On a continuous time model of gradient descent dynamics and instability in deep learning

Anonymous authors
Paper under double-blind review

Abstract

The recipe behind the success of deep learning has been the combination of neural networks and gradient-based optimization. Understanding the behavior of gradient descent however, and particularly its instability, has lagged behind its empirical success. To add to the theoretical tools available to study gradient descent we propose *the principal flow* (PF), a continuous time flow that approximates gradient descent dynamics. To our knowledge, the PF is the only continuous flow that captures the divergent and oscillatory behaviors of gradient descent, including escaping local minima and saddle points. Through its dependence on the eigendecomposition of the Hessian the PF sheds light on the recently observed edge of stability phenomena in deep learning. Using our new understanding of instability we propose a learning rate adaptation method which enables us to control the trade-off between training stability and test set evaluation performance.

1 Introduction

Our goal is to use continuous time models to understand the behavior of gradient descent. Using continuous dynamics to understand discrete time systems opens up tools from dynamical systems such as stability analysis, and has a long history in optimization and machine learning (Glendinning, 1994; Saxe et al., 2013; Nagarajan and Kolter, 2017; Lampinen and Ganguli, 2018; Arora et al., 2018; Advani et al., 2020; Elkabetz and Cohen, 2021; Vardi and Shamir, 2021; Franca et al., 2020; Barrett and Dherin, 2021; Smith et al., 2021). Most theoretical analysis of gradient descent using continuous time systems uses the negative gradient flow, but this has well known limitations such as not being able to explain any behavior contingent on the learning rate. To mitigate these limitations we find a new continuous time flow which reveals important new roles of the Hessian in gradient descent training. To do so, we use backward error analysis (BEA), a method with a long history in the numerical integration community (Hairer et al., 2006) that has only recently been used in the deep learning context (Barrett and Dherin, 2021; Smith et al., 2021).

We find that the proposed flow sheds new light on gradient descent stability, including but not limited to divergent and oscillatory behavior around a fixed point. Instability — areas of training where the loss consistently increases — and edge of stability behaviors (Cohen et al., 2021) —areas of training where the loss does not behave monotonically but decreases over long time periods — are pervasive in deep learning and occur for all learning rates and architectures Cohen et al. (2021); Gur-Ari et al. (2018); Gilmer et al. (2021); Lewkowycz et al. (2020). We use our novel insights to understand and mitigate these instabilities.

We make the following theoretical and empirical contributions:

- We introduce **the principal flow** (the PF), a flow defined by the eigendecomposition of the Hessian (Section 3). To our knowledge the PF is the first continuous time flow that captures that gradient descent can diverge around local minima and saddle points, behaviors unexplained by existing flows (Section 2).
- We show how the PF can be used to model neural network training behavior (Section 4) and use the PF to shed new light on edge of stability behaviors in deep learning (Section 5).
- We introduce DAL (Drift Adjusted Learning rate), an approach of setting the learning rate based on our new understanding of gradient descent and show it can reduce instabilities (Section 6). We

explore the trade-off between stability and performance in deep learning, and show how DAL can be adapted to balance this trade-off (Section 6.3). We end by briefly showing how DAL can be used in conjunction with other optimization schemes (Section 7).

Notation: We denote as E the loss function, θ the parameter vector of dimension D, $\nabla_\theta^2 E$ the loss Hessian and λ_i and u_i the i'th largest eigenvalue and eigenvector respectively of $\nabla_\theta^2 E$. As a convention, since if u_i is an eigenvector of $\nabla_\theta^2 E$ so is $-u_i$, we always use u_i such that $Re[\nabla_\theta E^T u_i] \geq 0$. In the context of a continuous time flow $\theta(h)$ refers to the solution of the flow at time h.

Experiments: A list of figures with details on how to reproduce each of them is provided in the Appendix.

2 Continuous time models of gradient descent

The aim of this work is to understand the dynamics of gradient descent updates with learning rate h

$$\theta_t = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1}) \tag{1}$$

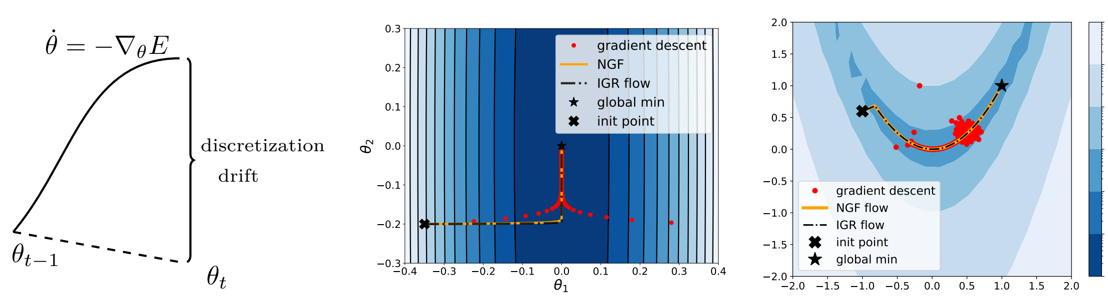
from the perspective of continuous dynamics. When using continuous time dynamics to understand gradient descent it is most common to use *the negative gradient flow* (NGF)

$$\dot{\theta} = -\nabla_\theta E \tag{2}$$

Gradient descent can be obtained from the NGF through Euler numerical integration, with an error of $\mathcal{O}(h^2)$ after one gradient descent step. The well-known discrepancy between Euler integration and the NGF, often called *discretization error* or *discretization drift* (Figure 1(a)) leads to certain limitations when using the NGF to describe gradient descent, namely: the NGF cannot explain divergence around a local minima for high learning rates or convergence to flat minima as often seen in the training of neural networks. Critically, since the NGF does not depend on the learning rate, it cannot explain any learning rate dependent behavior.

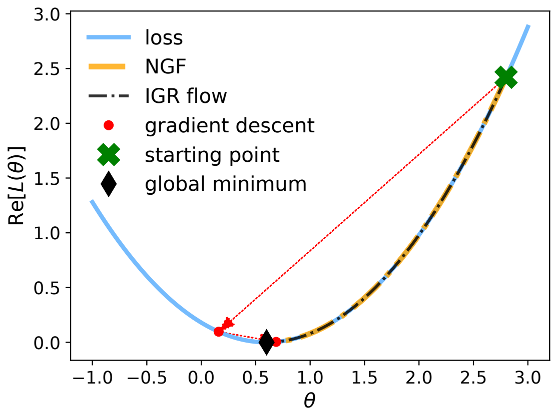
The appeal of continuous time methods together with the limitations of the NGF have inspired the machine learning community to look for other continuous time systems which may better approximate the gradient descent trajectory. One approach to constructing continuous time flows approximating gradient descent that takes into account the learning rate is backward error analysis (BEA). Using this approach, Barrett and Dherin (2021) introduce the Implicit Gradient Regularization flow (IGR flow):

$$\dot{\theta} = -\nabla_\theta E - \frac{h}{2}\nabla_\theta^2 E \nabla_\theta E \tag{3}$$

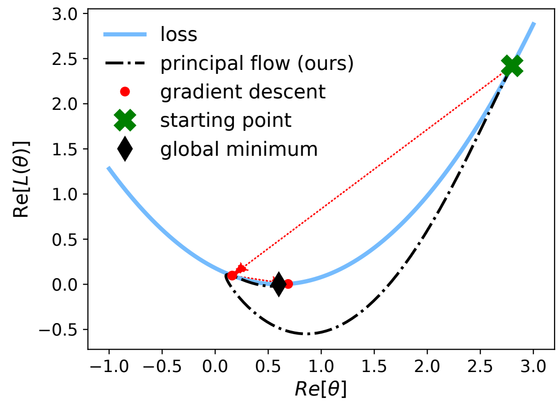


(a) Discretisation drift. (b) 2D convex case. (c) Banana function.

Figure 1: **Motivation**. Using continuous time flows to understand gradient descent is limited by the gap between the discrete and continuous dynamics. In the case of the negative gradient flow, we call this gap *discretization drift*. Other flows have been introduced to capture part of the drift, but they also fail to capture the oscillatory or unstable behavior of gradient descent.



(a) Real flows.



(b) Complex flow.

Figure 2: **Complex flows capture oscillations and divergence around local minima**. In the real space, the trajectory going from the starting point to the second gradient descent iterate goes through the global minima, and real flows stop there. In complex space however, that need not be the case.

which tracks the dynamics of the gradient descent step $\theta_t = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1})$ with an error of $\mathcal{O}(h^3)$, thus reducing the order of the error compared to the NGF. Unlike the NGF flow, the IGR flow depends on the learning rate h. This dependence explains certain properties of gradient descent, such as avoiding paths with high gradient norm towards a local minima; the authors connect this behavior to convergence to flat minima.

Like the NGF flow however, the IGR flow does not explain the instabilities of gradient descent, as we illustrate in Figure 1. Indeed, Barrett and Dherin (2021) (their Remark 3.4) show that performing stability analysis around local minima using the IGR flow does not lead to qualitatively different conclusions from those using the NGF: both NGF and the IGR flow predict gradient descent to be always locally attractive around a local minima, contradicting the empirically observed behavior of gradient descent (proofs in Section A.4.1). To understand why both the NFG and the IGR flow cannot capture oscillations and divergence around a local minima, we note that stationary points $\nabla_\theta E = 0$ are fixed points for both flows. We visualize an example in Figure 2(a): since to go from the initial point to the gradient descent iterates requires passing through the local minimum, both flows would stop at the local minimum. To overcome this issue, we show that a flow in complex space does not have to go through the real valued local minimum and thus will not stop there, as shown in Figure 2(b). We will later show the importance of operating in complex space in order to understand oscillatory and instability behaviors of gradient descent.

In the case of neural networks we show in Figure 30 in the Appendix that while the IGR flow is better than the NGF at describing gradient descent, a substantial gap remains. Inspired by the progress made by the IGR flow to understand gradient descent and motivated by the remaining limitations of existing flows we expand the application of BEA in deep learning towards understanding empirically observed instabilities.

2.1 Backward error analysis

Backward error analysis (BEA) is a tool in numerical analysis developed to understand the discretization error of numerical integrators. We now present an overview of how to use it in the context of gradient descent; for a general overview see Hairer et al. (2006). BEA provides a modified vector field:

$$\tilde{f}_n(\theta) = -\nabla_\theta E + hf_1(\theta) + \cdots + h^n f_n(\theta), \tag{4}$$

by finding functions f_1, ... f_n such that the solution of the modified ODE at order n, that is,

$$\dot{\tilde{\theta}} = -\nabla_\theta E + hf_1(\theta) + \cdots + h^n f_n(\theta) \tag{5}$$

follows the discrete dynamics of the gradient descent update with an error $\|\theta_t - \tilde{\theta}(h)\|$ of order $\mathcal{O}(h^{n+2})$, where $\tilde{\theta}(h)$ is the solution of the modified equation truncated at order n at time h, with $\tilde{\theta}(0) = \theta_{t-1}$. The full modified vector field with all orders ($n \to \infty$)

$$\tilde{f}(\theta) = -\nabla_\theta E + h f_1(\theta) + \cdots + h^n f_n(\theta) + \cdots, \tag{6}$$

is usually divergent and only forms an asymptotic expansion. What BEA provides is the Taylor expansion in h of an unknown h-dependent vector field $f_h(\theta)$ developed at $h = 0$:

$$\tilde{f}(\theta) = \text{Taylor}_{h=0} f_h(\theta). \tag{7}$$

Thus a strategy for finding f_h is to find a series of the form in Eq 6 via BEA and then find the function f_h such that its Taylor expansion in h at 0 results in the found series. Using this approach we can find the flow $\dot{\tilde{\theta}} = f_h(\tilde{\theta})$ which exactly describes the gradient descent step $\theta_t = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1})$.

While flows obtained using BEA are constructed to approximate one gradient descent step, the same flows can be used over multiple gradient descent steps as shown in Section A.8 in the Appendix.

BEA proofs. The general structure of BEA proofs is as follows: start with a Taylor expansion in h of the modified flow in Equation 5; write each term in the Taylor expansion as a function of $\nabla_\theta E$ and the desired f_i (this often requires applying the chain rule repeatedly); group together terms of the same order in h in the expansion; and identify f_i such that all terms of $\mathcal{O}(h^p)$ are 0 for $p \geq 2$, as is the case in the gradient descent update. A formal overview of BEA proofs can be found in Section A.1 in the Appendix.

