

DeepMind



# The importance of discretisation drift in deep learning

Mihaela Rosca  
2/08/2023



# Discussed works

- *Implicit Gradient Regularisation* [Barrett and Dherin, ICLR 2021](#)
- *Discretization drift in two player games* [Rosca et al, ICML 2021](#)
- *On a continuous time model of gradient descent dynamics and instability in deep learning*  
[Rosca et al, TMLR 2023](#)
- *On the Origin of Implicit Regularization in Stochastic Gradient Descent* [Smith et al, ICLR 2021](#)
- *Implicit regularisation in stochastic gradient descent: from single-objective to two-player games*, [Rosca et al, 2023, ICML workshop](#)

*There is a lot of interesting related work, check out the related work section in these papers!*

*You can also find an overview in [my thesis](#), “On discretisation drift and smoothness regularisation in deep learning”.*

**Aim: analyse gradient descent updates aiming to minimise function  $E$ .**

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})$$

parameter dimension:  $D$

$h$  denotes the learning rate throughout this talk.



Continuous dynamics

$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} E$$

Euler integration



Discrete dynamics

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})$$

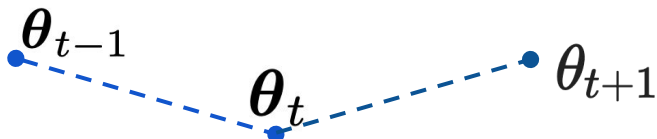
**Gradient descent!**

# Approaches of analysing optimisation

## Discrete time

- analyse updates as used in practice
- directly accounts for the learning rate

$t - 1 \rightarrow t$                        $t \rightarrow t + 1$

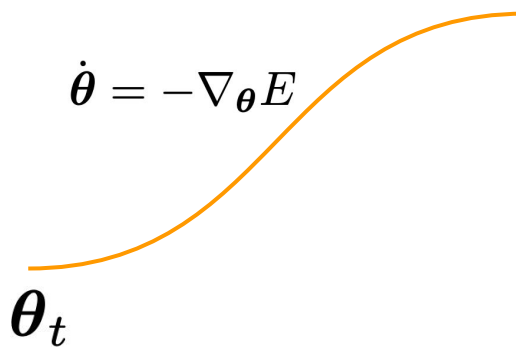


$$\theta_t = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1})$$

## Continuous time

- analyse the underlying continuous system
- tends to be easier analytically
- the gradient flow does not account for learning rates (there can be a gap between continuous analysis results and what happens in practice)

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



## Appeal and challenges of continuous time

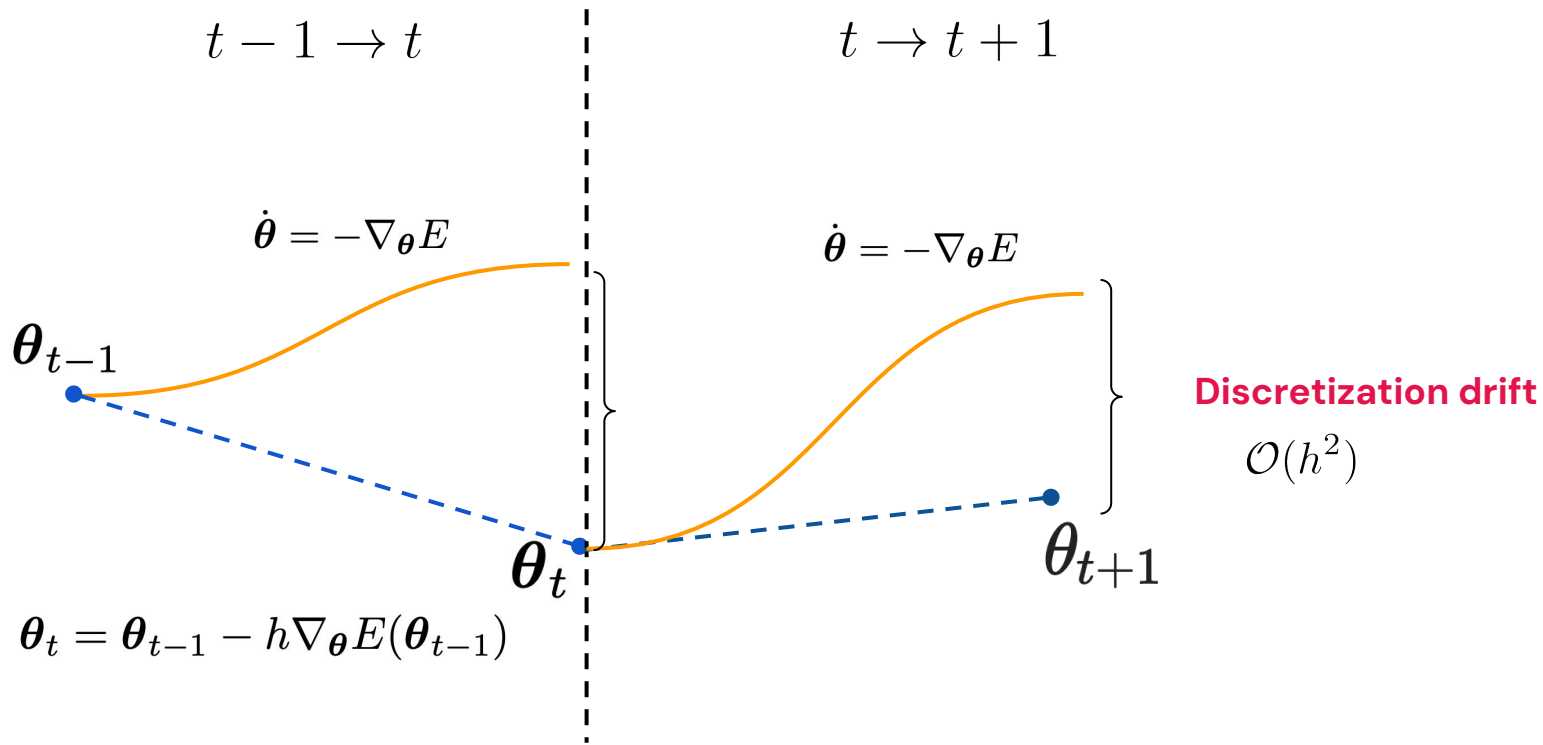
Negative gradient flow (NGF):  $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} E$

$$\frac{dE}{dt} = \frac{dE}{d\boldsymbol{\theta}}^T \frac{d\boldsymbol{\theta}}{dt} = -(\nabla_{\boldsymbol{\theta}} E)^T \nabla_{\boldsymbol{\theta}} E = -\|\nabla_{\boldsymbol{\theta}} E\|^2 \leq 0$$

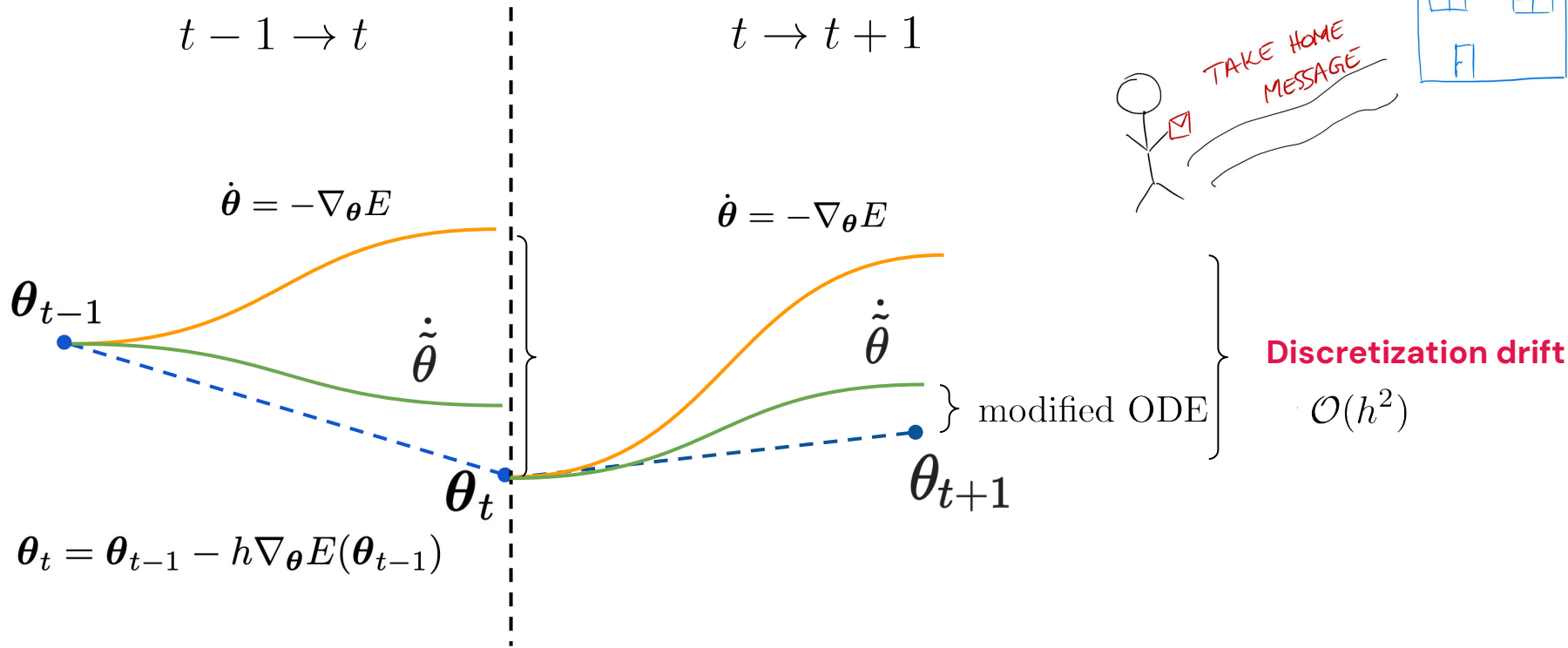
**following the NGF decreases E and that's easy to prove, but that is not true of gradient descent!**



# Discretization drift for Euler updates



# Discretization drift for Euler updates





# Modified loss functions

$$t - 1 \rightarrow t$$

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

Minimises original loss function E.

$\theta_{t-1}$

$$\dot{\tilde{\theta}} = -\nabla_{\theta} \tilde{E}$$

Minimises modified loss function.  
Closer to what gradient descent does.  
We can find out which losses are implicit  
minimised when we train models with gradient  
descent.

