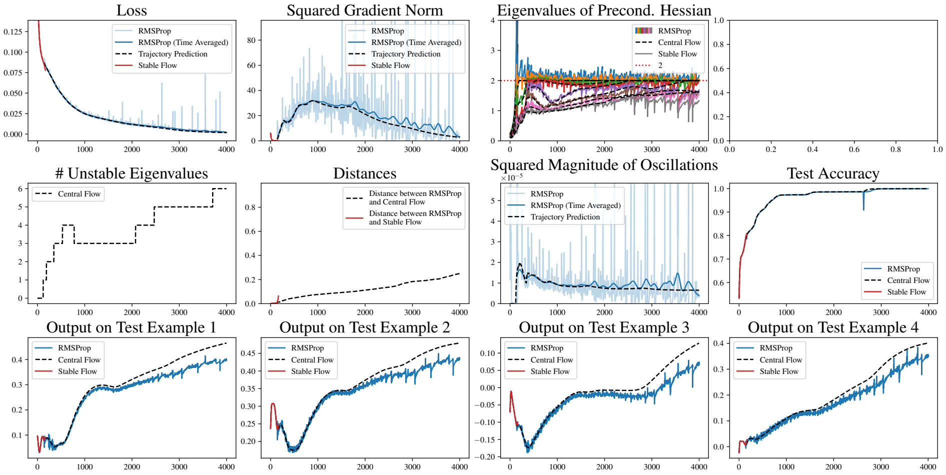


(c) $\eta = 4 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

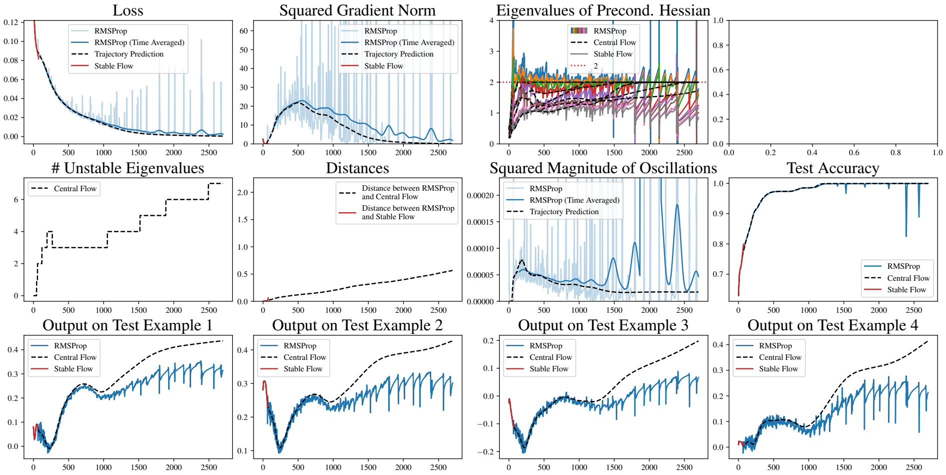
Figure 34: RMSProp on an RNN on a synthetic sorting task with MSE loss



(a) $\eta = 1 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction



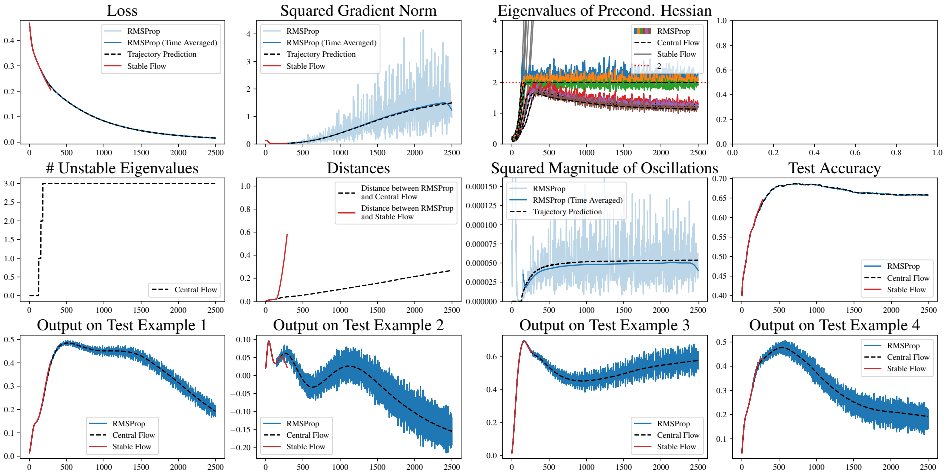
(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction



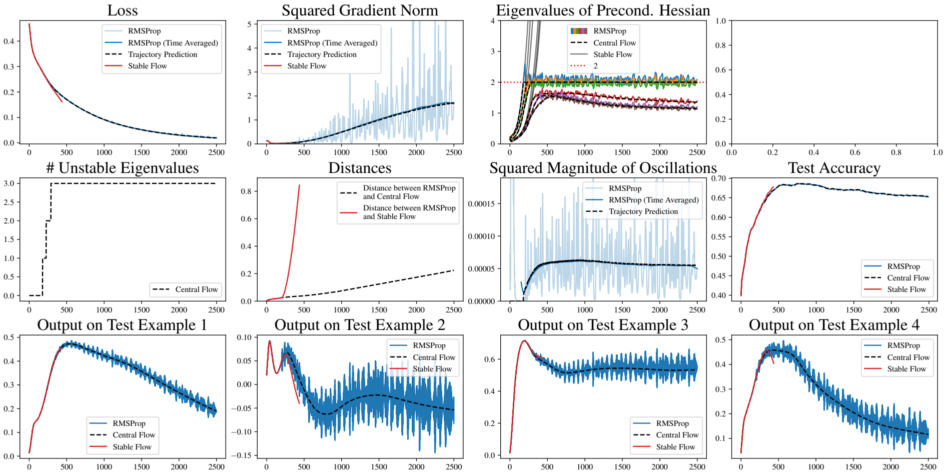
(c) $\eta = 4 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

Figure 35: RMSProp on a GPT-style transformer on a synthetic sorting task with MSE loss

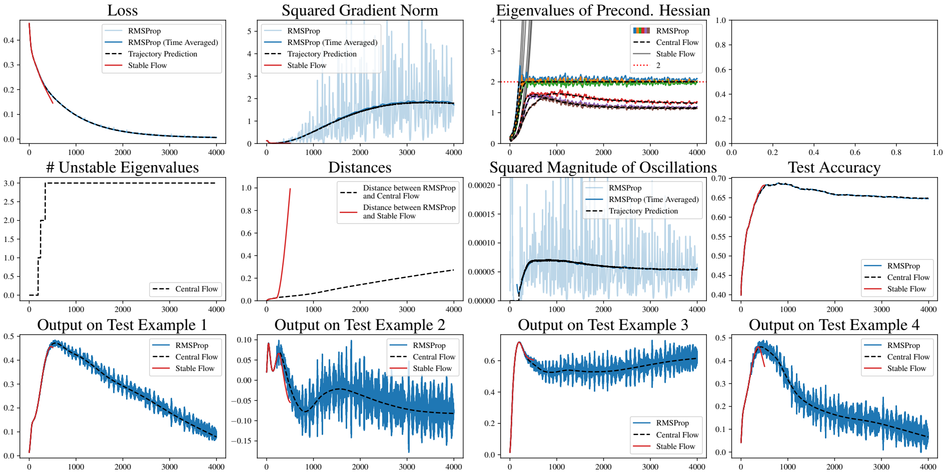
D.3.3 THE EFFECT OF β_2



(a) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.95$, $\epsilon = 10^{-7}$, bias correction



(b) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.99$, $\epsilon = 10^{-7}$, bias correction



(c) $\eta = 2 \times 10^{-5}$, $\beta_2 = 0.995$, $\epsilon = 10^{-7}$, bias correction