We now exemplify how to use BEA to find the IGR flow (Eq 3) (Barrett and Dherin, 2021). Since we are only looking for the first correction term, we only need to find f_1. We perform a Taylor expansion to find the value of $\tilde{\theta}(h)$ up to order $\mathcal{O}(h^3)$ and then identify f_1 from that expression such that the error $\|\theta_t - \tilde{\theta}(h)\|$ is of order $\mathcal{O}(h^3)$. We have: $\tilde{\theta}(h) = \theta_{t-1} + h\tilde{\theta}^{(1)}(\theta_{t-1}) + \frac{h^2}{2}\tilde{\theta}^{(2)}(\theta_{t-1}) + \mathcal{O}(h^3)$. We know by the definition of the modified vector field (Eq 5) that $\tilde{\theta}^{(1)} = -\nabla_\theta E + hf_1(\tilde{\theta})$. We can then use the chain rule to obtain $\tilde{\theta}^{(2)} = \frac{-\nabla_\theta E + hf_1(\theta)}{dt} = \frac{-\nabla_\theta E}{dt} + \mathcal{O}(h) = \frac{-\nabla_\theta E}{d\theta}\frac{d\theta}{dt} + \mathcal{O}(h) = \nabla_\theta^2 E \nabla_\theta E + \mathcal{O}(h)$. Thus $\tilde{\theta}(h) = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1}) + h^2 f_1(\theta_{t-1}) + \frac{h^2}{2}\nabla_\theta^2 E(\theta_{t-1})\nabla_\theta E(\theta_{t-1}) + \mathcal{O}(h^3)$. We can then write $\theta_t - \tilde{\theta}(h) = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1}) - \Big(\theta_{t-1} - h\nabla_\theta E(\theta_{t-1}) + hf_1(\theta_{t-1}) + \frac{h^2}{2}\nabla_\theta^2 E(\theta_{t-1})\nabla_\theta E(\theta_{t-1}) + \mathcal{O}(h^3)\Big)$. After simplifying we obtain $\theta_t - \tilde{\theta}(h) = h^2 f_1(\theta_{t-1}) + \frac{h^2}{2}\nabla_\theta^2 E(\theta_{t-1})\nabla_\theta E(\theta_{t-1}) + \mathcal{O}(h^3)$. For the error to be of order $\mathcal{O}(h^3)$ the terms of order $\mathcal{O}(h^2)$ have to be 0. This entails $f_1 = -\frac{1}{2}\nabla_\theta^2 E \nabla_\theta E$ leading to Eq 3.

3 The principal flow

In the previous section we have seen how BEA can be used to define continuous time flows which capture the dynamics of gradient descent up to a certain order in learning rate. We have also explored the limitations of these flows, including the lack of ability to explain oscillations observed empirically when using gradient descent. To further expand our understanding of gradient descent via continuous time methods, we would like to get an intuition for the structure of higher order modified vector fields provided by BEA. We start with the following modified vector field, which we will call *the third order flow* (proof in Section A.2):

$$\dot{\theta} = -\nabla_\theta E - \frac{h}{2}\nabla_\theta^2 E \nabla_\theta E - h^2\left(\frac{1}{3}(\nabla_\theta^2 E)^2 \nabla_\theta E + \frac{1}{12}\nabla_\theta E^T(\nabla_\theta^3 E)\nabla_\theta E\right) \tag{8}$$

The third order flow tracks the dynamics of the gradient descent step $\theta_t = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1})$ with an error of $\mathcal{O}(h^4)$, thus further reducing the order of the error compared to the IGR flow. Like the IGR flow and the NGF, the third order flow has the property that $\dot{\theta} = 0$ if $\nabla_\theta E = 0$ and thus will exhibit the same limitations observed in Figure 2. We can now spot a pattern: the correction term of order $\mathcal{O}(h^n)$ in the BEA modified flow describing gradient descent contains the term $(\nabla_\theta^2 E)^n \nabla_\theta E$ and terms which contain higher order derivatives with respect to parameters, terms which we will denote as $\mathcal{C}(\nabla_\theta^3 E)$.

We will use the terms of the form $(\nabla_\theta^2 E)^n \nabla_\theta E$ to construct a new continuous time flow. We will take a three-step approach. First, for an arbitrary order $\mathcal{O}(h^n)$ we will find the terms containing only first and second order derivatives in the modified vector field given by BEA and show they are of the form $(\nabla_\theta^2 E)^n \nabla_\theta E$ (Theorem 3.1). Second, we will use all orders to create a series (Corollary 3.1). Third, we will use the series to find the modified flow given by BEA (Theorem 3.2). All proofs are provided in Section A of the Appendix.

Theorem 3.1 *The modified vector field with an error of order $\mathcal{O}(h^{n+2})$ to the gradient descent update $\theta_t = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1})$ has the form:*

$$\dot{\theta} = \sum_{p=0}^{n} \frac{-1}{p+1} h^p (\nabla_\theta^2 E)^p \nabla_\theta E + \mathcal{C}(\nabla_\theta^3 E) \tag{9}$$

where $\mathcal{C}(\nabla_\theta^3 E)$ denotes the family of functions which can be written as a sum of terms, each term containing a derivative of higher order than 3 with respect to parameters.

The result for $n = 1, 2$ and 3 follows from the NGF, IGR and third order flows. For higher order terms, the proof uses induction to find the term in f_i depending on $\nabla_\theta^2 E$ and $\nabla_\theta E$ only and follows the structure highlighted in Section 2.1, but Step 3 is modified to not account for terms in $\mathcal{C}(\nabla_\theta^3 E)$. From the above, we can obtain the following corollary by using all orders n and the eigen decomposition of $\nabla_\theta^2 E$:

Corollary 3.1 *The modified flow capturing gradient descent discrete updates exactly is of the form:*

$$\dot{\theta} = \sum_{p=0}^{\infty} \frac{-1}{p+1} h^p (\nabla_\theta^2 E)^p \nabla_\theta E + \mathcal{C}(\nabla_\theta^3 E) = \sum_{p=0}^{\infty} \frac{-1}{p+1} h^p \left(\sum_{i}^{D} \lambda_i^p u_i u_i^T \right) \nabla_\theta E + \mathcal{C}(\nabla_\theta^3 E) \tag{10}$$

$$= \sum_{i=1}^{D} \left(\sum_{p=0}^{\infty} \frac{-1}{p+1} h^p \lambda_i^p \right) (\nabla_\theta E^T u_i) u_i + \mathcal{C}(\nabla_\theta^3 E) \tag{11}$$

where λ_i and u_i are the respective eigenvalues and eigenvectors of the Hessian $\nabla_\theta^2 E$.

Definition 3.1 *We define the **principal flow** (PF) as*

$$\dot{\theta} = \sum_{i}^{D} \frac{\log(1 - h\lambda_i)}{h\lambda_i} (\nabla_\theta E^T u_i) u_i \tag{12}$$

Theorem 3.2 *The Taylor expansion in h at $h = 0$ of the PF vector field coincides with the series coming from the BEA of gradient descent (Eq 11).*

Proof: Using the Taylor expansion $\text{Taylor}_{z=0} \frac{\log(1-z)}{z} = \sum_{p=0}^{\infty} \frac{-1}{p+1} z^p$ we obtain:

$$\text{Taylor}_{h=0} \sum_{i=1}^{D} \frac{\log(1 - h\lambda_i)}{h\lambda_i} (\nabla_\theta E^T u_i) u_i = \sum_{i=1}^{D} \left(\sum_{p=0}^{\infty} \frac{-1}{p+1} h^p \lambda_i^p \right) (\nabla_\theta E^T u_i) u_i \tag{13}$$

□

We have thus used BEA to find the flow that when Taylor expanded at $h = 0$ leads to the series in Eq 11. Although the full modified vector field also contains terms which are not accounted for in the PF we will show both theoretically and empirically that the PF captures key features of the gradient descent dynamics in stable or unstable regions of training, around and outside critical points, for small examples or large neural networks. We start with a few observations:

Remark 3.1 *For quadratic losses of the form $E = \frac{1}{2}\theta^T A \theta + b^T \theta$, the PF captures gradient descent exactly. This case has been proven in Hairer et al. (2006). The solution of the PF can also be computed exactly in terms of the eigenvalues of $\nabla_\theta^2 E$: $\theta(t) = \sum_{i=1}^{D} e^{\frac{\log(1-h\lambda_i)}{h} t} \theta_0^T u_i u_i + t \sum_{i=1}^{D} \frac{\log(1-h\lambda_i)}{h\lambda_i} b^T u_i$.*

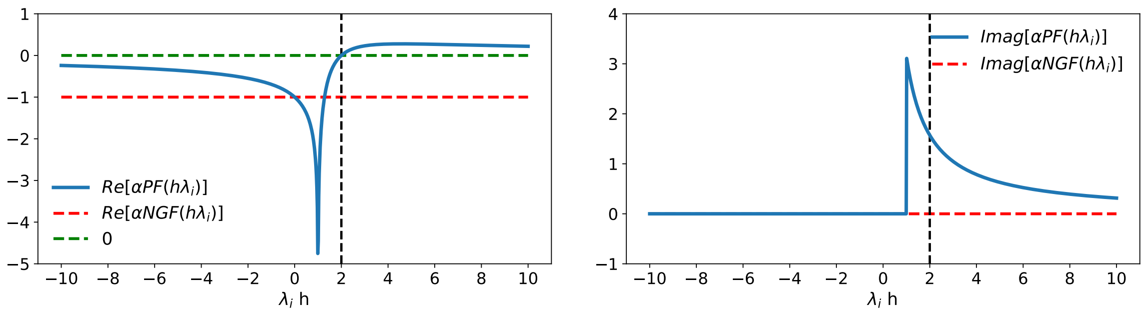
Remark 3.2 *We note that $\lim_{\lambda \to 0} \frac{\log(1-h\lambda)}{h\lambda} = -1$ and thus the PF is well defined when the Hessian $\nabla_\theta^2 E$ is not invertible.*

Definition 3.2 *The terms $\mathcal{C}(\nabla_\theta^3 E)$ are called **non-principal terms**. The term $\frac{1}{12} \nabla_\theta E^T (\nabla_\theta^3 E) \nabla_\theta E$ in Eq 8 is a non-principal term (we will call this term non-principal third order term).*

Remark 3.3 *In a small enough neighborhood around a critical point (where higher order derivatives can be ignored) the non-principal terms have a small magnitude and the PF can be used to describe gradient descent dynamics closely. We show this also using a linearization argument in Section A.5 in the Appendix.*

Negative Gradient Flow	IGR Flow	Principal Flow
$\dot{\theta} = \sum_i^D -(\nabla_\theta E^T u_i) u_i$	$\dot{\theta} = \sum_i^D -(1 + \frac{h}{2}\lambda_i)(\nabla_\theta E^T u_i) u_i$	$\dot{\theta} = \sum_i^D \frac{\log(1-h\lambda_i)}{h\lambda_i}(\nabla_\theta E^T u_i) u_i$
$\alpha_{NGF}(h\lambda_i) = -1$	$\alpha_{IGR}(h\lambda_i) = -(1 + \frac{h}{2}\lambda_i)$	$\alpha_{PF}(h\lambda_i) = \frac{\log(1-h\lambda_i)}{h\lambda_i}$

Table 1: Understanding the differences between the flows discussed in terms of the eigendecomposition of the Hessian. All flows have the form $\dot{\theta} = \sum_{i=1}^D \alpha(h\lambda_i)(\nabla_\theta E^T u_i) u_i$ with different α summarized here.



(a) Real part.

(b) Imaginary part.

Figure 3: Comparing the coefficients α_{NGF} and α_{PF} across the training landscape.