$\theta_t$

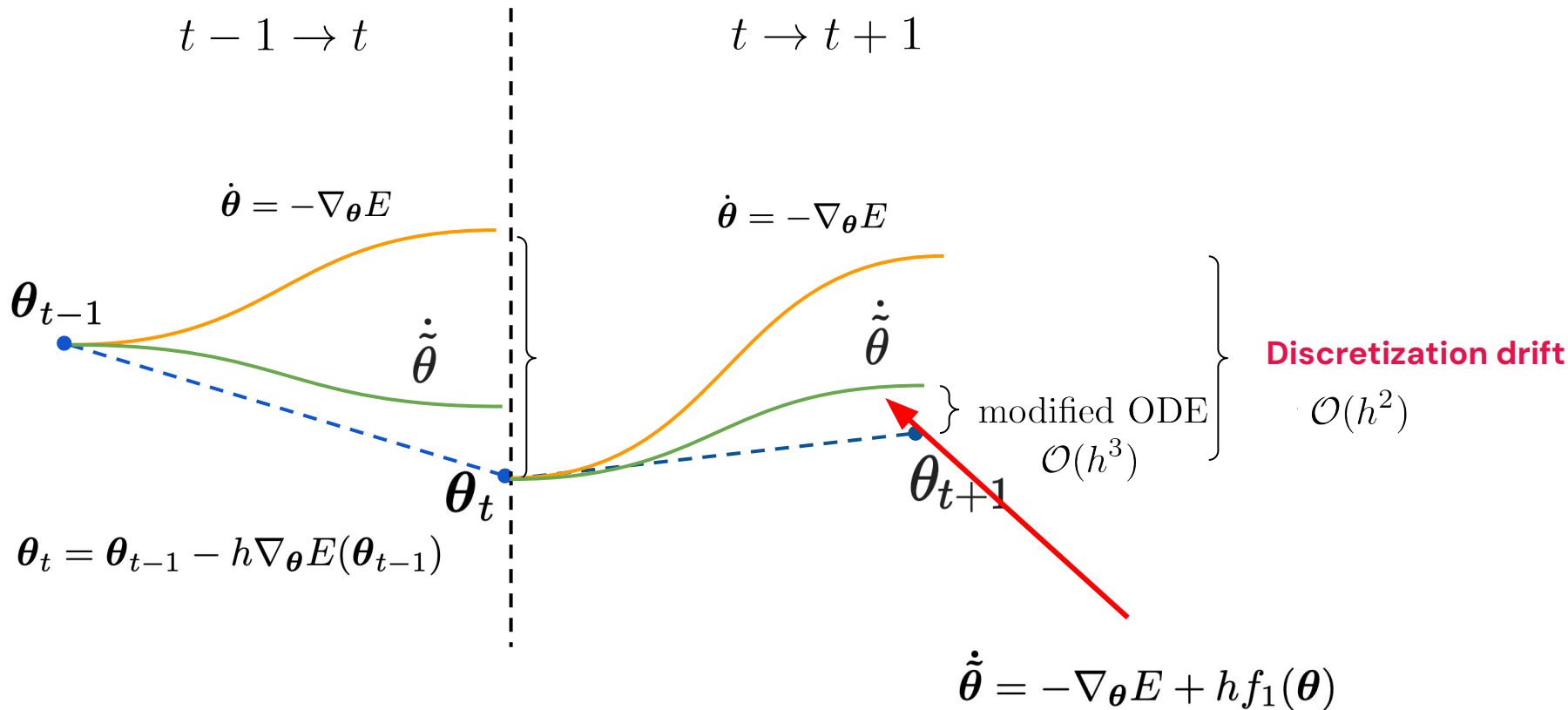
$$\theta_t = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1})$$



# Implicit gradient regularisation

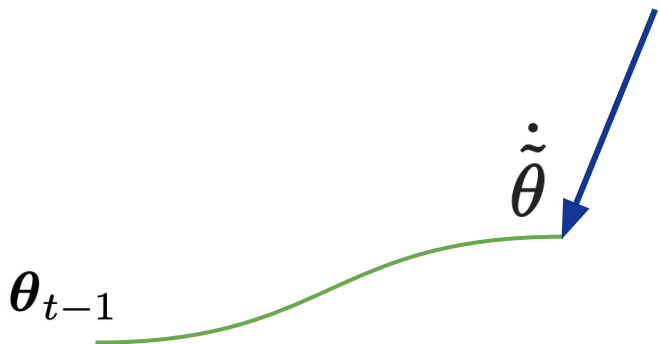
**Barrett et Dherin, ICLR 2021**

# Backward error analysis



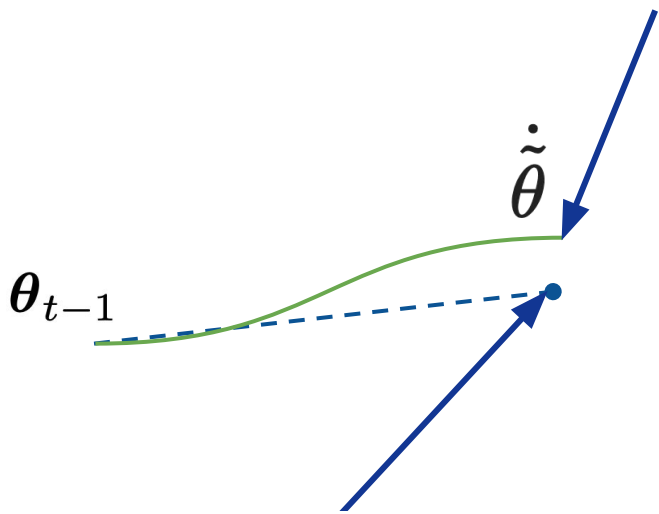
## BEA proof visualisation

$$\tilde{\theta}(h) = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1}) + h^2 \left[ f_1(\theta_{t-1}) + \frac{1}{2} \nabla_{\theta}^2 E(\theta_{t-1}) \nabla_{\theta} E(\theta_{t-1}) \right] + \mathcal{O}(h^3)$$



## BEA proof visualisation

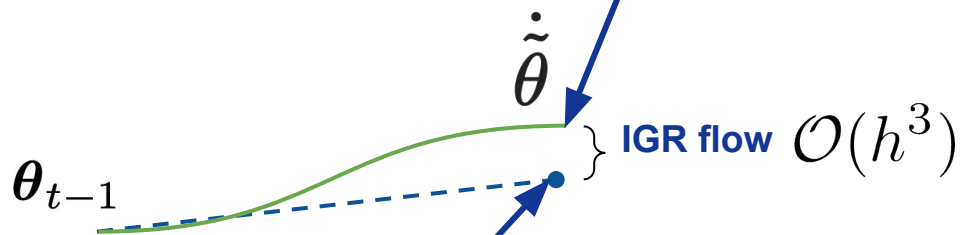
$$\tilde{\theta}(h) = \theta_{t-1} - h\nabla_{\theta}E(\theta_{t-1}) + h^2 \left[ f_1(\theta_{t-1}) + \frac{1}{2} \nabla_{\theta}^2 E(\theta_{t-1}) \nabla_{\theta} E(\theta_{t-1}) \right] + \mathcal{O}(h^3)$$



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

## BEA proof visualisation

$$\tilde{\theta}(h) = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1}) + h^2 \left[ f_1(\theta_{t-1}) + \frac{1}{2} \nabla_{\theta}^2 E(\theta_{t-1}) \nabla_{\theta} E(\theta_{t-1}) \right] + \mathcal{O}(h^3)$$

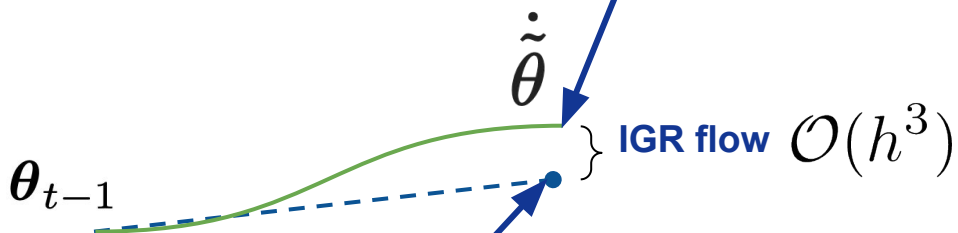


$$\theta_t = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1})$$

## BEA proof visualisation

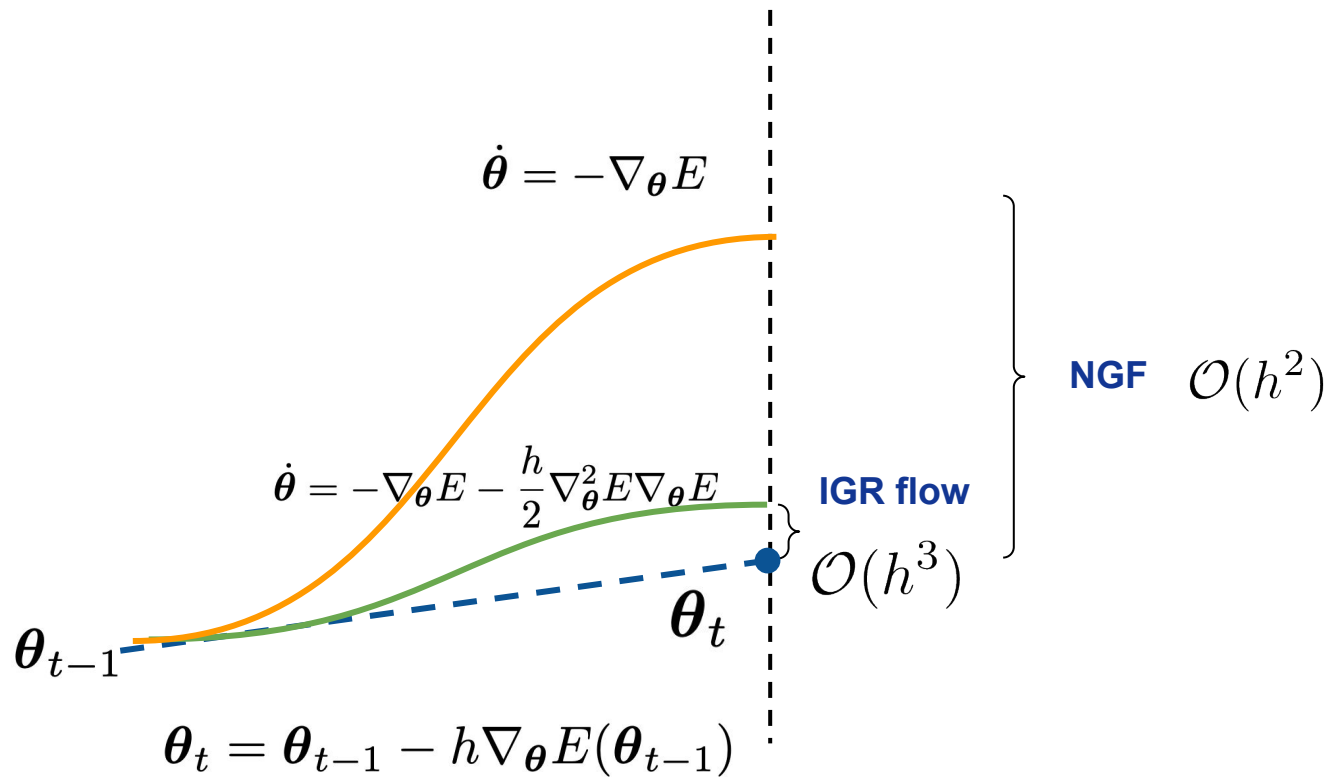
$$\tilde{\theta}(h) = \theta_{t-1} - h \nabla_{\theta} E(\theta_{t-1}) + h^2 \left[ f_1(\theta_{t-1}) + \frac{1}{2} \nabla_{\theta}^2 E(\theta_{t-1}) \nabla_{\theta} E(\theta_{t-1}) \right] + \mathcal{O}(h^3)$$

=0



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

# IGR flow



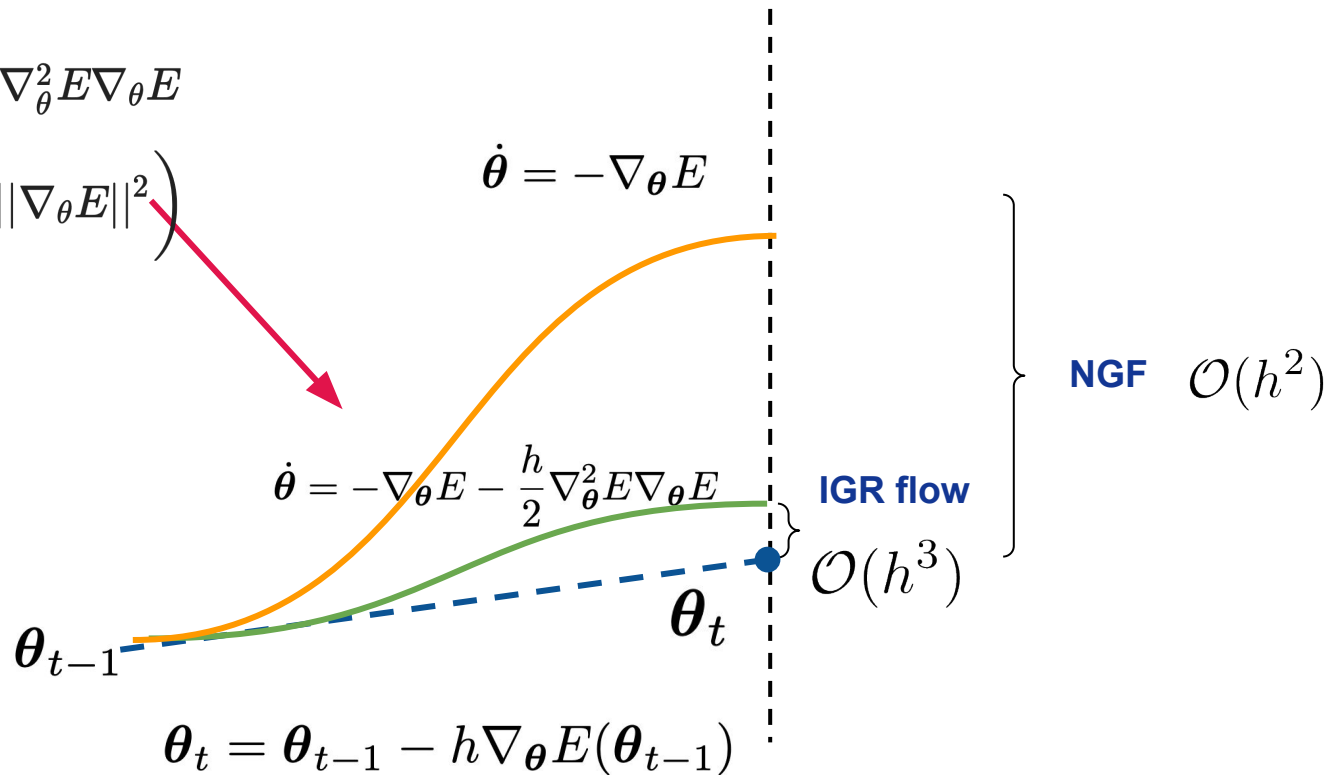


# From modified ODEs to modified losses: vector fields as negative gradients

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

$$= -\nabla_{\theta} \left( E + \frac{h}{4} \|\nabla_{\theta} E\|^2 \right)$$

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



# Modified loss functions

$t - 1 \rightarrow t$

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

Minimises  $E$ .