Figure 36: RMSProp on a CNN on a 1000 example, 4 class subset of CIFAR10 with MSE loss and various β_2

E EXTRA FIGURES

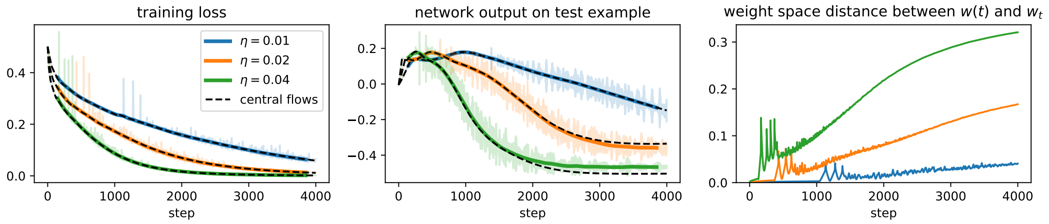
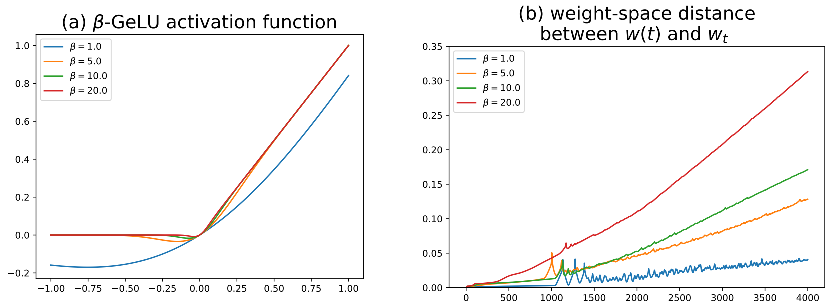


Figure 37: **Central flow approximation is less accurate at larger learning rates**. We run both gradient descent and its central flow at three learning rates (colors). The larger the learning rate, the faster the growth in the accumulated approximation error (right). [1]Indeed, at larger learning rates, 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).



(a) β-GeLU activation function

(b) weight-space distance between $w(t)$ and w_t

(c) loss and network output on test example

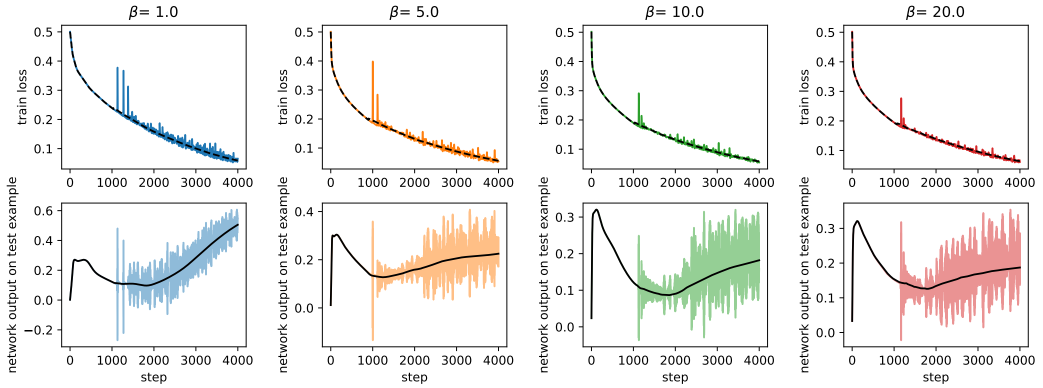


Figure 38: **Accuracy of central flow degrades as activation function becomes less smooth.** We consider networks with the β-GeLU activation function from Dauphin et al. (2024), defined as $x \mapsto x\Phi(\beta x)$ where Φ is the standard Gaussian CDF. This activation interpolates between GeLU when $\beta = 1$ and 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.

---

[1] While the rate of error accumulation in this figure can be partially explained by the fact that larger learning rates travel farther through weight space over a fixed number of steps t, this effect persists even after rescaling time and plotting $\eta \times t$, rather than t, on the x axis.