Remark 3.4 *If $\lambda_0 > 1/h$, the PF is complex, since in that case $\log(1 - h\lambda_0) = \log(h\lambda_0 - 1) + i\pi$.*

Definition 3.3 *We define the **principal flow with third order non principal term** as*

$$\dot{\theta} = \sum_i^D \frac{\log(1 - h\lambda_i)}{h\lambda_i}(\nabla_\theta E^T u_i) u_i - \underbrace{\frac{h^2}{12}\nabla_\theta E^T (\nabla_\theta^3 E)\nabla_\theta E}_{\textit{third order non principal term}} \tag{14}$$

Remark 3.5 *Unlike the NGF and the IGR flow, the modified vector field of the PF cannot be always written as the gradient of a loss function in $\mathbb{R}$.*

3.1 The principal flow and the eigen decomposition of the Hessian

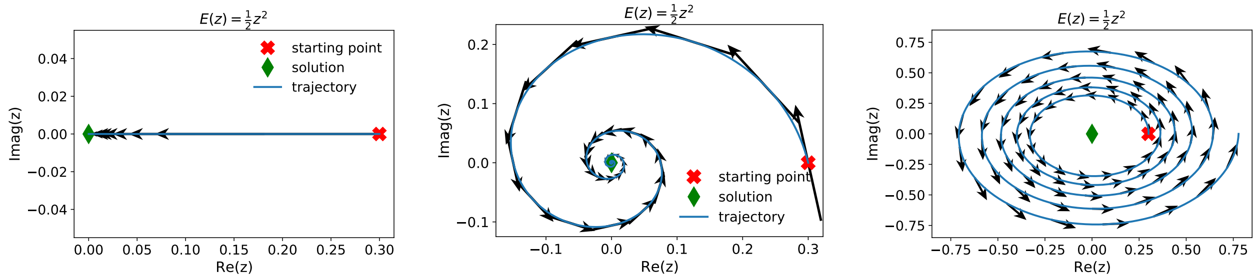
All flows considered here have the form form $\dot{\theta} = \sum_{i=1}^D \alpha(h\lambda_i)(\nabla_\theta E^T u_i) u_i$, where α is a function computing the corresponding coefficient; we will denote the one associated with each flow as α_{NGF}, α_{PF} and α_{IGR} respectively. For a side-by-side comparison between the NGF, IGR flow and the PF as functions of the Hessian eigendecomposition see Table 1. Since $\nabla_\theta E^T u_i \geq 0$, the α function determines the sign of a modified vector field in the direction u_i. For brevity, it will be useful to define the coefficient of u_i in the vector field of the PF:

Definition 3.4 *We call $sc_i = \frac{\log(1-h\lambda_i)}{h\lambda_i}(\nabla_\theta E^T u_i) = \alpha_{PF}(h\lambda_i)\nabla_\theta E^T u_i$ the **stability coefficient** for eigendirection i. $sign(sc_i) = sign(\alpha_{PF}(h\lambda_i))$.*

In order to understand the PF and how it is different to the NGF, we explore the change in each eigendirection u_i and we perform case analysis on the relative value of the eigenvalues λ_i and the learning rate h. To do so, we will compare α_{NGF} and α_{PF} based on the value of $h\lambda_i$; comparing with the NGF is especially insightful, since that is the direction in which the function E is minimized. Since our goal is to understand the behavior of gradient descent, we perform the case by case analysis of what happens at the start of a gradient descent iteration and thus use real values for λ_i and u_i, even when the PF is complex. We visualize α_{NGF} and α_{PF} in Figure 3 and we use Figure 4 to show examples of each case using a simple function.

Real stable case: $\lambda_i < 1/h$. $\text{sign}(\alpha_{NGF}) = \text{sign}(\alpha_{PF}) = -1$.

$\alpha_{NGF} = -1$ and $\alpha_{PF}(h\lambda_i) = \frac{\log(1-h\lambda_i)}{h\lambda_i} < 0$. The coefficients of both the NGF and PF in eigendirection u_i are both negative and real. The case is exemplified in Figure 4(a)..



(a) $\lambda_0 < 1/h, \quad \alpha_{PF}(\lambda_0 h) < 0$ (b) $1/h < \lambda_0 < 2/h, \;\; Re(\alpha_{PF}(\lambda_0 h)) < 0$ (c) $2/h < \lambda_0, \;\; Re(\alpha_{PF}(\lambda_0 h)) > 0$

Figure 4: The behavior of PF on $E(z) = \frac{1}{2}z^2$ with solution $z(t) = e^{\log(1-h)/h}z(0)$. When $\lambda_0 < 1/h$, $z(t) = (1-h)^{t/h}z(0)$ which is in real space and converges to the equilibrium. When $\lambda_0 > 1/h$, $z(t) = (h-1)^{t/h}\left(\cos(\pi t/h) + i\sin(\pi t/h)\right)z(0)$. This exhibits oscillatory behavior, and when $\lambda_0 > 2/h$, diverges.

Complex stable case: $1/h < \lambda_i < 2/h$. $\text{sign}(\alpha_{NGF}) = \text{sign}(Re[\alpha_{PF}]) = -1$. $\alpha_{PF} \in \mathbb{C}$.

$\alpha_{NGF} = -1$ and $\alpha_{PF}(h\lambda_i) = \frac{\log(1-h\lambda_i)}{h\lambda_i} = \frac{\log(-1+h\lambda_i)+i\pi}{h\lambda_i} \in \mathbb{C}$ and $Re[\alpha_{PF}] = \frac{\log(-1+h\lambda_i)}{h\lambda_i} < 0$. The real part of the coefficient of both the NGF and PF in eigendirection u_i are both negative. However, the imaginary part of α_{PF} can still introduce instability and oscillations, as we show in Figure 4(b).

Unstable complex case: $2/h < \lambda_i$. $\text{sign}(\alpha_{NGF}) \neq \text{sign}(Re[\alpha_{PF}])$. $\alpha_{PF} \in \mathbb{C}$.

$\alpha_{NGF} = -1$ and $\alpha_{PF}(h\lambda_i) = \frac{\log(1-h\lambda_i)}{h\lambda_i} = \frac{\log(-1+h\lambda_i)+i\pi}{h\lambda_i} \in \mathbb{C}$ and $Re[\alpha_{PF}] = \frac{\log(-1+h\lambda_i)}{h\lambda_i} > 0$. The real part of the coefficient the of NGF in eigendirection u_i is negative, while the real part of the coefficient of the PF is positive. Since the direction minimising the function E is given by the NGF and is negative, the change in sign given by the PF can cause instabilities. The complex component can still introduce oscillations, however, the higher $\lambda_i h$, the smaller the complex component is. We show a simple example in Figure 4(c).

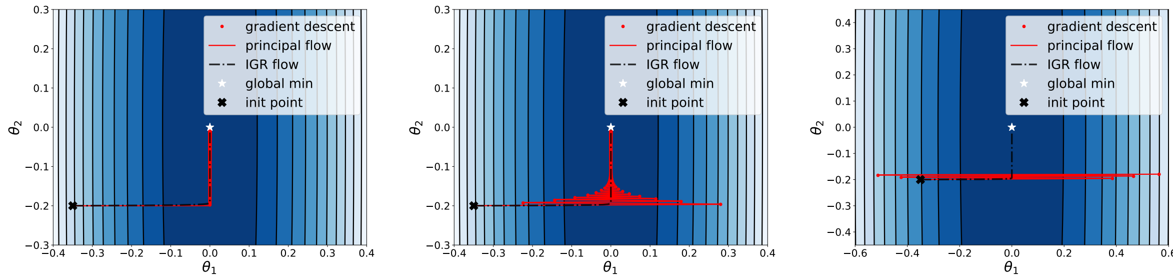
The importance of the largest eigenvalue λ_0. The largest eigenvalue λ_0 plays an important part in the PF. Since $h\lambda_0 \geq h\lambda_i \quad \forall i$, λ_0 determines where in the above cases the PF is situated and thus whether there are oscillations and unstable behavior in training. For all flows of the form we consider we can write:

$$\frac{dE(\theta)}{dt} = \frac{dE(\theta)}{d\theta}^T\frac{d\theta}{dt} = \nabla_\theta E^T \sum_{i=1}^{D} \alpha(h\lambda_i)\nabla_\theta E^T u_i u_i = \sum_{i=1}^{D} \alpha(h\lambda_i)(\nabla_\theta E^T u_i)^2 \tag{15}$$

and thus if $\alpha(h\lambda_i) \in \mathbb{R}$ and $\alpha(h\lambda_i) < 0 \; \forall i$ then $\frac{dE(\theta)}{dt} \leq 0$ and following the corresponding flow minimises E. In the case of the PF this gets determined by λ_0. If $\lambda_0 < \frac{1}{h}$ then $\alpha_{PF}(h\lambda_i) < 0 \; \forall i$ (real stable case above) and the PF minimises E. If $1/h < \lambda_0 < \frac{2}{h}$ then $Re[\alpha_{PF}(h\lambda_i)] < 0 \; \forall i$ (complex stable case above) close to a gradient descent iteration $\lambda_i, u_i \in \mathbb{R}$ we can write that $\frac{dRe[E(\theta)]}{dt} = \sum_{i=1}^{D} Re[\alpha_{PF}(h\lambda_i)](\nabla_\theta E^T u_i)^2$ and thus the real part of the loss function decreases. If $\lambda_0 > \frac{2}{h}$ then $Re[\alpha_{PF}(h\lambda_0)] > 0$ (unstable complex case above) and if $(\nabla_\theta E^T u_0)^2$ is sufficiently large we can no longer ascertain the behavior of E. We present a discrete time argument for this observation in Section A.7.1.

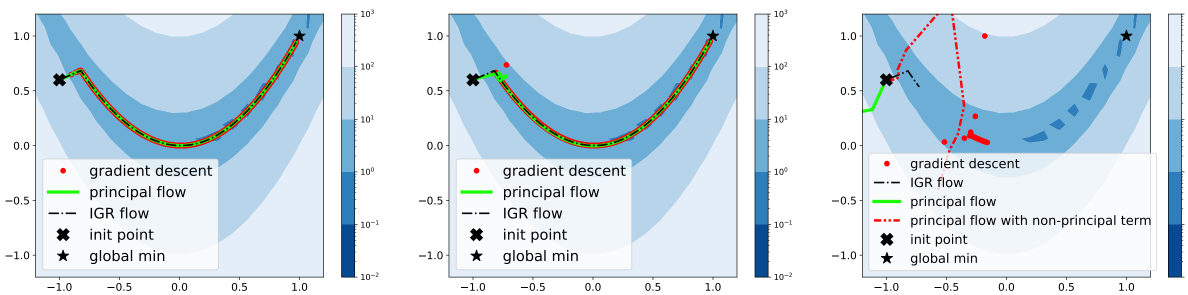
Building intuition. For quadratic objective $E(\theta) = \frac{1}{2}\theta^T A\theta$ the PF describes gradient descent exactly. We show examples Figures 2 and 5. Unlike the NGF or the IGR flow, the PF captures the oscillatory and divergent behavior of gradient decent. Importantly, to capture the unstable behavior which occurs when $\lambda_0 > 1/h$ the imaginary part of the PF is needed. To expand intuition outside the quadratic case, we show the PF for the banana function (Rosenbrock, 1960) in Figure 6 and an additional example in 1D with a non-quadratic function (Figure 29 in the Appendix). In this case, the PF no longer follows the gradient descent trajectory exactly, but we still we observe the importance of the PF in capturing instabilities of gradient descent; we also observe that adding non-principal terms can restabilize the trajectory.

Remark 3.6 *For the banana function, the principal terms have a destabilizing effect when $h > 2/\lambda_0$ while the non principal terms can have a stabilizing effect.*



(a) $\lambda_0 < 1/h$ (stability) (b) $1/h < \lambda_0 < 2/h$ (oscillations) (c) $\lambda_0 > 2/h$ (divergence)

Figure 5: **Quadratic losses in 2 dimensions**. The PF captures the behavior of gradient descent exactly for quadratic losses, including oscillatory behavior and divergence.



(a) $\lambda_0 < 1/h$ (b) $\lambda_0 < 2/h (\lambda_0 \approx 1.9/h)$ (c) $\lambda_0 >> 2/h (\lambda_0 \approx 5/h)$

Figure 6: **Banana function**. The PF can capture instability and the gradient descent trajectory over many iterations when λ_0 is close to $2/h$. When $\lambda_0 >> 2/h$ (right) the PF does not track the GD trajectory over many gradient descent steps, but when including a non-principal term the flow is able to capture the general trajectory of gradient descent and unstable behavior of gradient descent.

3.2 The stability analysis of the principal flow

We now perform stability analysis on the PF, to understand how it can be used to predict certain behaviors of gradient descent around critical points of the loss function E. Consider θ^* such a critical point, i.e $\nabla_\theta E(\theta^*) = 0$. For a critical point θ^* to be locally attractive, all eigenvalues of the Jacobian evaluated at θ^* need to have negative real part.

The principal flow has the following Jacobian (proof in Section A.4 in the Appendix):

$$J_{PF}(\theta^*) = \sum_{i=1}^{D} \frac{\log(1 - h\lambda_i^*)}{h} u_i^* {u_i^*}^T \tag{16}$$

where λ_i^*, u_i^* are the eigenvalues and eigenvectors of the Hessian $\nabla_\theta^2 E(\theta^*)$. We thus have that the eigenvalues of the Jacobian $J_{PF}(\theta^*)$ at the critical point θ^* are $\frac{1}{h}\log(1 - h\lambda_i^*)$ for $i = 1, \ldots, D$.