$\theta_{t-1}$

$$\dot{\theta} = -\nabla_{\theta} \left( E + \frac{h}{4} \|\nabla_{\theta} E\|^2 \right)$$

Minimises  $\tilde{E} = E + \frac{h}{4} \|\nabla_{\theta} E\|^2$



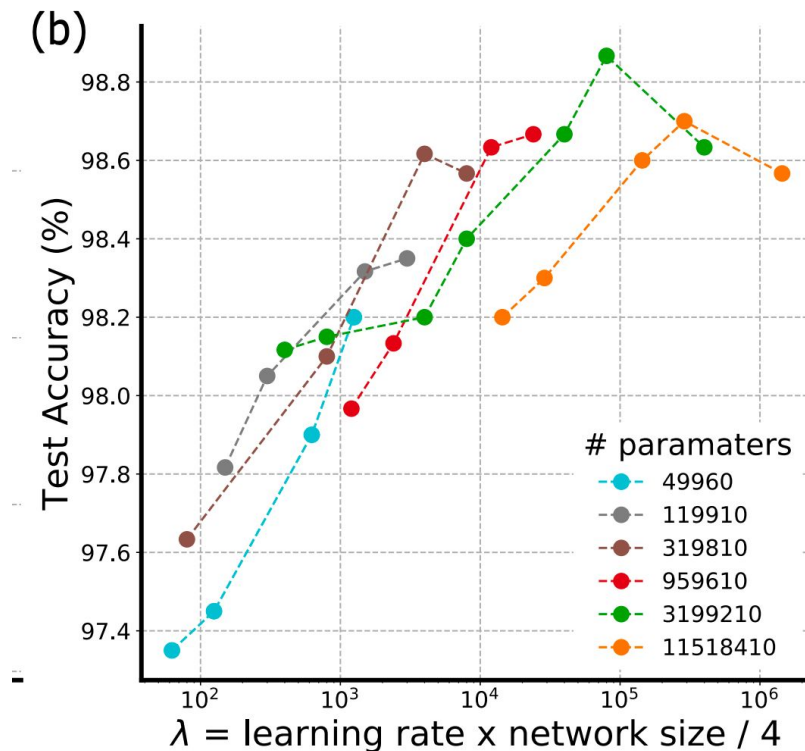
## Modified losses in supervised learning

$$\tilde{E} = E + \underbrace{\frac{h}{4} \|\nabla_{\theta} E\|^2}_{\text{Implicit gradient regularization}}$$

Implicit gradient regularization

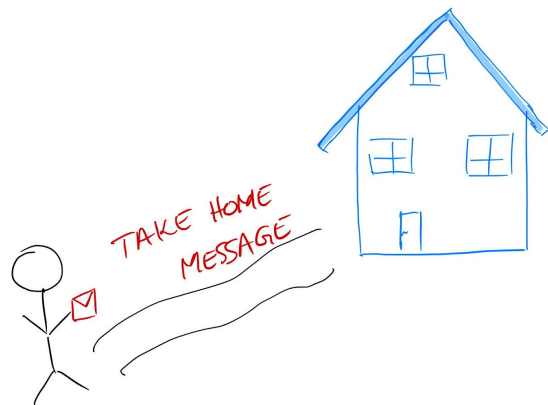
# Implicit gradient regularization

Implicit gradient regularization aids generalization.



# Summary

- In supervised learning, gradient descent introduces an implicit regularisation effect.
- Higher learning rates increase regularisation and help generalisation.



**How about instability?**

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

Rosca et al, TMLR 2023

**Our goal: use a continuous time perspective to analyse gradient descent, including its observed instabilities.**





# Existing continuous time flows

Negative gradient flow (NGF):

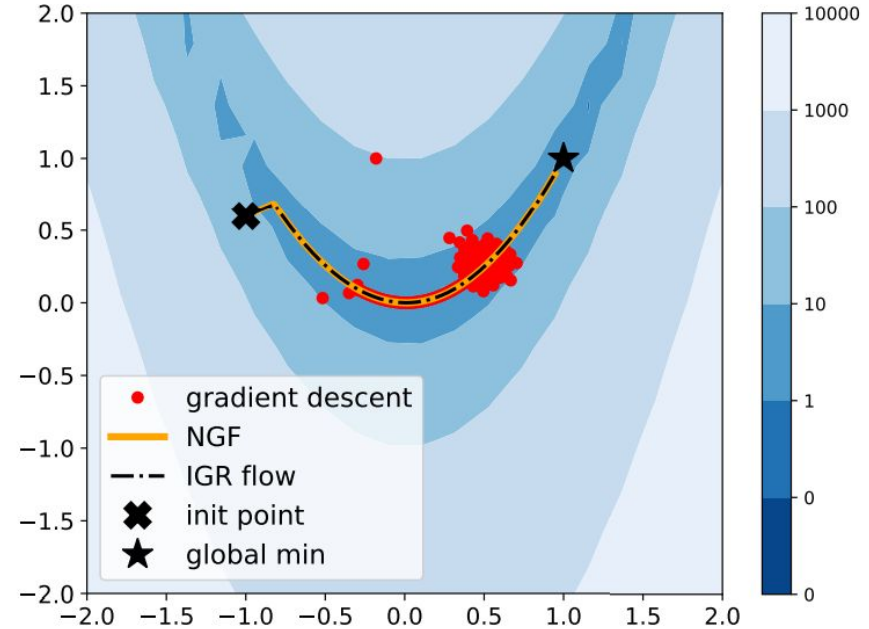
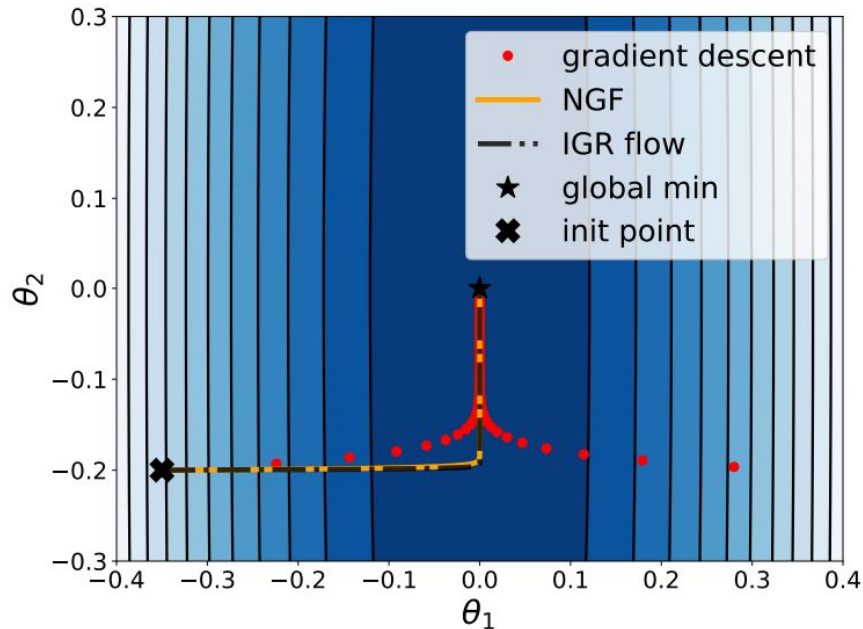
$$\dot{\theta} = -\nabla_{\theta} E$$

Implicit Gradient Regularization  
flow (IGR flow):

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



# Limitations of existing continuous time flows

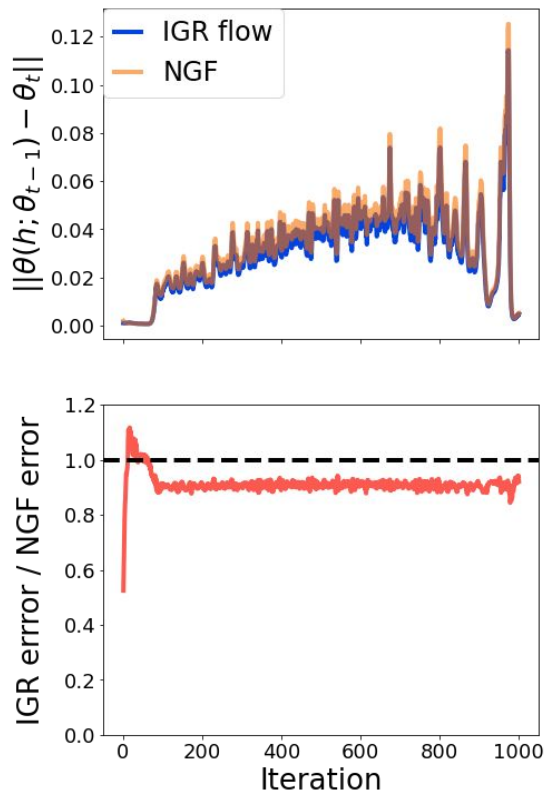


Existing continuous time flows do not handle instabilities observed using gradient descent.



# Plenty of space to improve: deep learning

CIFAR-10, VGG



## Our approach: Backward Error Analysis

Find


$$\dot{\tilde{\theta}} = -\nabla_{\theta} E + h f_1(\theta) + \cdots + h^n f_n(\theta)$$

such that the difference between gradient descent and the continuous time flow after 1 discrete update is  $\mathcal{O}(h^{n+2})$ .

The correction terms  $f_i$  will depend on  $E$  and its derivatives.



## Our approach: Backward Error Analysis

$$\dot{\tilde{\theta}} = -\nabla_{\theta} E + h f_1(\theta) + \cdots + h^n f_n(\theta)$$


Our approach: look at all correction terms (in all  $n$ ) which only contain first and second order derivatives of  $E$ .



# The Principal Flow

Hessian  $\nabla_{\theta}^2 E$  eigenvalues and eigenvectors

$$\dot{\theta} = \sum_{i=0}^{D-1} \frac{\log(1 - h\lambda_i)}{h\lambda_i} (\nabla_{\theta} E^T \mathbf{u}_i) \mathbf{u}_i$$

Properties of the Principal Flow (PF):

- is exact for quadratic losses
- generalisation of the NGF
- stability analysis correctly predicts that gradient descent is not always attracted to local minima



## The Principal Flow

PF

$$\dot{\boldsymbol{\theta}} = \sum_{i=0}^{D-1} \frac{\log(1 - h\lambda_i)}{h\lambda_i} (\nabla_{\boldsymbol{\theta}} E^T \mathbf{u}_i) \mathbf{u}_i$$

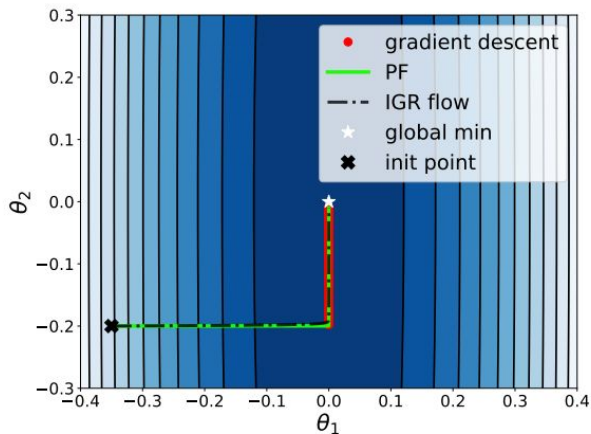
stability coefficient in eigendirection with index  $i$

NGF

$$\dot{\boldsymbol{\theta}} = \sum_{i=0}^{D-1} -(\nabla_{\boldsymbol{\theta}} E^T \mathbf{u}_i) \mathbf{u}_i$$

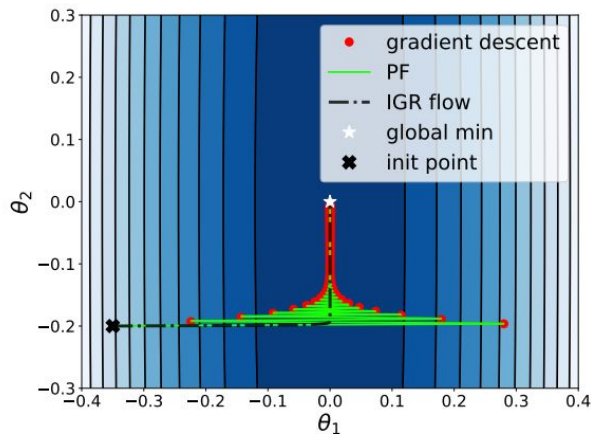


# The Principal Flow: exact for quadratic case



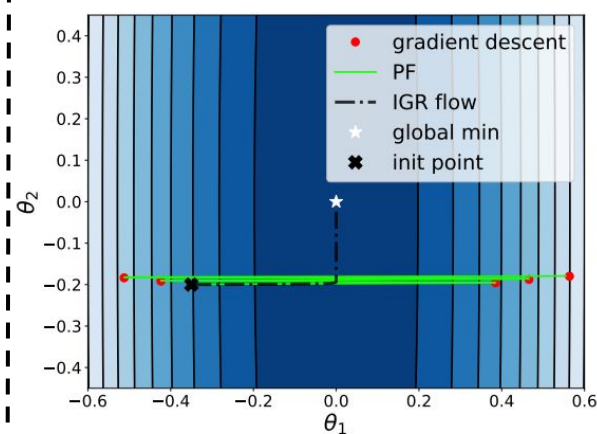
(a)  $\lambda_0 < 1/h$  (stability)

Stability coefficient is real and negative.



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

Stability coefficient is complex, with negative real part.

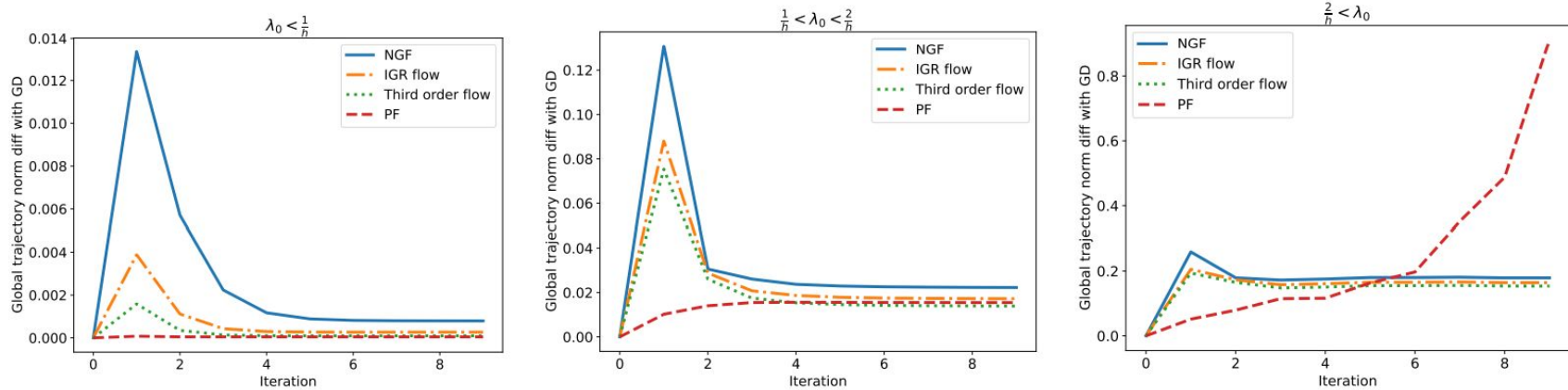


(c)  $\lambda_0 > 2/h$  (divergence)

Stability coefficient is complex, with positive real part.



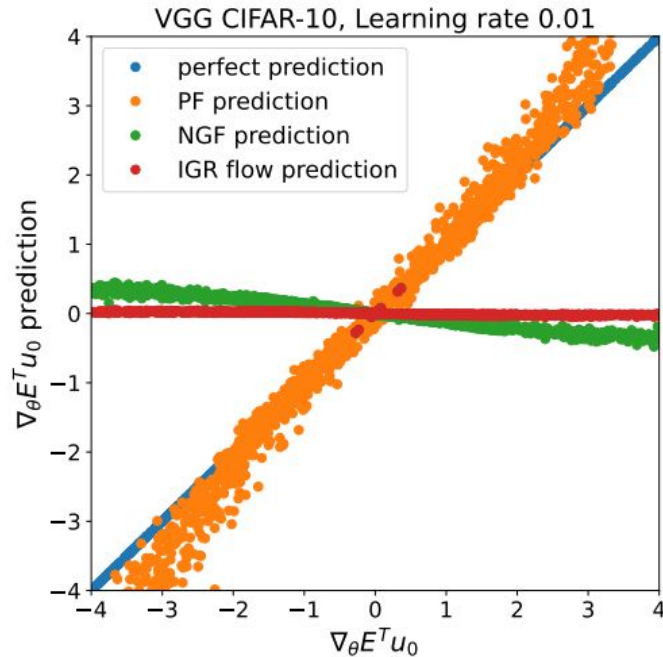
# The Principal Flow and neural networks: small neural nets



**The Principal Flow captures gradient descent better than existing flows in neural network training locally (when the number of iterations is small) both in the stable and unstable cases.**



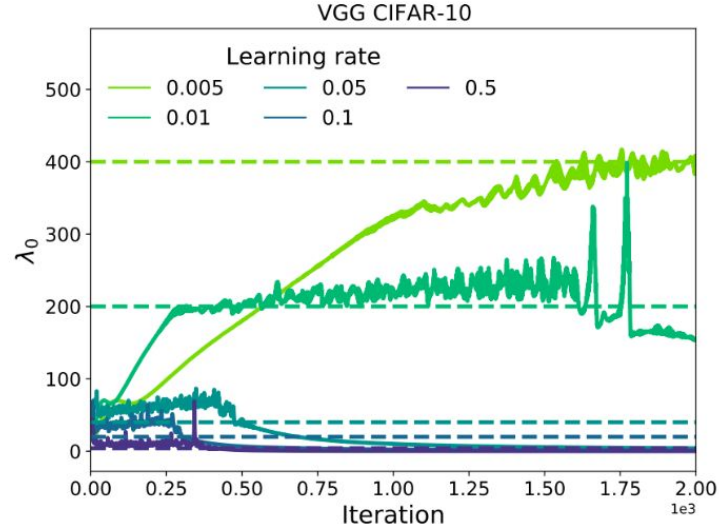
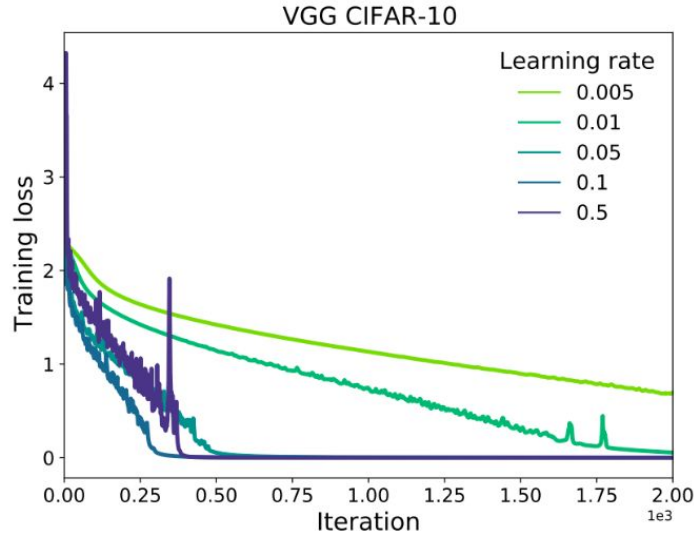
# The Principal Flow and neural networks



The Principal Flow captures the dynamics of the dot product between the gradient and the leading Hessian eigenvector better than existing flows.



# The edge of stability in deep learning



The edge of stability phenomenon in deep learning has been coined by Cohen et al, 2021.