Local minima. Suppose that θ^* is a local minimum. Then all Hessian eigenvalues are positive $\lambda_i^* \geq 0$. We perform the stability analysis in cases given by the value of λ_i^*, corresponding to the cases in Section 3.1:

$h < 1/\lambda_i^*$. The corresponding eigenvalue of the Jacobian $\frac{1}{h}\log(1 - h\lambda_i^*)$ is negative, since $0 < 1 - h\lambda_i^* < 1$. The principal vector field is attractive in the corresponding eigenvector direction.

$h \in [1/\lambda_i^*, 2/\lambda_i^*]$. The corresponding eigenvalue of the Jacobian $\frac{1}{h}\log(1 - h\lambda_i^*) = \frac{1}{h}\log(h\lambda_i^* - 1) + i\frac{\pi}{h}$ is complex, with negative real part, since since $h\lambda_i^* - 1 < 1$. The principal vector field is attractive in the corresponding eigenvector direction.

$h > 2/\lambda_i^*$. The corresponding eigenvalue of the Jacobian $\frac{1}{h}\log(1-h\lambda_i^*) = \frac{1}{h}\log(h\lambda_i^*-1)+i\frac{\pi}{h}$ is complex, with positive real part, since since $h\lambda_i^* - 1 > 1$. The principal vector field is not attractive in the corresponding eigenvector direction.

The last case tells us that the PF is not always attracted to local minima, as it is not attractive in eigendrections where $h > 2/\lambda_i^*$. Thus **like gradient descent, the PF can be repelled around local minima for large learning rates**. This is in contrast to the NGF and the IGR flow, which always predict convergence around a local minima: the eigenvalues of the NGF Jacobian are $-\lambda_i^*$, and for the IGR flow the eigenvalues are $-\lambda_i^* - \frac{h^2}{2}{\lambda_i^*}^2$, both are negative when λ_i^* is positive. For derivations see Section A.4.1 in the Appendix.

Remark 3.7 *For quadratic losses, where the PF is exact, the results above recover the classical gradient descent result for quadratic losses namely that gradient descent convergences if $\lambda_0 < 2/h$, otherwise diverges.*

Saddle points. Suppose that θ^* is a saddle point. In this case there exists λ_s^* such that $\lambda_s^* < 0$. We want to analyse the behavior of the PF in the direction of the corresponding eigenvector u_s^*. In that case, $\log(1 - h\lambda_s^*) > 0$ which entails that the PF is repelled in the eigendirections of saddle points. Note that this is also the case for the NGF since the corresponding eigenvalues of the Jacobian of the NGF would be $-\lambda_s^*$, also positive. Unlike the NGF however, the subspace of eigendirections that the PF is repelled by can be larger since it includes also eigendirections where $\lambda_i^* > 2/h > 0$.

4 Predicting neural network gradient descent dynamics with the principal flow

Computing the PF on large neural networks during training is computationally prohibitive, as it requires finding all eigenvalues of the Hessian matrix once for each step of the flow simulation, corresponding to many eigen-decompositions per gradient descent step. To build intuition about the PF for neural networks, we start with a small MLP for a 2 dimensional input regression problem, with random inputs and labels. Here we can understand the behavior of the PF since we can compute its modified vector field exactly and compare it with the behavior of gradient descent. We show results in Figure 7, where we visualize the norm of the difference between gradient descent parameters at each iteration and the parameters produced by the continuous time flows we compare with. We observe that *short term the principal flow is better than all other flows at tracking the behavior of gradient descent.* As the number of iterations increases however, the PF accumulates error in the case of $\lambda_0 > 2/h$; this is likely due to the fact that while gradient descent parameters are real, this is not the case for the PF, as discussed in Remark 4.1. Since we are primarily concerned with using the PF to understand gradient descent for a small number of iterations this will be less of a concern in our experimental settings. Additional results which confirm the PF is better than the other flows at tracking gradient descent on a bigger network trained the UCI breast cancer dataset are shown in Figure 31 in the Appendix.

Remark 4.1 ***On the multiple iteration behavior of the principal flow.*** *We note that while gradient descent parameters are real for any iteration θ_t, θ_{t+1}, ... θ_{t+n}, when we approximate the behavior of gradient descent by initializing $\theta(0) = \theta_t$ and running the PF for time nh, there is nothing enforcing that $\theta(h)$, ... $\theta(nh)$ will be real when the PF is complex ($\lambda_0 > 1/h$). For long term trajectories (larger n), this can have an effect on long term error between gradient descent and PF trajectories, through an accumulating effect of the imaginary part in the PF. This can be mitigated by using the PF to understand the short term behavior of gradient descent (small n).*

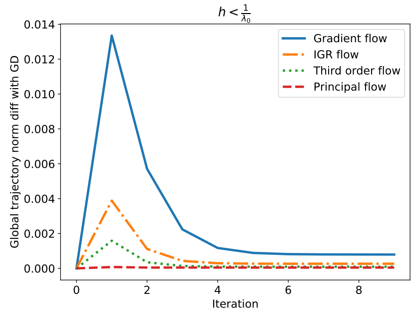
4.1 Predicting $\nabla_\theta E^T u_0$ using the principal flow

For large neural networks, instead of computing the entire PF describing how the entire parameter vector changes in time we can use the PF to approximate changes in time in one eigendirection only. This will allow us to compare the predictions of the PF against the predictions of the NGF and IGR flow on realistic settings. To do so, we first have to compute how the gradient changes in time:

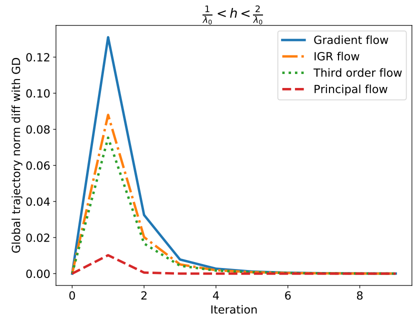
Corollary 4.1 *If θ follows the PF, then: $\dot{(\nabla_\theta E)} = \sum_i^D \frac{\log(1-h\lambda_i)}{h}(\nabla_\theta E^T u_i)u_i$.*

This follows from applying the chain rule and using the definition of the PF. We contrast this with how the gradient evolves if the parameters follow the NGF:

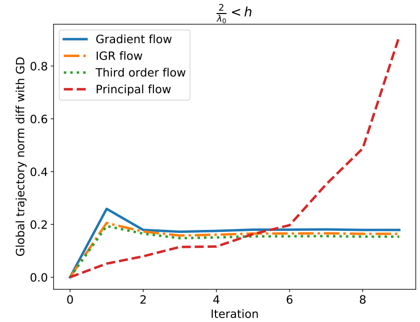
Corollary 4.2 *If θ follows the NGF, then: $\dot{(\nabla_\theta E)} = \sum_i^D -\lambda_i(\nabla_\theta E^T u_i)u_i$*



(a) $\lambda_0 < 1/h$



(b) $1/h < \lambda_0 < 2/h$



(c) $\lambda_0 > 2/h$

Figure 7: Error between gradient descent parameters and parameters obtained following continuous time flows for multiple iterations: $\|\theta_n - \theta(nh)\|$ with $\theta(0) = \theta_0$. For small n, the PF is better at capturing the behavior of gradient descent across all cases.

Corollary 4.3 *If θ follows the IGR flow, then:* $(\dot{\nabla_\theta E}) = \sum_i^D -\left(\lambda_i + \frac{h}{2}\lambda_i^2\right)(\nabla_\theta E^T u_i)u_i$

We would like to use the above to assess how $\nabla_\theta E^T u_i$ changes in time under the above flows and check their predictions empirically against results obtained when training neural networks with gradient descent. Since u_i is an eigenvector of the Hessian it also changes in time according to the changes given by the corresponding flow, making $(\nabla_\theta \dot{E}^T u_i)$ difficult to calculate. Even when if we wrote an exact flow for $(\nabla_\theta \dot{E}^T u_i)$, it would be computationally challenging to simulate it since finding the new values of u_i would depend on the full Hessian and would lead to the same computational issues we are trying to avoid in the case of large neural networks. In order to mitigate these concerns, we will make the additional approximation that λ_i and u_i do not change inside an iteration which will allow us to approximate changes to $\nabla_\theta E^T u_i$ and compare them against empirical observations. We note that we will not use this approximation for any other results.

Remark 4.2 *If we assume that λ_i, u_i do not change between iterations, if θ follows the PF then* $(\nabla_\theta \dot{E}^T u_i) = \frac{\log(1-h\lambda_i)}{h}\nabla_\theta E^T u_i$.

Remark 4.3 *If we assume that λ_i, u_i do not change between iterations, if θ follows the NGF we can write* $(\nabla_\theta \dot{E}^T u_i) = -\lambda_i \nabla_\theta E^T u_i$.

Remark 4.4 *If we assume that λ_i, u_i do not change between iterations, if θ follows the IGR flow we can write* $(\nabla_\theta \dot{E}^T u_i) = -\left(\lambda_i + \frac{h}{2}\lambda_i^2\right)\nabla_\theta E^T u_i$.

The above flows have the form $\dot{x} = cx$, with solution $x(t) = x(0)e^{ct}$. We can thus test these solutions empirically by training neural networks with gradient descent with learning rate h and at each step compute $\nabla_\theta E(\theta_t)^T (u_i)_{t-1}$ and compare it with the prediction $x(h)$ obtained from the solution from each flow initialized at the previous iteration, i.e. $x(0) = \nabla_\theta E(\theta_{t-1})^T (u_i)_{t-1}$. We show results with a VGG model trained on CIFAR-10 in Figure 8. The results show that the PF is substantially better than the NGF and IGR flows at predicting the behavior of $\nabla_\theta E^T u_0$. Since the NGF and the IGR flow solutions scale the initial value by the inverse of an exponential of magnitude given by λ_0 for large eigenvalues this leads to a small prediction, which is not aligned with what is observed empirically. We also note that the higher the value of $\nabla_\theta E^T u_0$, the worse the prediction of the PF; these are the areas where the approximations made in the above remarks are likely not to hold due to large gradient norms.

5 The principal flow, stability coefficients and edge of stability results

We now show the PF can be used to explain phenomena observed empirically when training neural networks using full-batch gradient descent.

Edge of stability results. Cohen et al. (2021) did a thorough empirical study to show that when training deep neural networks with full batch gradient descent the largest eigenvalue of the Hessian λ_0 keeps growing until reaching approximately $2/h$ (a phase of training they call *progressive sharpening*), after which it remains

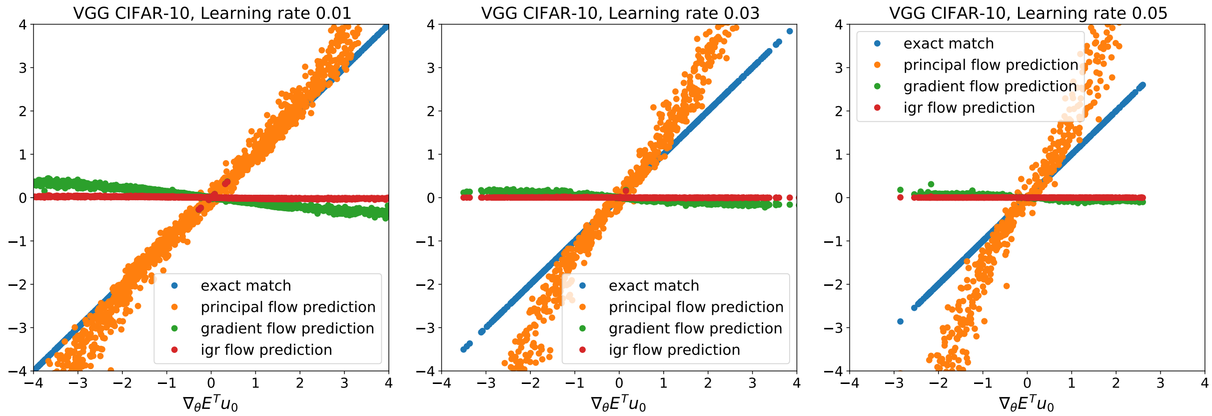


Figure 8: Predictions of $\nabla_\theta E^T u_0$ according to the NGF, IGR flow and the PF. On the x axis we plot the value of $\nabla_\theta E^T u_0$ as measured empirically in training, and on the y axis we plot the corresponding prediction according to the flows from the value of the dot product at the previous iteration. The 'exact match' line indicates a perfect prediction, the upper bound of performance. The PF performs best from all the compared flows, however for higher learning rates its performance degrades when $\nabla_\theta E^T u_0$ is large; this is due to the fact that the higher the learning rate and the higher the gradient norm, the more likely it is that the additional assumption we used that λ_i, u_i do not change does not hold.