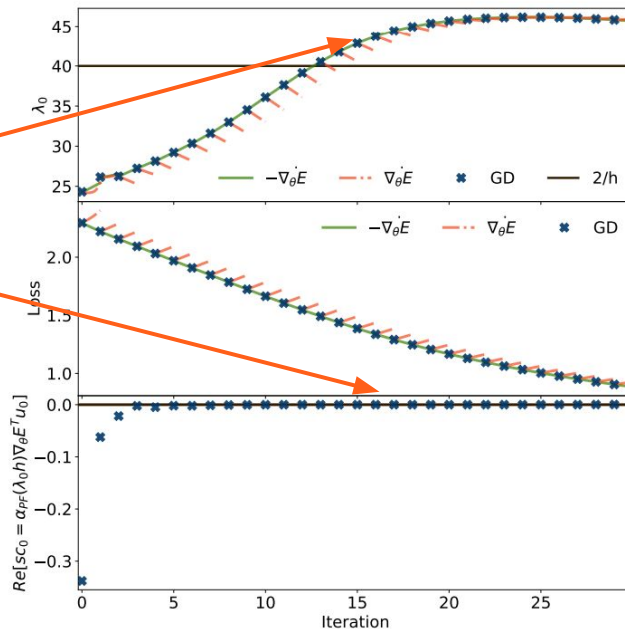


# The Principal Flow and edge of stability: more than just eigenvalues

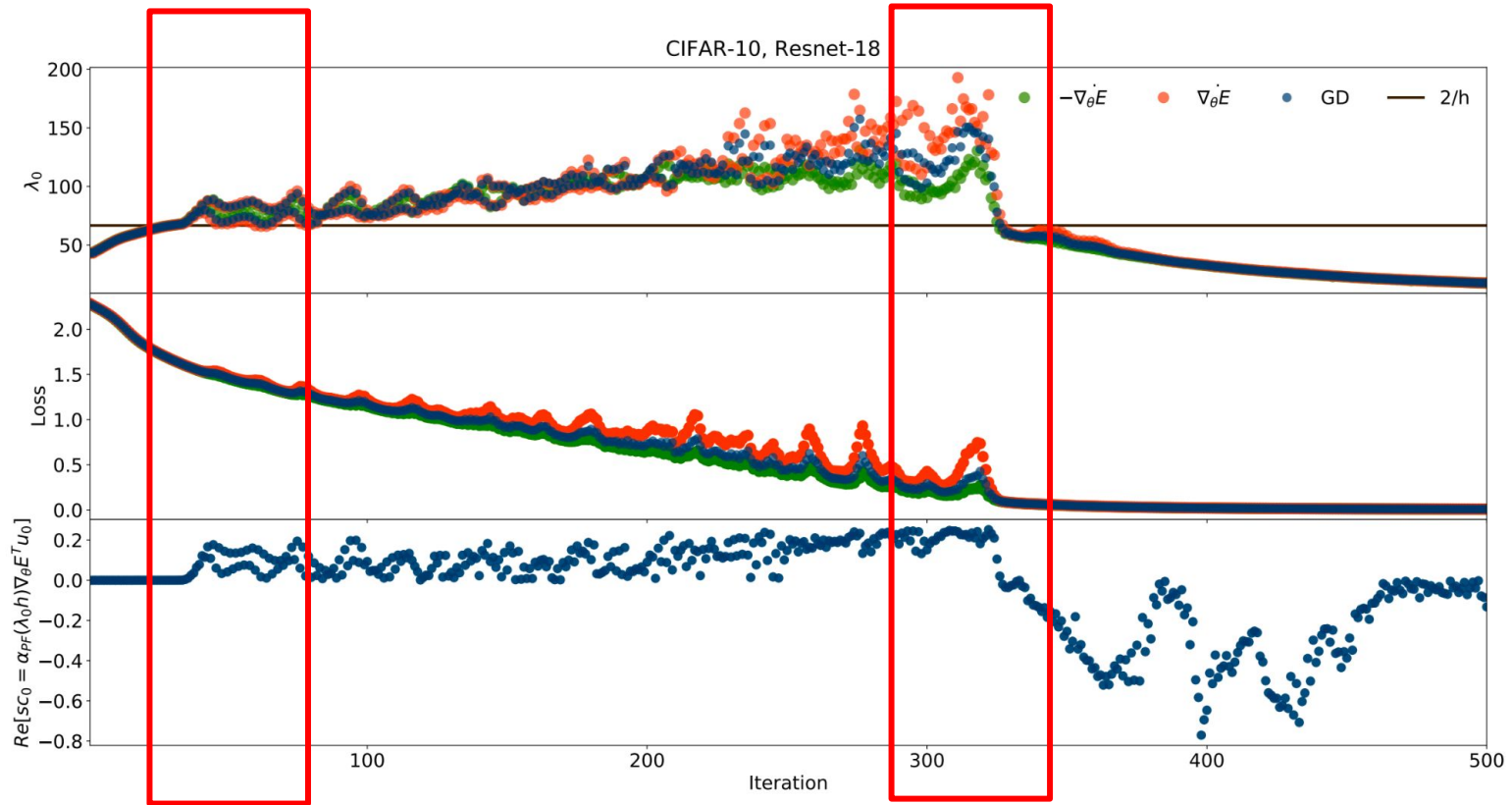
The stability coefficients show the strength of instabilities according to the local behaviour of other flows.

$$\dot{\theta} = \sum_{i=0}^{D-1} \frac{\log(1 - h\lambda_i)}{h\lambda_i} (\nabla_{\theta} E^T \mathbf{u}_i) \mathbf{u}_i$$

stability coefficient in eigendirection with index  $i$



# The Principal Flow and edge of stability



The stability coefficients show the strength of instabilities according to the local behaviour of other flows.



# Stabilising training



# Understanding total drift

The PF reveals an important quantity

$$\nabla_{\boldsymbol{\theta}}^2 E \nabla_{\boldsymbol{\theta}} E = \sum_{i=1}^D \lambda_i \nabla_{\boldsymbol{\theta}} E^T \mathbf{u}_i \mathbf{u}_i.$$



# Understanding total drift

Investigating further reveals:

$$\nabla_{\theta}^2 E \nabla_{\theta} E = 0 \quad \longrightarrow \quad \text{PF} = \text{NGF}$$





# Understanding total drift

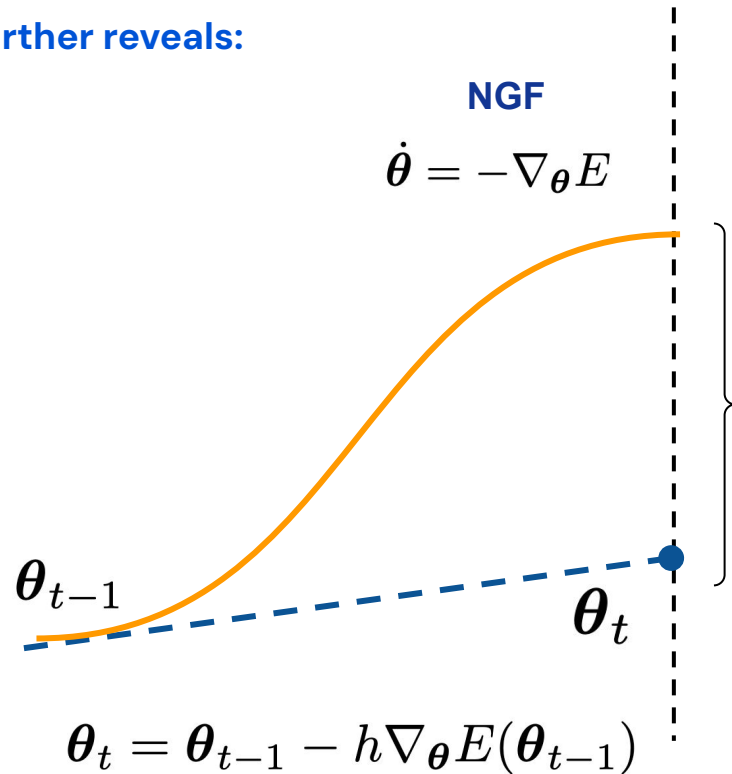
Investigating further reveals:

$$\nabla_{\theta}^2 E \nabla_{\theta} E = 0 \quad \longrightarrow \quad \text{GD} = \text{NGF}$$



# Understanding total discretisation drift

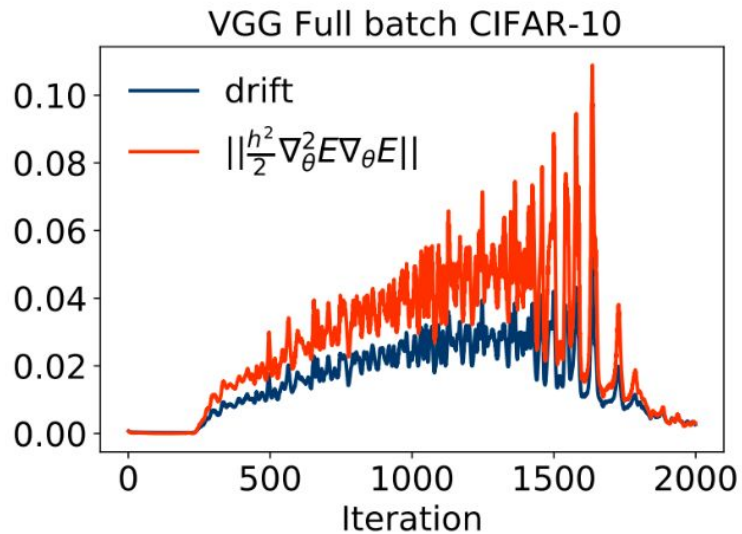
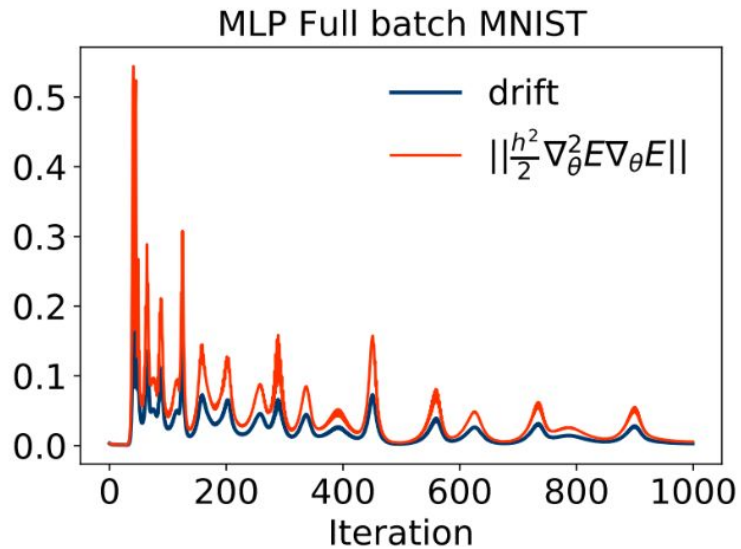
Investigating further reveals:



$$\frac{h^2}{2} \nabla_{\theta}^2 E(\theta') \nabla_{\theta} E(\theta')$$

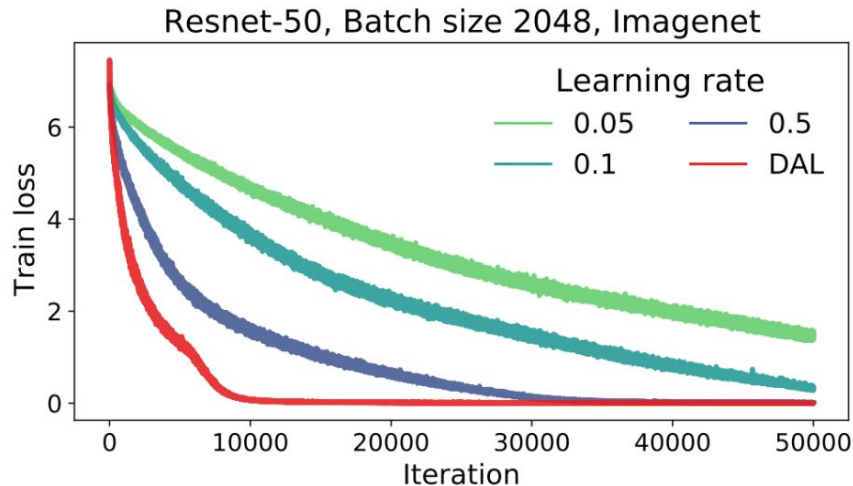
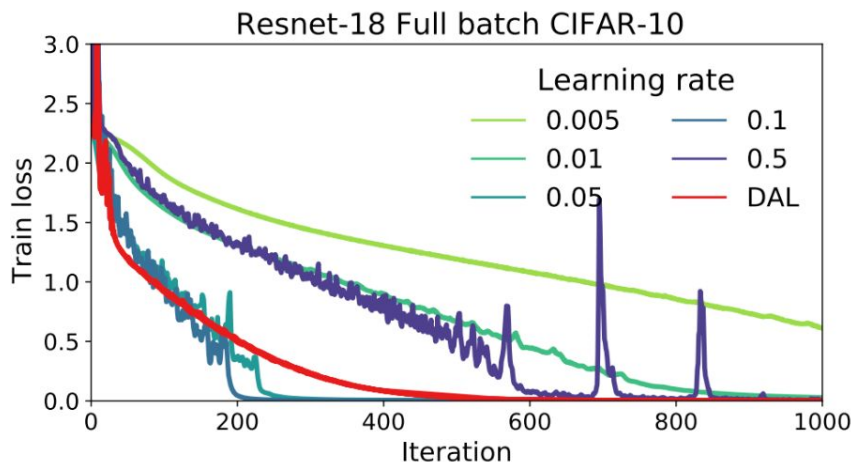