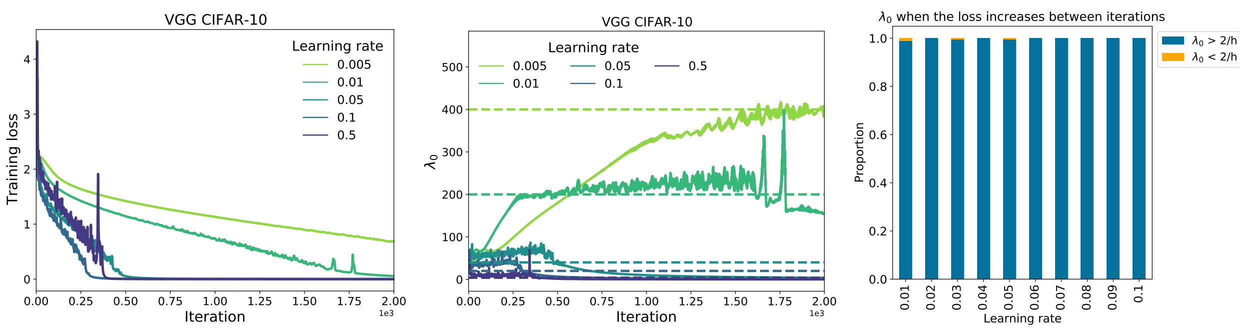
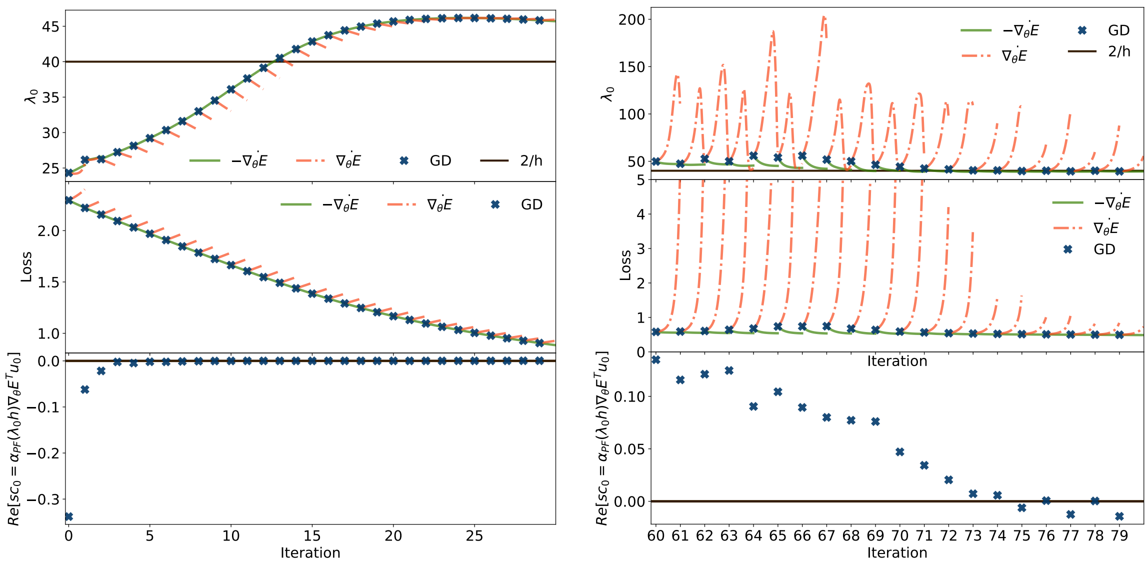


Figure 9: Edge of stability in neural networks (Cohen et al., 2021): instability occurs when $\lambda_0 > 2/h$.

in that area; for mean squared losses this continues indefinitely while for cross entropy losses they show it further decreases later in training. They also show that instabilities in training occur when $\lambda_0 > 2/h$. Their empirical study spans neural architectures, data modalities and loss functions. We visualize the edge of stability behavior they observe in Figure 9: training is stable until the $2/h$ threshold is achieved, and that eigenvalues keep growing until reaching the threshold; since we use a cross entropy loss, the eigenvalues later decrease in training. We also visualize that iterations where the loss increases compared to the previous iteration overwhelmingly occur when $\lambda_0 > 2/h$. Cohen et al. (2021) also empirically observe that $\theta^T u_0$ has oscillatory behavior in the edge of stability area but is 0 or small outside it.

Connection with the principal flow: stability coefficients. The PF contains some of the key quantities observed in the edge of stability phenomenon: the eigenvalues of the Hessian λ_i, the dot product $\nabla_\theta E^T u_i$ and the threshold $2/h$. All these quantities appear in the PF via the stability coefficient $sc_i = \frac{\log(1-h\lambda_i)}{h\lambda_i} \nabla_\theta E^T u_i = \alpha_{PF}(\lambda_i h) \nabla_\theta E^T u_i$ of eigendirection u_i. Through the PF, by connecting the case analysis in Section 3.1 with existing and new empirical observations, we can shed light on the edge of stability behavior in deep learning.

First phase of training (progressive sharpening): $\lambda_0 < 2/h$. This entails $Re[sc_i] = Re[\alpha_{PF}(h\lambda_i)] \leq 0, \forall i$ (Real stable and complex stable cases of the analysis in Section 3.1). $\text{sign}(\alpha_{NGF}) = \text{sign}(\alpha_{PF}) = -1$ and following the PF minimises E or its real part (Eq 15). To understand the behavior of λ_0, we now have to make use of empirical observations about the behavior of the NGF early in the training of neural networks. It has been empirically observed that in early areas of training, λ_0 increases here when following the NGF (Cohen et al.,



(a) Early training

(b) Edge of stability.

Figure 10: Understanding the edge of stability results using the PF on a 4 layer MLP: we plot the behavior of the NGF $\dot{\theta} = -\nabla_\theta E$ and the positive gradient flow $\dot{\theta} = \nabla_\theta E$ initialized at each gradient descent iteration parameters, and see that the behavior of gradient descent is connected to the behavior of the respective flow through the stability coefficient.

2021); we further show this in Figure 47 in the Appendix. Since in this part of training gradient descent follows closely the NGF, it exhibits similar behavior and λ_0 increases. We show this case in Figure 10(a).

Second phase of training (edge of stability) $\lambda_0 \geq 2/h$. This entails $Re[sc_0(\theta)] = Re[\alpha_{PF}(h\lambda_i)] \geq 0$. (Unstable complex case of the analysis in Section 3.1). We can no longer say that following the PF minimizes E. $\text{sign}(\alpha_{NGF}(h\lambda_0)) \neq \text{sign}(Re[(\alpha_{PF}(h\lambda_0)])$, since $\alpha_{NGF}(h\lambda_0) = -1$ and $\text{sign}(Re[(\alpha_{PF}(h\lambda_0)]) > 0$ meaning that in that direction gradient descent resembles the positive gradient flow $\dot{\theta} = \nabla_\theta E$ rather than the NGF. The positive gradient flow component can cause instabilities, and the strength of the instabilities depends on the stability coefficient $sc_0 = \alpha_{PF}(h\lambda_0)\nabla_\theta E^T u_0$. We show in Figures 10(b) and 12 how the behavior of the loss and λ_0 are affected by the behavior of the positive gradient flow when $\lambda_0 > 2/h$.

More than λ_0: the importance of stability coefficients. While the sign of the real part of the stability coefficient sc_0 is determined by λ_0, its magnitude is modulated by the dot product $\nabla_\theta E^T u_0$, since $sc_0 = \alpha_{PF}(h\lambda_0)\nabla_\theta E^T u_0$. The magnitude of $\nabla_\theta E^T u_0$ plays an important role, since if λ_0 is the only eigenvalue greater than $2/h$ training is stable if $\nabla_\theta E^T u_0 = 0$, as we see in Figure 10. *To understand instabilities, we have to look at stability coefficients, not only eigenvalues.* We show in Figure 11 how the instabilities in training can be related with the stability coefficient sc_0: the increases in loss occur when the corresponding $Re[sc_0]$ is positive and large. In Figure 12 we show the equivalent results in the behavior of λ_0: λ_0 increases or decreases based on the behavior of the corresponding flow and the strength of the stability coefficient and that gets reflected in instabilities in the loss function; specifically when $\lambda_0 > 2/h$, the increase both in loss value and λ_0 of gradient descent are proportional to the increase of the positive gradient flow in that area. We show additional results in Figures 35 and 36 in the Appendix.

Is one eigendirection enough to cause instability? One question that arises from the PF is whether the leading eigendirection u_0 can be sufficient to cause instabilities, especially in the context of deep networks with millions of parameters. To assess this we train a model with gradient descent until it reaches the edge of stability ($\lambda_0 \approx 2/h$), after which we approximate the continuous flow $\dot{\theta} = \nabla_\theta E^T u_0 u_0 + \sum_{i=1}^{D-1} -\nabla_\theta E^T u_i u_i$. The coefficients of the modified vector field of this flow are negative for all eigendirections except from u_0, which is positive; this is also the case for the PF when λ_0 is the only eigenvalue greater than $2/h$. In Figure 13 we show the results: a positive coefficient for u_0 can be responsible for an increase in loss value and a significant change in λ_0.

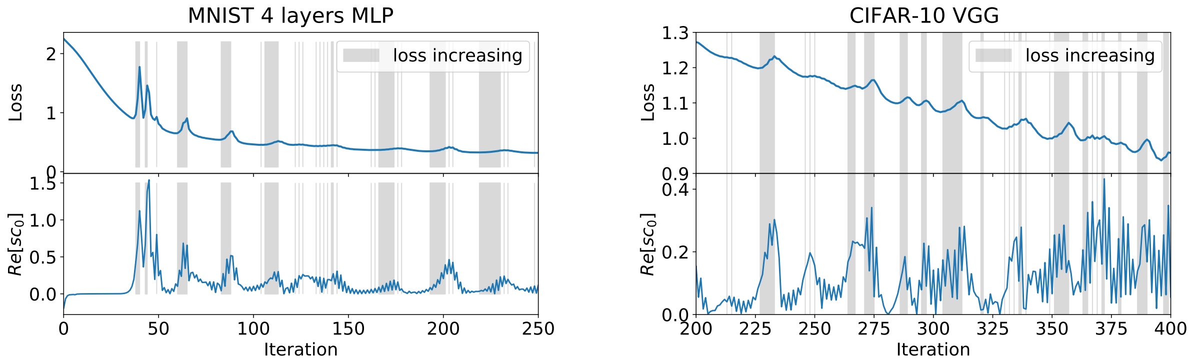


Figure 11: Understanding changes to the loss using the PF: areas where the loss increases corresponds to areas where the sc_0 is large. The highlighted areas correspond to regions where the loss increases.

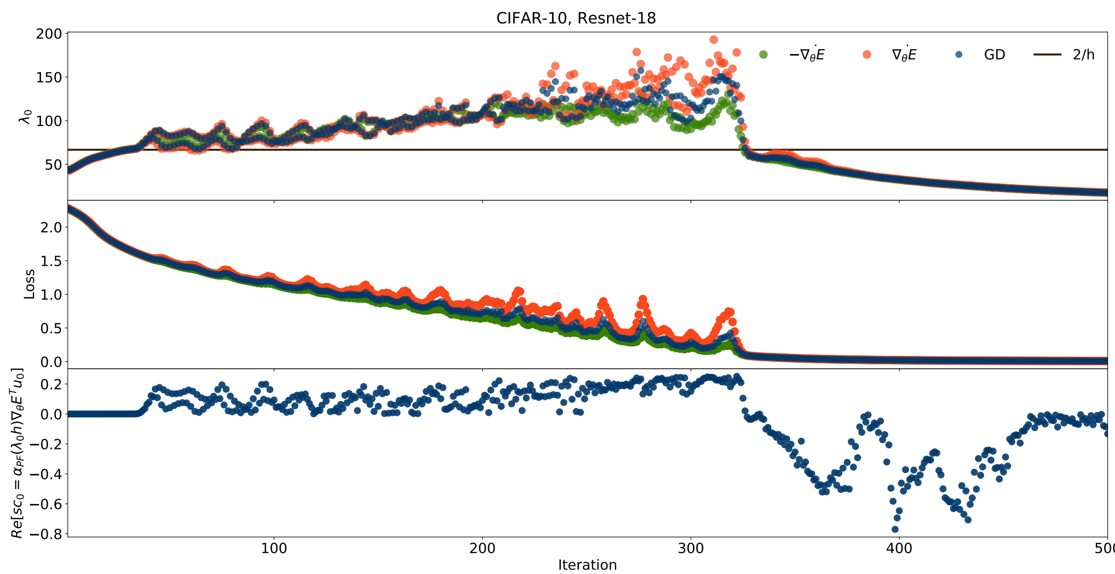


Figure 12: Understanding changes to the loss using the PF: areas where the loss increases corresponds with areas where the sc_0 is large. Together with the behavior of gradient descent, we also plot the behavior of the NGF and positive gradient flow initialized at θ_t and simulated for time h for each iteration t. As expected, the NGF decreases the loss around a gradient descent iteration, while the positive gradient flow increases it. Interestingly, when $\lambda_0 > 2/h$, the increases in loss value of gradient descent are proportional to the increase of the positive gradient flow in that area (can be seen best between iterations 200 and 350). The same behavior can be seen in relation to the eigenvalue λ_0.