# Understanding discretisation drift



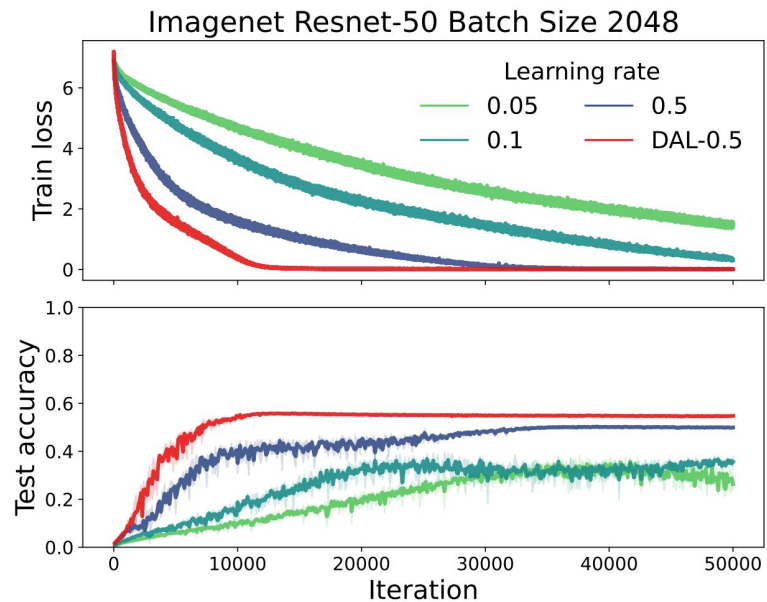
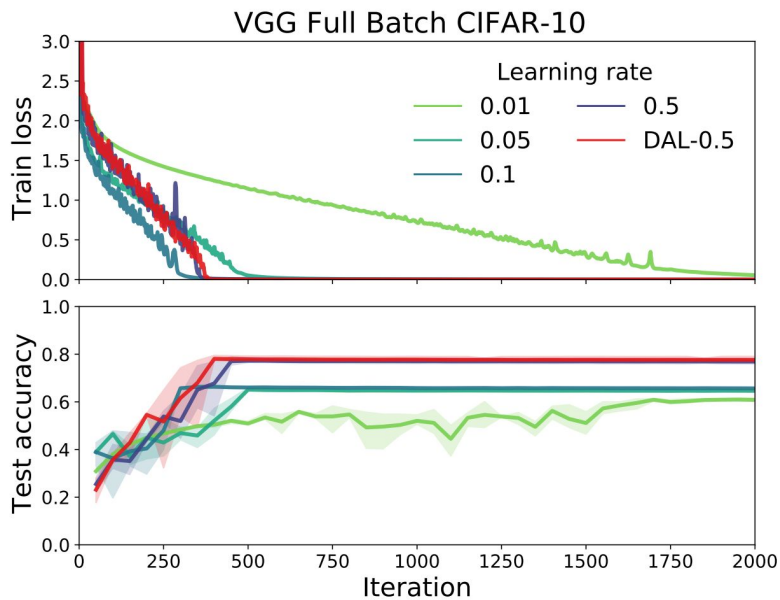
# Stabilising training with Drift Adjusted Learning (DAL)

$$h(\theta) = \frac{\|\nabla_{\theta}^2 E \nabla_{\theta} E\|}{\|\nabla_{\theta} E\|^2} = \frac{\|\nabla_{\theta}^2 E \hat{\mathbf{g}}(\theta)\|}{\|\nabla_{\theta} E\|^2}$$

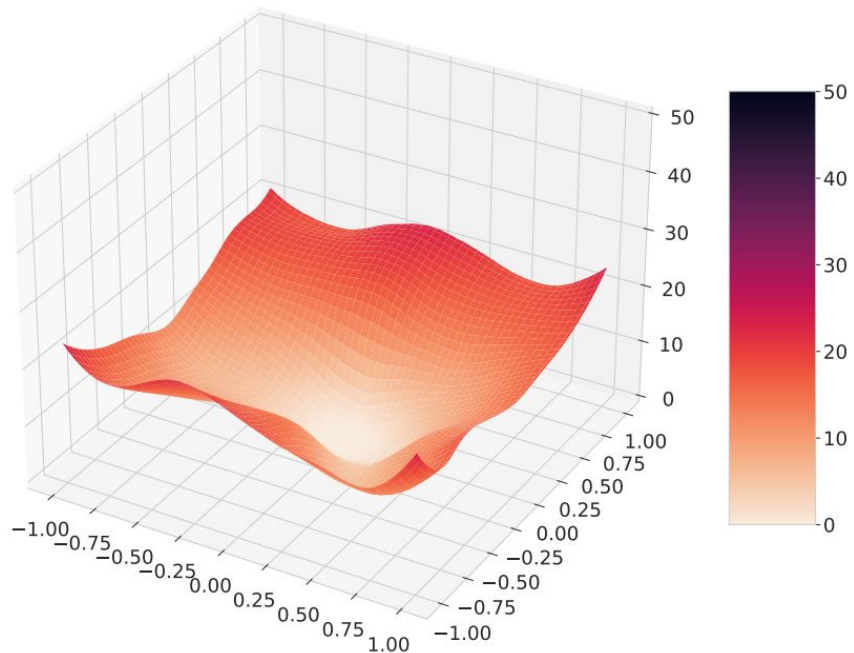


# DAL-p: controlling generalisation versus stability

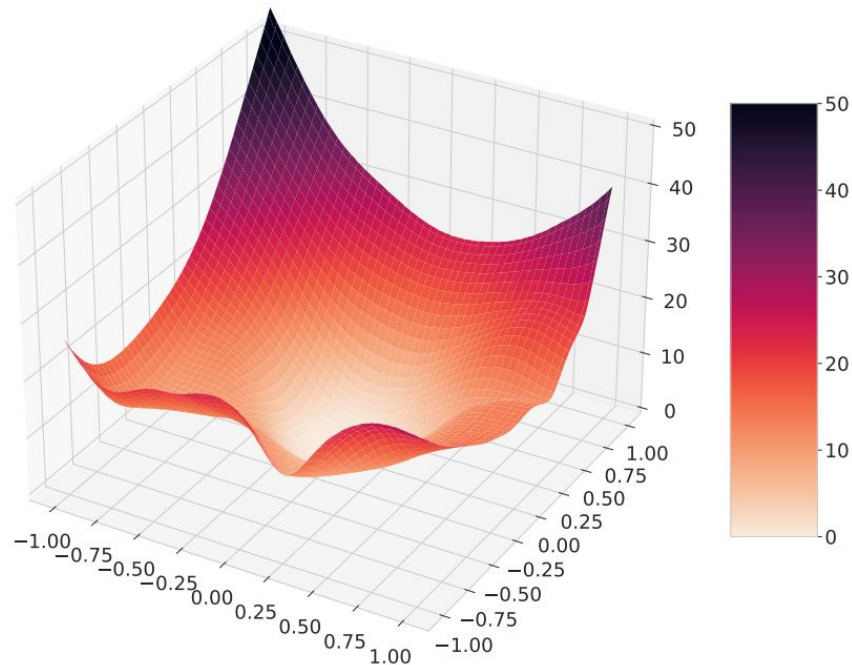
$$h_p(\boldsymbol{\theta}) = \frac{2}{(\|\nabla_{\boldsymbol{\theta}}^2 E \hat{\mathbf{g}}(\boldsymbol{\theta})\|)^p}$$



## DAL-p: also leads to flat minima



(a) DAL



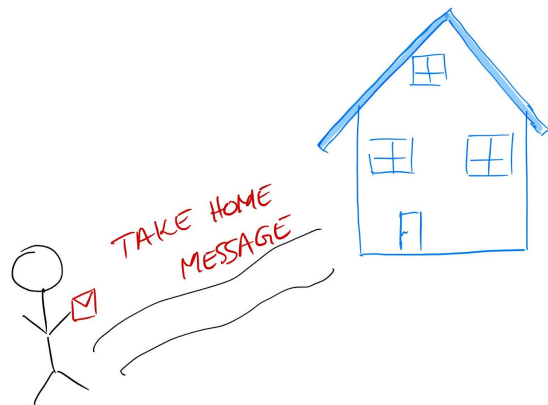
(b) SGD

**At convergence, DAL leads to flatter minima than SGD.**



# Summary

- We need to operate in complex space to handle instabilities observed with gradient descent.
- Model continuous-time models of gradient descent can be a useful tool to understand and improve optimisation.



**How about stochasticity?**



## How about stochasticity?

$$\boldsymbol{\theta}_{t+\mu} = \boldsymbol{\theta}_{t+\mu-1} - \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t+\mu-1}; \mathbf{X}^{t+\mu}), \quad \mu \in \{0, \dots, n-1\},$$

# Implicit regularisation in stochastic gradient descent: from single-objective to two-player games

Rosca et Deisenroth, ICML workshop 2023

## Stochasticity so far

All we have said thus far also applies to SGD, but only to 1 step of SGD.

How can we model what happens when we take multiple SGD steps?

Are there implicit regularisation effects specific to SGD?

## Stochasticity: expectation over one epoch (Smith et al)

If one takes expectations over the shufflings in one epoch, Smith et al find the implicit regulariser:

$$\mathbb{E}_\sigma [E_{sgd}(\boldsymbol{\theta})] = E(\boldsymbol{\theta}; \{\mathbf{X}^t, \dots, \mathbf{X}^{t+n-1}\}) + \frac{h}{4n} \sum_{k=0}^{n-1} \left\| \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}; \mathbf{X}^{t+k}) \right\|^2 .$$



The effect of implicit regularization is stronger for smaller batch sizes.

*On the Origin of Implicit Regularization in Stochastic Gradient Descent, Smith et al, ICLR 2021*

## Recent work: no need to work in expectation

If we allow for a dependence on initial parameters, we can write (for any  $n$ ):

$$\begin{aligned} \tilde{E}(\boldsymbol{\theta}) = & E(\boldsymbol{\theta}; \{\mathbf{X}^t, \dots, \mathbf{X}^{t+n-1}\}) + \frac{nh}{4} \underbrace{\left\| \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}; \{\mathbf{X}^t, \dots, \mathbf{X}^{t+n-1}\}) \right\|^2}_{\text{full batch norm regularisation}} \\ & - \frac{h}{n} \sum_{\mu=1}^{n-1} \underbrace{\left[ \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}; \mathbf{X}^{t+\mu})^T \left( \sum_{\tau=0}^{\mu-1} \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1}; \mathbf{X}^{t+\tau}) \right) \right]}_{\text{mini-batch gradient alignment}}. \end{aligned}$$

## Recent work: no need to work in expectation

$n=2$

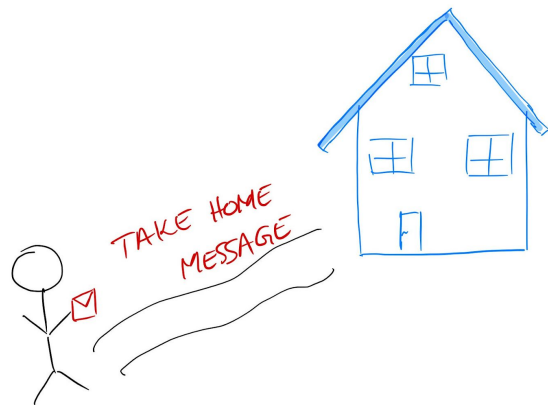
$$\begin{aligned} \tilde{E}(\boldsymbol{\theta}) = & E(\boldsymbol{\theta}; \{\mathbf{X}^t, \mathbf{X}^{t+1}\}) + \frac{h}{2} \underbrace{\|\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}; \{\mathbf{X}^t, \mathbf{X}^{t+1}\})\|^2}_{\text{full batch norm regularisation}} \\ & - \frac{h}{2} \underbrace{\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}; \mathbf{X}^{t+1})^T (\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1}; \mathbf{X}^t))}_{\text{mini-batch gradient alignment}}. \end{aligned}$$

## Recent work: no need to work in expectation

**Remark 5** *When taking a second SGD step, there is an implicit regularisation term maximising the dot product between the gradient at the current step and the gradient at the previous step:  $\nabla_{\theta} E^T \nabla_{\theta} E(\theta_{t-1})$ . This can be achieved by aligning the direction of the gradients between the two iterations or increasing the gradient norm.*