Decreasing the learning rate. Cohen et al. (2021) show that if the edge of stability behavior is reached and the learning rate is decreased, the training stabilizes and λ_0 keeps increasing (Figure 32 in the Appendix). The PF tells us that decreasing the learning rate entails going from $Re[sc_0] \geq 0$ to $Re[sc_0] \leq 0$ since $\lambda_0 < 2/h$ after the learning rate change. Since all stability coefficients are now negative, this reduces instability. The increase in λ_0 is likely due to the behavior of the NGF in that area (as can be seen in Figure 13 when changing from gradient descent training to the NGF in an edge of stability area leads to an increase of λ_0).

The behavior of $\nabla_\theta E^T u_0$. The PF also allows us to explain the unstable behavior of $\nabla_\theta E^T u_0$ around edge of stability areas. As done in Section 4.1, we assume that λ_i, u_i do not change substantially between iterations and write $\dot{\nabla_\theta E^T u_i} = \frac{\log(1-h\lambda_i)}{h} \nabla_\theta E^T u_i$ under the PF, with solution $(\nabla_\theta E^T u_i)(t) = (\nabla_\theta E^T u_i)(0) e^{\frac{\log(1-h\lambda_i)}{h} t}$. This solution has different behavior depending on the value of λ_0 relative to $2/h$: decreasing below $2/h$ and increasing above $2/h$. We show this theoretically predicted behavior in Figure 14, alongside empirical

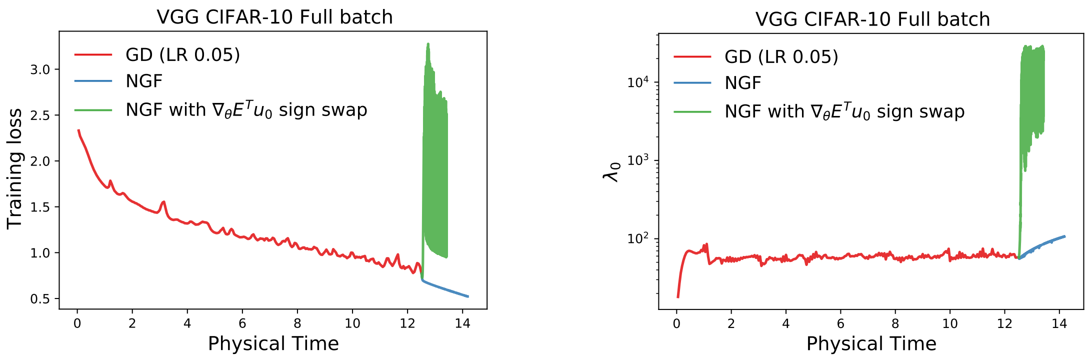


Figure 13: One eigendirection is sufficient to lead to instabilities. We construct a flow given by the NGF in all eigendirections but u_0; in the direction of u_0, we change the sign of the flow. This leads to the flow $\dot{\theta} = \nabla_\theta E^T u_0 u_0 + \sum_{i=1}^{D-1} -\nabla_\theta E^T u_i u_i$. We show this flow can be very unstable when initialised in an edge of stability area.

behavior showcasing the fluctuation of $\nabla_\theta E^T u_0$ in the edge of stability area, which confirms the theoretical prediction. We also compute the prediction error of the proposed flow and show it can capture the dynamics of $\nabla_\theta E^T u_0$ closely in this setting. We present a discrete time argument for this observation in Section A.7.2.

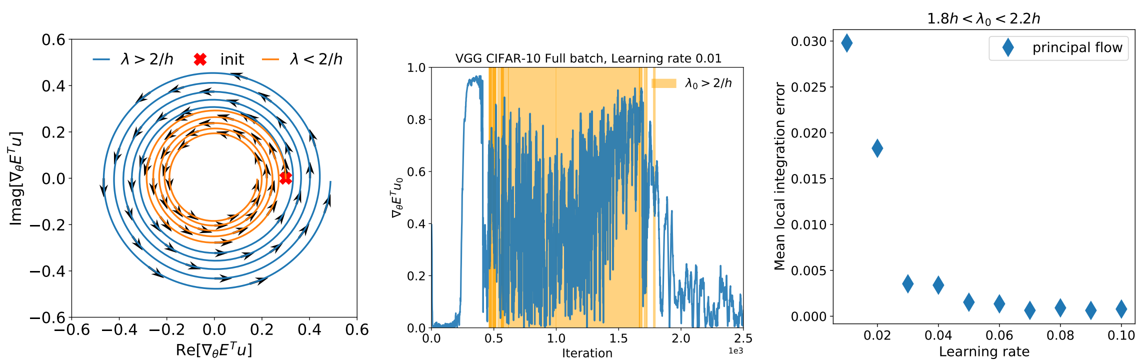


Figure 14: The unstable dynamics of $\nabla_\theta E^T u$ in the edge of stability area ($\lambda \approx 2/h$). Left: the predicted behavior of $\nabla_\theta E^T u$ under $\dot{\nabla_\theta E^T u_i} = \frac{\log(1-h\lambda_i)}{h} \nabla_\theta E^T u_i$, with an inflection point at $\lambda = 2/h$. Middle: empirical behavior of $\nabla_\theta E^T u$ for a model shows instabilities in the edge of stability area (highlighted). Right: the approximation made to derive the flow is suitable around $\lambda \approx 2/h$.

In this section we have shown that despite not capturing non-principal terms, the PF closely predicts the behavior of gradient descent in neural network training. When $\lambda_0 < 2/h$ the PF predicts stable behavior (Eq 15), which is consistent with empirical observations of gradient descent (RHS of Figure 9); this suggests that when $\lambda_0 < 2/h$ non-principal terms do not have a strong unstable presence (at least not sufficiently strong to overcome the stability coming from the PF). When $\lambda_0 > 2/h$, the PF predicts instability, which is what is observed in practice with gradient descent. While we do not know all non-principal terms and their behavior, in Section B in the Appendix we provide a justification for why the non-principal term we do know (Eq 14) can have a stabilizing effect. This evidence suggests that the PF plays a major role in capturing instability in deep learning, perhaps due to the specific structure of neural network models. While we focus on a continuous time approach, a discrete time approach can be used to motivate some of our observations and results (Section A.7); this is complementary to our approach which focuses on continuous time but nonetheless related, since it also ignores higher order derivatives of the loss and further suggests the strength of a quadratic approximation of the loss in the case of neural networks, as observed by Cohen et al. (2021).

6 Stabilizing training by adjusting discretization drift

The PF allows us to understand not only how gradient descent differs from the trajectory given by the NGF, but also when do they follow each other very closely. Understanding when gradient descent behaves like the NGF flow is important, since in those areas training can be sped up by increasing the learning rate. Prior works have empirically observed that gradient descent follows the NGF early in neural network training (Cohen et al., 2021) and this observation can be used to explain why decaying learning rates (Loshchilov and Hutter, 2016) or learning rate warm up (He et al., 2019) are successful when training neural networks: having a high learning rate in areas where the drift is small will not cause instabilities and can speed up training and decaying the learning rate avoids instabilities later in training when the drift is larger.

6.1 $\nabla_\theta^2 E \nabla_\theta E$ in determines discretization drift

In previous sections we have seen that the Hessian plays an important role in defining the PF. Using a condition on the Hessian, we can see when the NGF and the PF are the same:

Remark 6.1 *In a region of the space where $\nabla_\theta^2 E \nabla_\theta E = 0$ the PF is the same as the NGF.*

To see why, we can expand

$$\nabla_\theta^2 E \nabla_\theta E = \sum_{i=1}^{D} \lambda_i \nabla_\theta E^T u_i u_i. \tag{17}$$

By multiplying the above with u_j we have that $\lambda_j \nabla_\theta E^T u_j = 0$. For each dimension j either $\lambda_j = 0$ or $\nabla_\theta E^T u_j = 0$. If $\lambda_j = 0$ then $\alpha_{NGF}(h\lambda_j) = \alpha_{PG}(h\lambda_j) = -1$. We thus get that $\dot{\theta} = \sum_{i=1}^{D} \alpha_{PF}(h\lambda_i)(\nabla_\theta E^T u_i)u_i = \sum_{i=1}^{D} \alpha_{NGF}(h\lambda_i)(\nabla_\theta E^T u_i)u_i$.

This however does not tell us whether $\nabla_\theta^2 E \nabla_\theta E = 0$ implies that gradient descent follows the NGF, due to the existence of non-principal terms not captured by the PF. With further introspection we can show:

Theorem 6.1 *The discretization drift (error between gradient descent and the NGF) after 1 iteration of gradient descent $\theta_t = \theta_t - h\nabla_\theta E(\theta_{t-1})$ is $\frac{h^2}{2}\nabla_\theta^2 E(\theta')\nabla_\theta E(\theta')$ for a set of parameters θ' in the neighborhood of θ_{t-1}.*

This follows from the Taylor reminder theorem of the NGF in mean value form (proof in Section A.9). From here we have:

Corollary 6.1 *In a region of space where $\nabla_\theta^2 E \nabla_\theta E = 0$ gradient descent follows the NGF.*

Thus $\nabla_\theta^2 E \nabla_\theta E$ is a core quantity in the discretization drift of gradient descent; it is also strongly connected to the PF since $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|^2 = \left\|\sum_{i=1}^{D} \lambda_i \nabla_\theta E^T u_i u_i\right\|^2 = \sum_{i=1}^{D} \left\|\lambda_i \nabla_\theta E^T u_i\right\|^2$; the higher each term in the sum, the higher the difference between the NGF and the PF. To measure the connection between per iteration drift and $\nabla_\theta^2 E \nabla_\theta E$ in the neural network setting we approximate it via $\left\|\theta_t - \widetilde{NGF}(\theta_{t-1}, h)\right\|$ where $\widetilde{NGF}$ is the numerical approximation to the NGF initialised at θ_{t-1}. We show results in Figures 15 and 16, which show the strong correlation between per iteration drift and $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|$ throughout training and across learning rates. Since Theorem 6.1 tells us the form of the drift but not the exact value of θ', we have used θ_{t-1} instead to evaluate $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|$ and thus some error exists.

Understanding this connection is advantageous since computing discretization drift is computationally expensive as it requires approximating the continuous time NGF but computing $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|$ via Hessian-vector products is cheaper and approximations are available, such as $\nabla_\theta^2 E \nabla_\theta E = \frac{1}{2}\nabla_\theta \left\|\nabla_\theta E\right\|^2 \approx \frac{E(\theta + \epsilon\nabla_\theta E) - E(\theta)}{\epsilon}$ which only requires an additional backward pass Geiping et al. (2021).

6.2 Drift adjusted learning rate (DAL)

A natural question to ask is how to use the correlation between $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|$ and the iteration drift to improve training stability; $\left\|\nabla_\theta^2 E \nabla_\theta E\right\|$ captures all the quantities we have shown to be relevant to instability

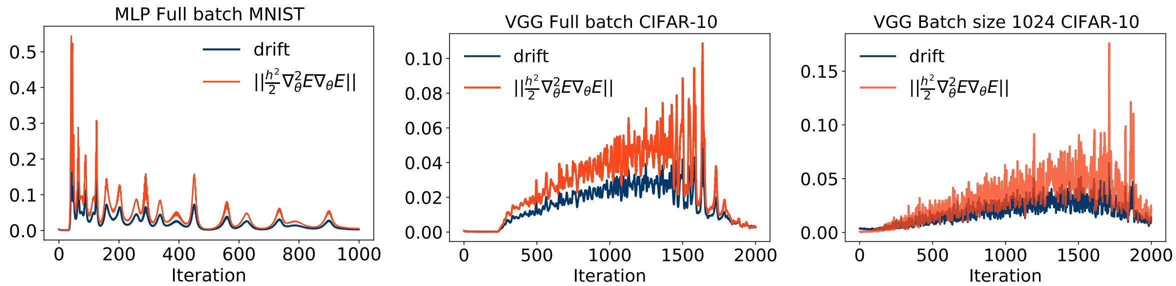


Figure 15: Connection between $||\nabla^2_\theta E \nabla_\theta E||$ and the per iteration drift as measured during training.

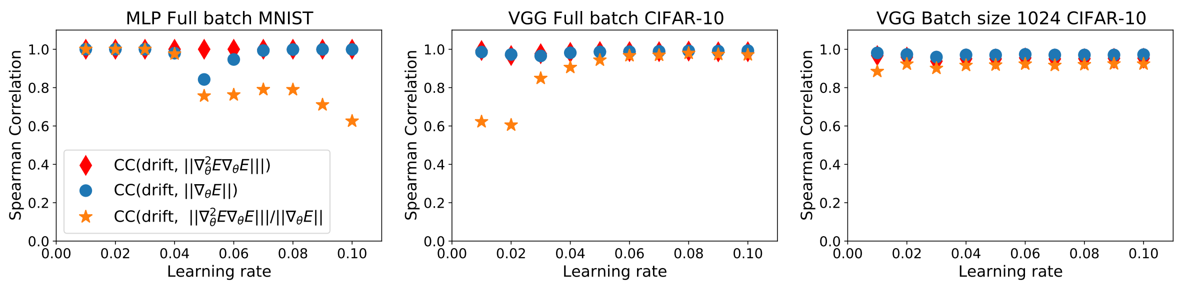


Figure 16: Correlation between $||\nabla^2_\theta E \nabla_\theta E||$ and the per iteration drift. Since $||\nabla^2_\theta E \nabla_\theta E|| = \left(||\nabla^2_\theta E \nabla_\theta E||\right) / ||\nabla_\theta E|| ||\nabla_\theta E||$, we plot the correlation with the individual terms as well.

highlighted by the PF: λ_i and $\nabla_\theta E^T u_i$ (Eq. 17). One way to use this information is to adapt the learning rate of the gradient descent update, such as using $\frac{2}{\|\nabla^2_\theta E \nabla_\theta E\|}$ as the learning rate. This learning rate slows down training when the drift is large — areas where instabilities are likely to occur — and it speeds up training in regions of low drift — areas where instabilities are unlikely to occur. Computing the norm of the update provided by this learning rate shows a challenge however since $2/\left\|\nabla^2_\theta E \nabla_\theta E\right\| \geq \frac{2}{\lambda_0 \|\nabla_\theta E\|}$; this implies that when using this learning rate the norm of the gradient descent update will never be 0 and thus training will not result in convergence. Furthermore, the magnitude of the gradient update will be independent of the gradient norm. To reinstate the gradient norm, we propose using the learning rate

$$h(\theta) = \frac{2}{\|\nabla^2_\theta E \nabla_\theta E\| \,/\, \|\nabla_\theta E\|} = \frac{2}{\|\nabla^2_\theta E \hat{g}(\theta)\|} \tag{18}$$

where $\hat{g}(\theta)$ is the unit normalised gradient $\nabla_\theta E / \|\nabla_\theta E\|$. We will call this learning rate **DAL** (Drift Adjusted Learning rate). As shown in Figure 15, $\left\|\nabla^2_\theta E \hat{g}(\theta)\right\|$ has a strong correlation with the per iteration drift.

We use DAL to set the learning rate and show results across architectures, models and datasets in Figures 17 (with additional results in Figure 39 in the Appendix). *Despite not requiring a learning rate sweep, DAL is stable compared to using fixed learning rates.* To provide intuition about DAL, we show the learning rate and the update norm in Figure 18: for DAL the learning rate decreases in training after which it slowly increases when reaching areas with low drift. Compared to larger learning static learning rates where the update norm can increase in the edge of stability area with DAL the update norm steadily decreases in training.

6.3 The trade-off between stability and performance

Since we are interested in understanding the dynamics of training gradient descent, we have so far focused on training performance. We now try to move our attention to test performance and generalization. Previous works (Li et al., 2019; Barrett and Dherin, 2021; Jastrzebski et al., 2019) have shown that the higher the learning rate the better the generalization performance. We now try to further correlate this information with the per iteration drift and the principal flow. To do so, we use learning rates with various degrees of

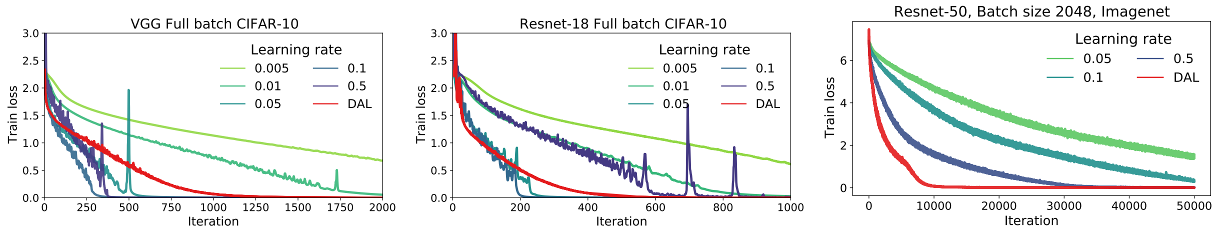


Figure 17: DAL: using the learning rate $\frac{2}{\|\nabla_\theta^2 E\hat{g}(\theta)\|}$ results in improved stability without requiring a hyper-parameter sweep.

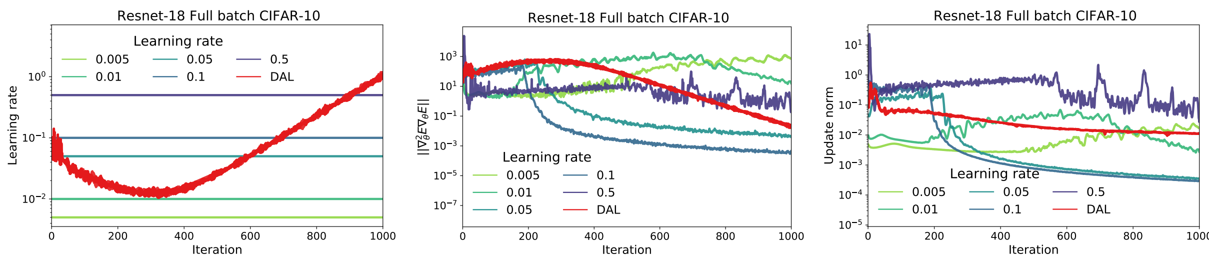


Figure 18: Key quantities in DAL versus fixed learning rate training: learning rate, and update norms.

sensitivity to iteration drift using DAL-p:

$$h_p(\theta) = \frac{2}{(\|\nabla_\theta^2 E\hat{g}(\theta)\|)^p} \tag{19}$$

The higher p, the slower the training and less drift there is; the lower p, there is more drift. We start with extensive experiments with $p = 0.5$, which we show in Figure 19, and show more results in Figure 40. Compared to $p = 1$ (DAL), there is faster training but at times also more instability. Performance on the test set shows that DAL-0.5 performs as well or better than when using fixed learning rates.

Remark 6.2 *We find that across datasets and batch sizes, DAL-*0.5 *performs best in terms of the stability generalization trade-off and in these settings can be used as a drop in replacement for a learning rate sweep.*

To further investigate the connection between drift and test set performance, we perform a set of sweeps over the power p and show results in Figure 20. These results show that the higher the drift (the smaller p), the more generalization. We also show in Figure 21 the correlation between mean per iteration drift and test accuracy both for learning rate and DAL-p sweeps. The results consistently show that the lower the mean iterations drift, the higher the test accuracy. We also show that the mean iteration drift has a connection to the largest eigenvalue λ_0: the lower the drift, the smaller λ_0. These results add further evidence to the idea that discretization drift is beneficial for generalization performance in the deep learning setting.

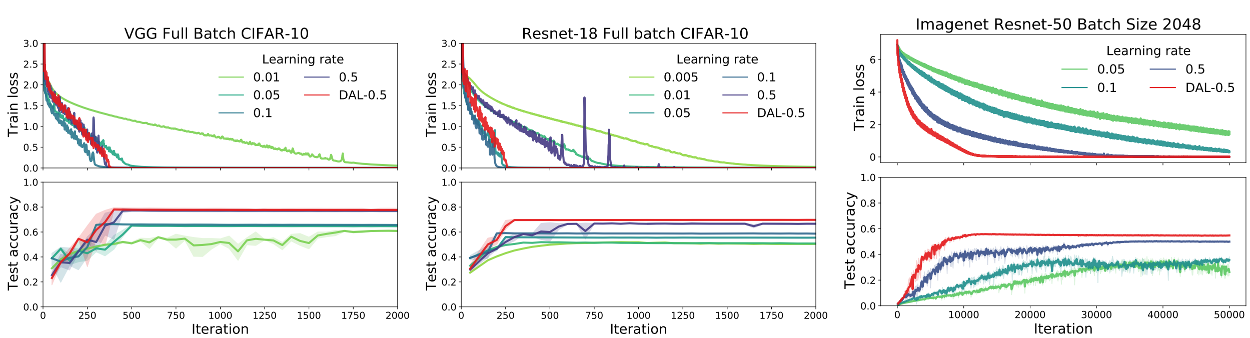


Figure 19: DAL-0.5: increased training speed and generalization compared to a sweep of fixed learning rates.

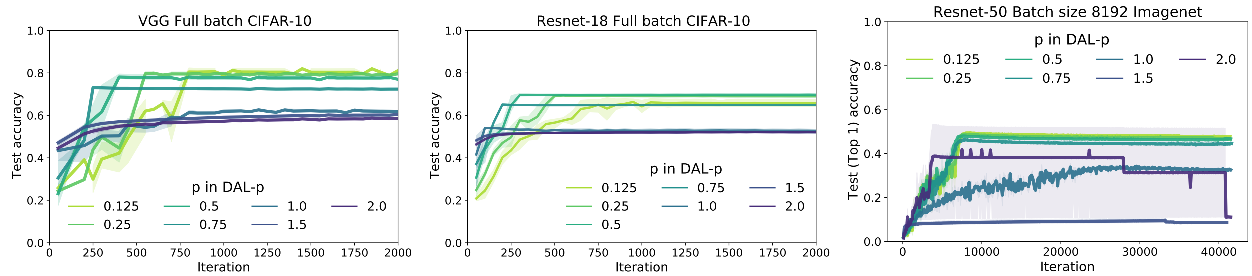
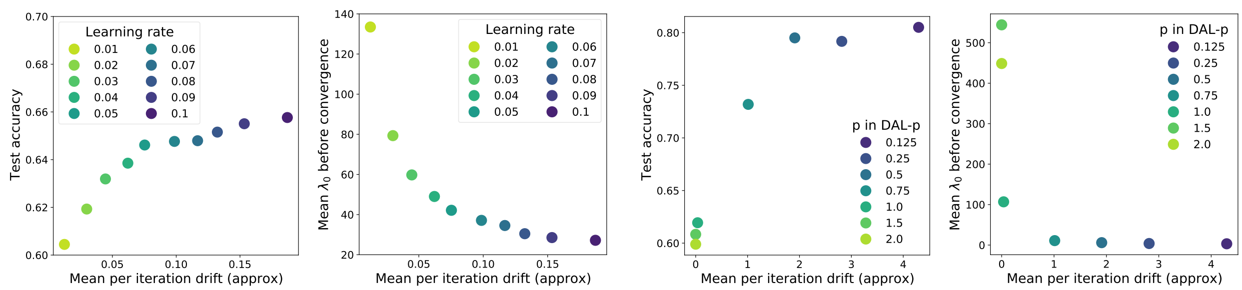


Figure 20: DAL-p sweep: discretization drift helps test performance at the cost of stability. Corresponding training curves and loss functions are present in the Figure 41 in the Appendix.



(a) Fixed learning rate sweep.

(b) DAL-p sweep.

Figure 21: The correlation between drift, test set performance and λ_0 in full batch training on CIFAR-10. The same pattern can be seen in SGD results in Figure 45.

7 Future work

Beyond gradient descent. In this work we focused on understanding vanilla gradient descent. Understanding discretization drift via the PF can be beneficial for improving other gradient based optimization algorithms as well, as we briefly illustrate for momentum updates with decay m and learning rate h:

$$v_t = mv_{t-1} - h\nabla_\theta E(\theta_{t-1}); \qquad \theta_t = \theta_{t-1} + v_t \tag{20}$$

We can scale $\nabla_\theta E(\theta_{t-1})$ in the above not by a fixed learning rate h, but by the approximation to the drift. This has two advantages: it removes the need for a learning rate sweep and it uses local information in adapting the moving average, such that in areas of large drift the contribution is decreased, while it is increased in areas where the drift is small (a more formal justification is provided in Section A.9). This leads to the following updates:

$$v_t = mv_{t-1} - \frac{1}{2||\nabla_\theta^2 E(\theta_{t-1})\hat{g}(\theta)(\theta_{t-1})||}\nabla_\theta E(\theta_{t-1}) \qquad \theta_t = \theta_{t-1} + v_t \tag{21}$$

As with DAL-p, we can use powers to control the stability performance trade-off: the lower p, the more the current update contribution is reduced in high drift (instability) areas. We tested this approach on Imagenet and show results in Figure 22. The results show that integrating drift information improves the speed of convergence compared to standard gradient descent (Figure 20), and leads to more stable training compared to using a fixed learning rate. We present additional experimental results in the Appendix.