# Summary

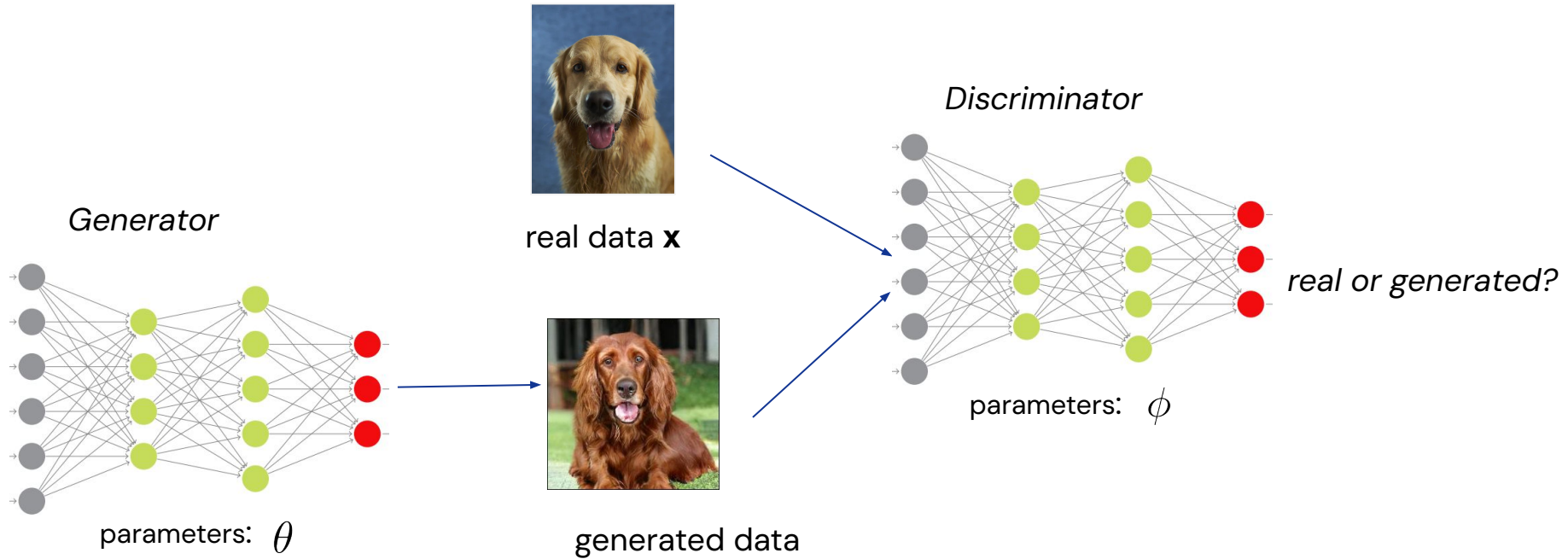
- We can model the implicit regularisation effects of more than 1 SGD step
- There is an alignment pressure between gradients at consecutive iterations
  - empirical work needed to show its effects



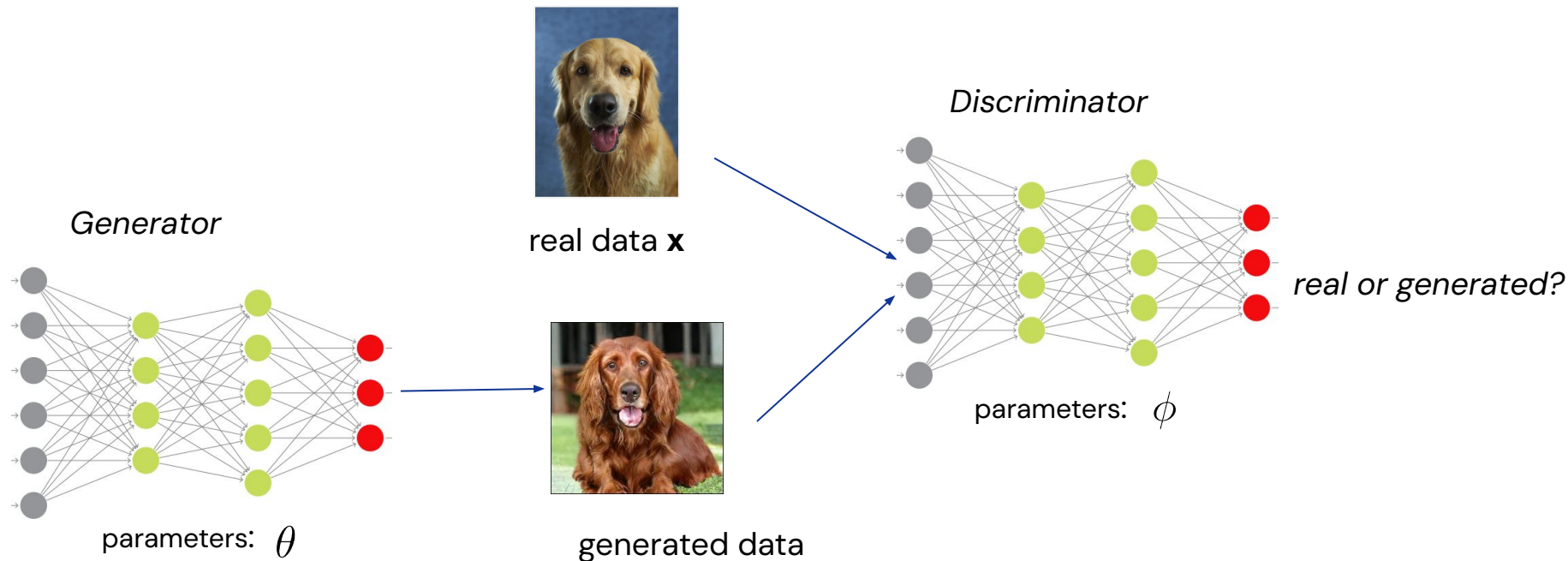


**How about two-player games?**

# Generative adversarial networks



# Generative adversarial networks



$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

# Discretization Drift in Two-Player Games

Rosca et al, ICML 2021

## Two-player games continuous dynamics

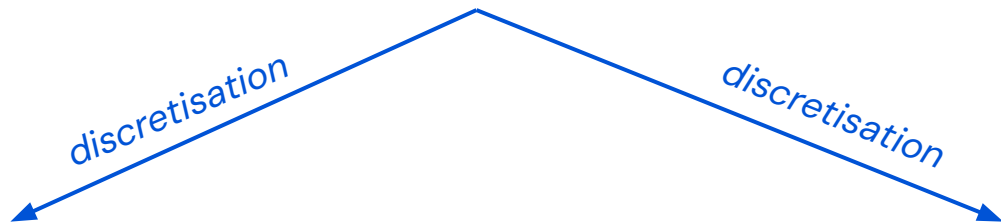
$$\dot{\phi} = f(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta)$$

## Two-player games

$$\dot{\phi} = f(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta)$$



Simultaneous Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_{t-1}, \theta_{t-1})$$

Alternating Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_t, \theta_{t-1})$$

## Differentiable two-player games

$$f = -\nabla_{\phi} L_f$$

$$g = -\nabla_{\theta} L_g$$

Gradient descent!

# Two-player games

Continuous dynamics

$$\dot{\phi} = f(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta)$$

Numerical integration



Discrete dynamics

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_{t-1}, \theta_{t-1})$$

Simultaneous Euler updates



# Two-player games

Continuous dynamics

$$\dot{\phi} = f(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta)$$

Numerical integration



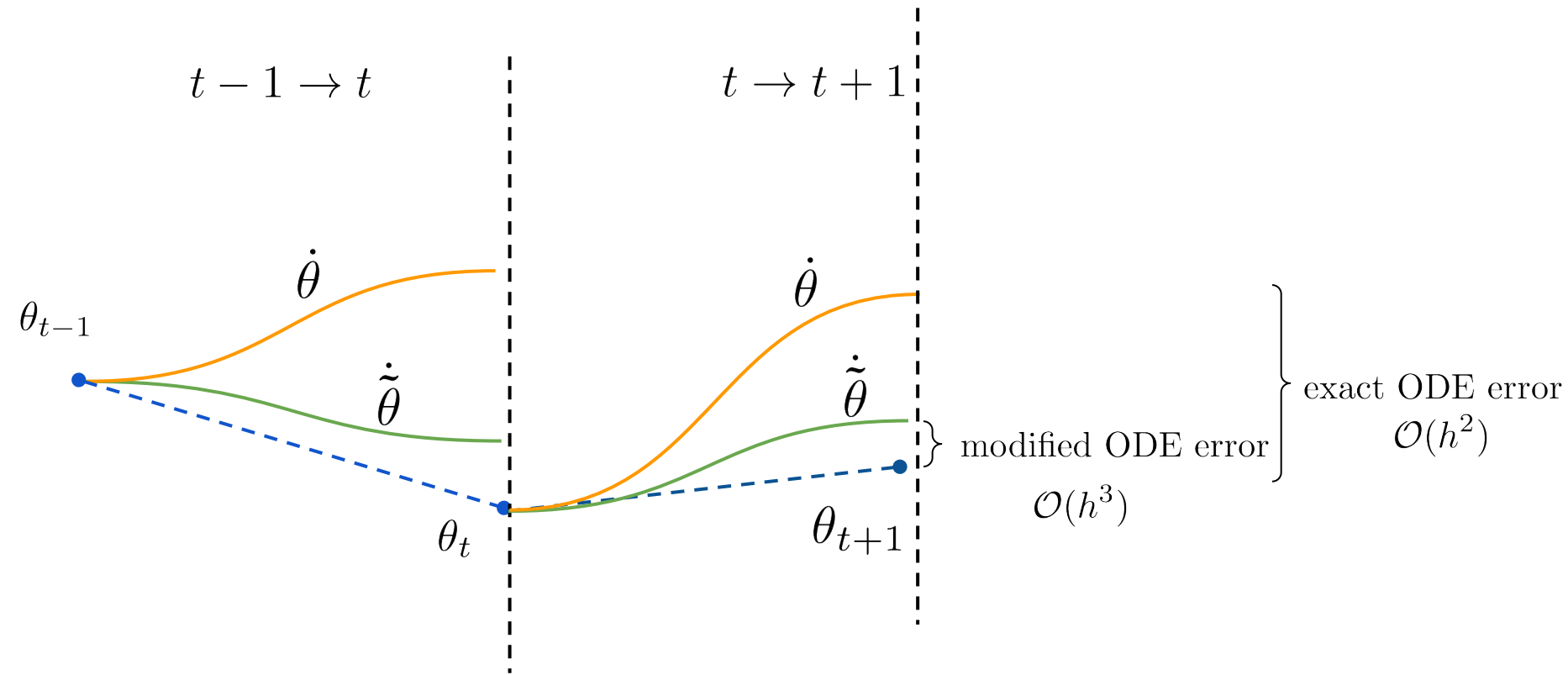
Discrete dynamics

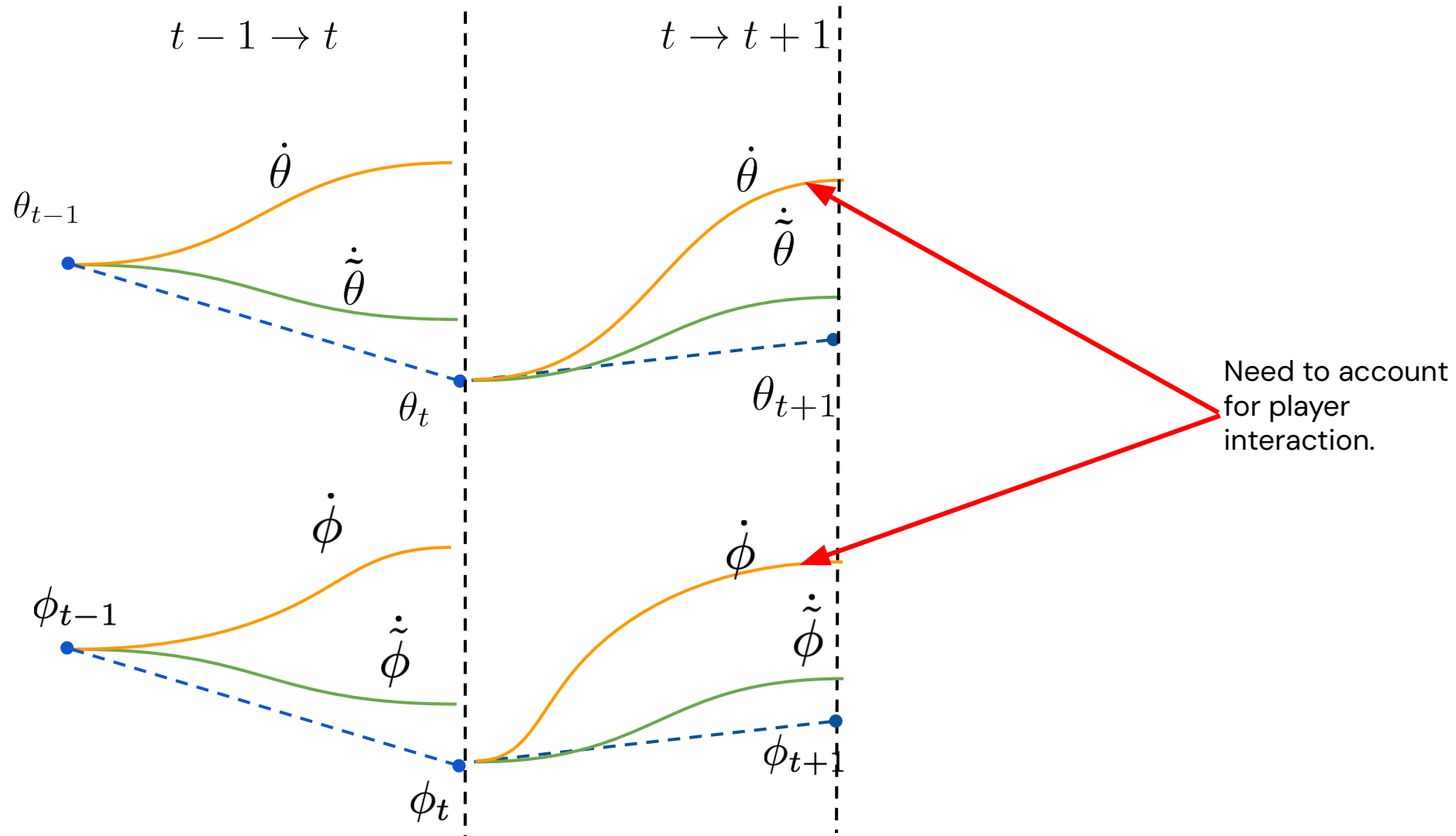
$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_t, \theta_{t-1})$$