Just as momentum is a common staple of optimization algorithms, so are adaptive schemes such as Adam (Kingma and Ba, 2015) and Adagrad (Duchi et al., 2011), which adjust the step taken for each parameter independently. We can also use the knowledge from the PF to set a per parameter learning rate: instead of using $\left\|\nabla_\theta^2 E(\theta_{t-1})\nabla_\theta E(\theta_{t-1})\right\|$ to set a global learning rate, we can use the per parameter information provided by $\nabla_\theta^2 E(\theta_{t-1})\nabla_\theta E(\theta_{t-1})$ to adapt the learning rate of each parameter. We present preliminary

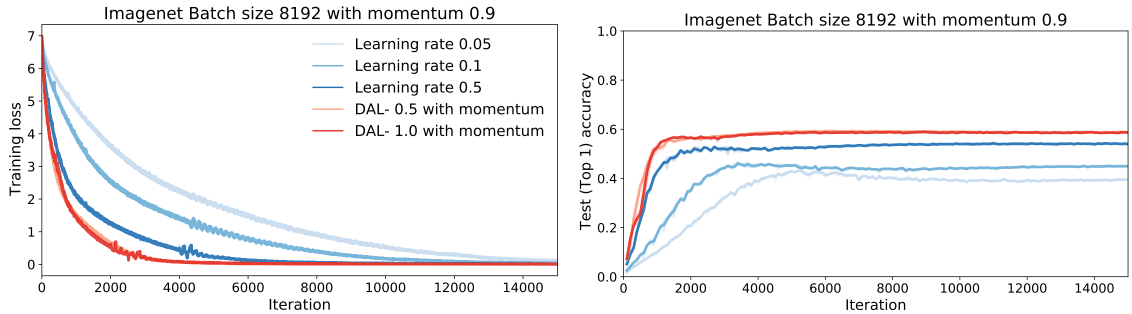


Figure 22: DAL with momentum: integrating drift information results in faster and more stable training compared to a fixed learning rate sweep. Compared to vanilla gradient descent there is also a significant performance and convergence speed boost.

results in the Appendix (Figures 43 and 44). The above two approaches (momentum and per-parameter learning rate adaptation) can be combined, bringing us closer to the most commonly used deep learning optimization algorithms. While we do not explore this avenue here, we are hopeful that this understanding of discretization drift can be leveraged further to stabilize and improve deep learning optimization.

Non-principal terms. This work focuses on understanding the effects of the principal flow on the behavior of gradient descent. The principal terms however are not the only terms in the discretization drift: we have found one non-principal term (Eq 8) and have seen that it can have a stabilising effect (Figure 6). We provide a preliminary explanation for the stabilising effect of this non-principal term together with results measuring its value in neural network training in Section B in the Appendix; we hope that future work will enhance our understanding of non-principal terms and their effects on training stability and generalization.

Incorporating neural network knowledge into theory. We have obtained the PF from the BEA of gradient descent on a general loss function E and have used it to explain empirically observed behavior of neural networks. The PF however does not account for the specific structure of the function E and further understanding might be provided by using the compositionality of simple functions specific to neural networks. An avenue for future work would be to explore arguments used to understand the NGF for neural networks (Du et al., 2018; Elkabetz and Cohen, 2021; Kunin et al., 2021) and expand them to the PF.

8 Related work

Modified flows for deep learning optimization. Barrett and Dherin (2021) found the first order correction modified flow for gradient descent using BEA and uncovered its regularization effects; they were the first to show the power of BEA in the deep learning context. Smith et al. (2021) find the first order error correction term in expectation during one epoch of stochastic gradient descent. Modified flows have also been used for other optimizers than vanilla gradient descent: Franca et al. (2020); Shi et al. (2021) compare momentum and Nesterov accelerated momentum; Kunin et al. (2021) study the symmetries of deep neural networks and use modified vector fields to show commonly used discrete updates break conservation laws present when using the NGF (for gradient descent they use the IGR flow while for momentum and weight decay they introduce different flows); Kovachki and Stuart (2021) use modified flows to understand the behavior of momentum by approximating Hamiltionian systems; França et al. (2021) construct optimizers controlling their stability and convergence rates while Li et al. (2017) construct optimizers with adaptive learning rates in the context of stochastic differential equations. In the context of two-player games, Rosca et al. (2021) compute the first order BEA correction terms while Chavdarova et al. (2021) use high-resolution differential equations to shed light on the properties of different saddle point optimizers.

Edge of stability and the importance of the Hessian. There have been a number of empirical studies on the Hessian in gradient descent. Cohen et al. (2021) observed the edge of stability behavior and performed an extensive study which led to many empirical observations used in this work. Jastrzębski et al. (2018) performed a similar study in the context of stochastic gradient descent. Sagun et al. (2017); Ghorbani et al. (2019); Papyan (2018) approximate the entire spectrum of the Hessian, and show that there are only a few negative eigenvalues, plenty of eigenvalues centered around 0, and a few positive eigenvalues with large magnitude. Similarly, Gur-Ari et al. (2018) discuss how gradient descent operates in a small subspace. Lewkowycz et al. (2020) discuss the large learning rate catapult in deep learning when the largest eigenvalue exceeds $2/h$. Gilmer et al. (2021) assess the effects of the largest Hessian eigenvalue in a large number of empirical settings.

There have been a series of concurrent works aimed at theoretically explaining the empirical results above. Ahn et al. (2022) connect the edge of stability behavior with what they coin as the 'relative progress ratio': $\frac{E(\theta-h\nabla_\theta E)-E(\theta)}{h\|\nabla_\theta E\|^2}$, which they empirically show is 0 in stable areas of training and 1 in the edge of stability areas. To see the connection between the relative progress ratio and the quantities discussed in this paper, one can perform a Taylor expansion on $\frac{E(\theta-h\nabla_\theta E)-E(\theta)}{h\|\nabla_\theta E\|^2} \approx \frac{-h\nabla_\theta E^T\nabla_\theta E + h^2/2\nabla_\theta E^T\nabla_\theta^2 E\nabla_\theta E}{h\|\nabla_\theta E\|^2} = -1 + h/2\frac{\nabla_\theta E^T\nabla_\theta^2 E\nabla_\theta E}{\|\nabla_\theta E\|^2}$. While this ratio is related to the quantities we discuss, we also note significant differences: it is a scalar, and not a parameter length vector and thus does not capture per eigendirection behavior as we see with the stability coefficients (Section 5). Arora et al. (2022) prove the edge of stability result occurs under certain conditions either on the learning rate or on the loss function. Ma et al. (2022) empirically observe the multi-scale structure of the loss landscape in neural networks and use it to theoretically explain the edge of stability behavior of gradient descent. Chen and Bruna (2022) use low dimensional theoretical insights around a local minima to understand the edge of stability behavior. These important works are complementary to our own work; they do not use continuous time approaches and tackle primarily the edge of stability problem or its subcases, while we focus on understanding gradient descent and applying that understanding broadly, including but not limited to the edge of stability phenomenon.

Understanding the difference between the negative gradient flow and discrete gradient descent. Elkabetz and Cohen (2021) recently examined the differences between gradient descent and the NGF in the deep learning context; their work examines the importance of the Hessian in determining when gradient descent follows the NGF. Their theoretical results show that neural networks are roughly convex and thus for reasonably sized learning rates one can expect that gradient descent follows the NGF flow closely. Their results complement ours and their approach might be extended to help us understand why the PF is sufficient to shed light on many instability observations in the neural network training.

Second-order optimization. By using second order information (or approximations thereof) to set the learning rate, DAL is related to second-order approaches used in deep learning. Many second-order methdos can be seen as approximates of Newton's method $\theta_t = \theta_{t-1} - \nabla_\theta^2 E^{-1}(\theta_{t-1})\nabla_\theta E(\theta_{t-1})$. Since computing the inverse of Hessian can be prohibitively expensive for large models, many practical methods approximate it with tractable alternatives (Martens and Grosse, 2015).

Connection between drift and generalization. We have made the connection between increased drift and increased generalization. This connection was first made by Barrett and Dherin (2021) through the IGR flow. Generalization has also been connected to the largest eigenvalue λ_0(Hochreiter and Schmidhuber, 1997; Keskar et al., 2016; Jastrzębski et al., 2018); recently Kaur et al. (2022) however showed a more complex picture, primarily in the context of stochastic gradient descent. The largest eigenvalue could be a confounder to the drift as we have observed in Section 6.3; we hope that future work can deepen these connections.

9 Conclusion

We have expanded on previous works which used backward error analysis in deep learning to find a new continuous time flow, called **the principal flow**, to analyze the behavior of gradient descent. We were able to show that the principal flow captures the behavior of gradient descent better than existing continuous time flows, including but not limited to instability and oscillatory behavior. Through the eigendecomposition of the Hessian we used the principal flow to shed light on newly observed empirical phenomena, such as the edge of stability results. After understanding the core quantities connected to instabilities in deep learning we devised an automatic learning rate schedule, DAL, which exhibits stable training. We concluded by cementing the connection between large discretization drift and increased generalization performance. Throughout this work we hope to have exemplified the potential of using continuous time methods to understand optimization.

References

Paul Glendinning. *Stability, instability and chaos: an introduction to the theory of nonlinear differential equations*. Cambridge university press, 1994.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in neural information processing systems*, pages 5585–5595, 2017.

Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv preprint arXiv:1809.10374*, 2018.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

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

Gal Vardi and Ohad Shamir. Implicit regularization in relu networks with the square loss. In *Conference on Learning Theory*, pages 4224–4258. PMLR, 2021.

Guilherme Franca, Jeremias Sulam, Daniel Robinson, and Rene Vidal. Conformal symplectic and relativistic optimization. In *Conference on Neural Information Processing Systems (NeurIPS 2020)*. 2020.

David GT Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

Ernst Hairer, Marlis Hochbruck, Arieh Iserles, and Christian Lubich. Geometric numerical integration. *Oberwolfach Reports*, 3(1):805–882, 2006.

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instability in deep learning. *arXiv preprint arXiv:2110.04369*, 2021.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The computer journal*, 3(3):175–184, 1960.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

Jonas Geiping, Micah Goldblum, Phillip E Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. *arXiv preprint arXiv:2109.14119*, 2021.

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

Stanislaw Jastrzebski, 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*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

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

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 31, 2018.

Daniel Kunin, Javier Sagastuy-Brena, and Hidenori Tanaka Ganguli, Surya Daniel L.K. Yamins. Symmetry, conservation laws, and learning dynamics in neural networks. In *International Conference on Learning Representations*, 2021.

Bin Shi, Simon S Du, Michael I Jordan, and Weijie J Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, pages 1–70, 2021.

Nikola B Kovachki and Andrew M Stuart. Continuous time analysis of momentum methods. *Journal of Machine Learning Research*, 22(17):1–40, 2021.

Guilherme França, Michael I Jordan, and René Vidal. On dissipative symplectic integration with applications to gradient-based optimization. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):043402, 2021.

Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, volume 70, pages 2101–2110, 2017.

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

Tatjana Chavdarova, Michael I Jordan, and Manolis Zampetakis. Last-iterate convergence of saddle point optimizers via high-resolution differential equations. *arXiv preprint arXiv:2112.13826*, 2021.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Kwangjun Ahn, Jingzhao Zhang, and Suvrit Sra. Understanding the unstable convergence of gradient descent. *arXiv preprint arXiv:2204.01050*, 2022.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on edge of stability in deep learning. *arXiv preprint arXiv:2205.09745*, 2022.

Chao Ma, Lei Wu, and Lexing Ying. The multiscale structure of neural network loss functions: The effect on optimization and origin. *arXiv preprint arXiv:2204.11326*, 2022.

Lei Chen and Joan Bruna. On gradient descent convergence beyond the edge of stability. *arXiv preprint arXiv:2206.04172*, 2022.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Simran Kaur, Jeremy Cohen, and Zachary C Lipton. On the maximum hessian eigenvalue and generalization. *arXiv preprint arXiv:2206.10654*, 2022.

Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261 (276):2, 1995.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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*, pages 770–778, 2016.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.