Alternating Euler updates

# The approach





# Modified ODEs for two-player games

Discrete dynamics

Simultaneous Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_{t-1}, \theta_{t-1})$$

Backward error analysis



Continuous dynamics

$$\dot{\phi} = f(\phi, \theta) + \alpha h f_1(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta) + \lambda h g_1(\phi, \theta)$$

# Modified ODEs for two-player games

Discrete dynamics

Simultaneous Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_{t-1}, \theta_{t-1})$$

Backward error analysis



Continuous dynamics

$$\dot{\phi} = f(\phi, \theta) + \alpha h f_1(\phi, \theta)$$

$$\dot{\theta} = g(\phi, \theta) + \lambda h g_1(\phi, \theta)$$

Goal: find  $f_1$  and  $g_1$

# Modified ODEs for two-player games

Discrete dynamics

Simultaneous Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_{t-1}, \theta_{t-1})$$

Backward error analysis



Continuous dynamics

$$\begin{aligned}\dot{\tilde{\phi}} &= f(\phi, \theta) - \frac{\alpha h}{2} (\underbrace{f \nabla_{\phi} f}_{\text{cyan}} + \underbrace{g \nabla_{\theta} f}_{\text{orange}}) \\ \dot{\tilde{\theta}} &= g(\phi, \theta) - \frac{\lambda h}{2} (\underbrace{g \nabla_{\theta} g}_{\text{cyan}} + \underbrace{f \nabla_{\phi} g}_{\text{orange}})\end{aligned}$$

**Self** and **interaction terms.**

# Modified ODEs for two-player games

Discrete dynamics

Alternating Euler updates

$$\phi_t = \phi_{t-1} + \alpha h f(\phi_{t-1}, \theta_{t-1})$$

$$\theta_t = \theta_{t-1} + \lambda h g(\phi_t, \theta_{t-1})$$

Backward error analysis

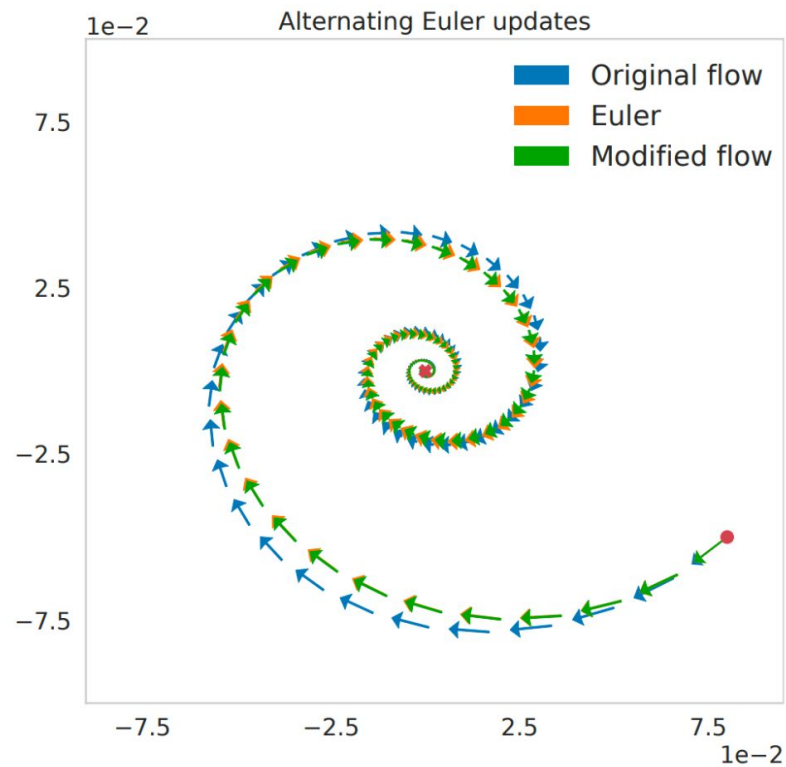
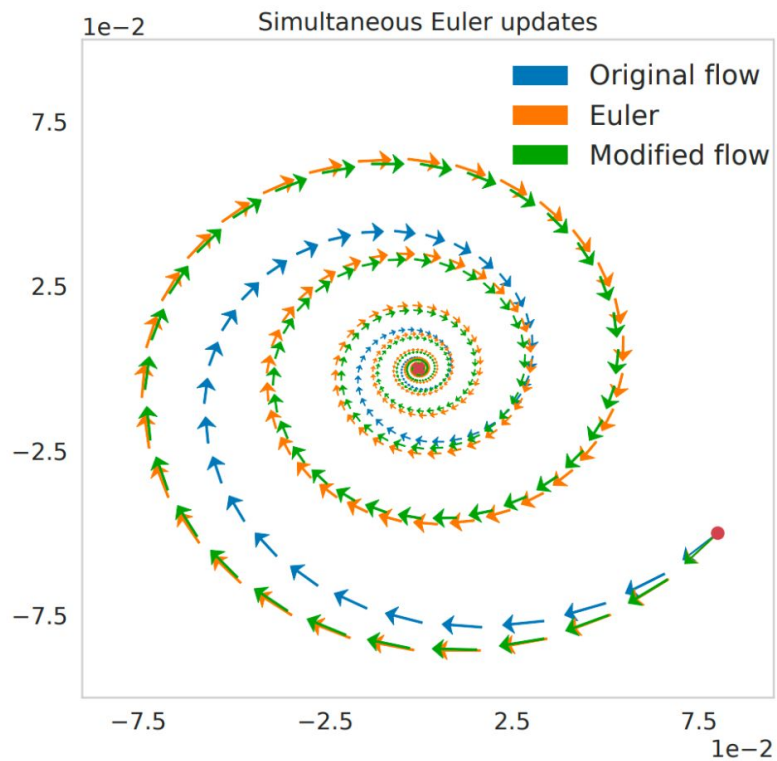


Continuous dynamics

$$\begin{aligned}\dot{\phi} &= f(\phi, \theta) - \frac{\alpha h}{2} ( \underbrace{f \nabla_{\phi} f}_{\text{Self}} + \underbrace{g \nabla_{\theta} f}_{\text{Interaction}} ) \\ \dot{\theta} &= g(\phi, \theta) - \frac{\lambda h}{2} \left( \underbrace{g \nabla_{\theta} g}_{\text{Self}} + \underbrace{\left(1 - \frac{2\alpha}{\lambda}\right) f \nabla_{\phi} g}_{\text{Interaction}} \right)\end{aligned}$$

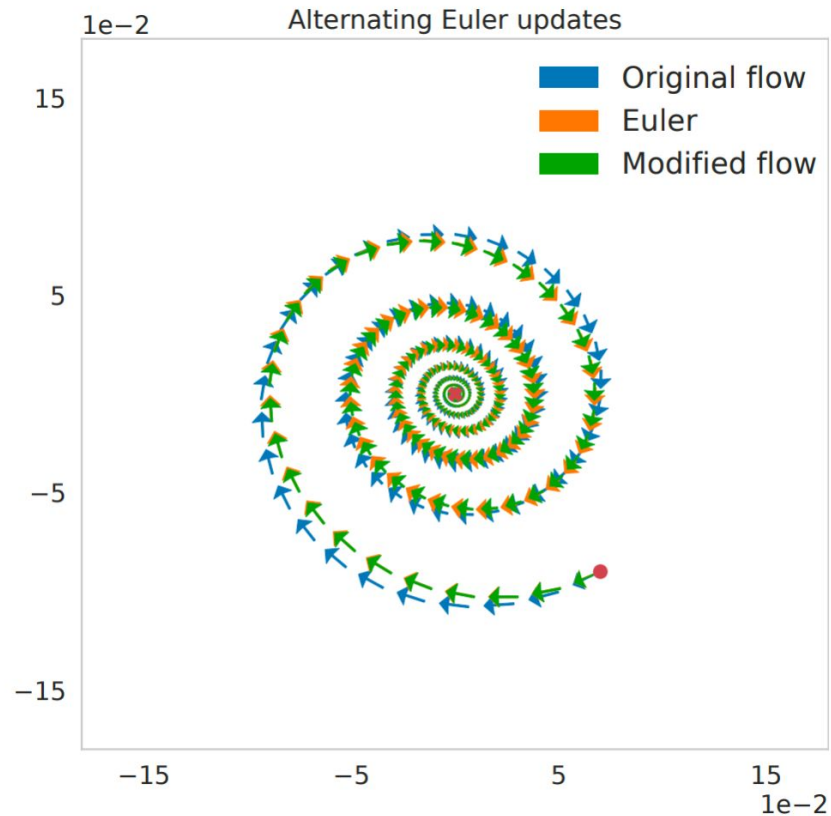
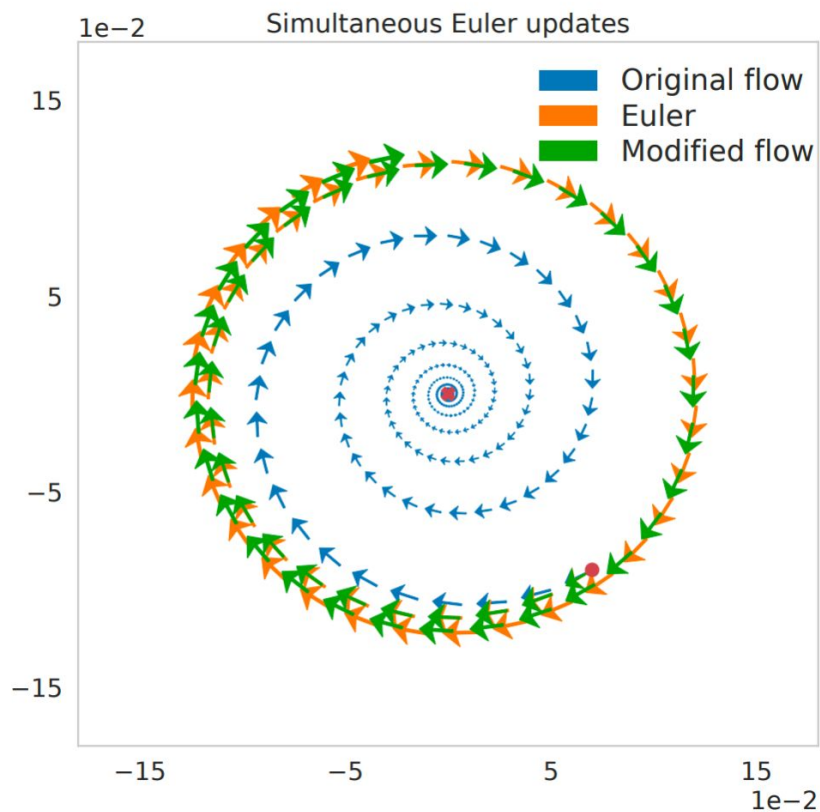
**Self** and **interaction terms.**

# Building intuition





# Building intuition



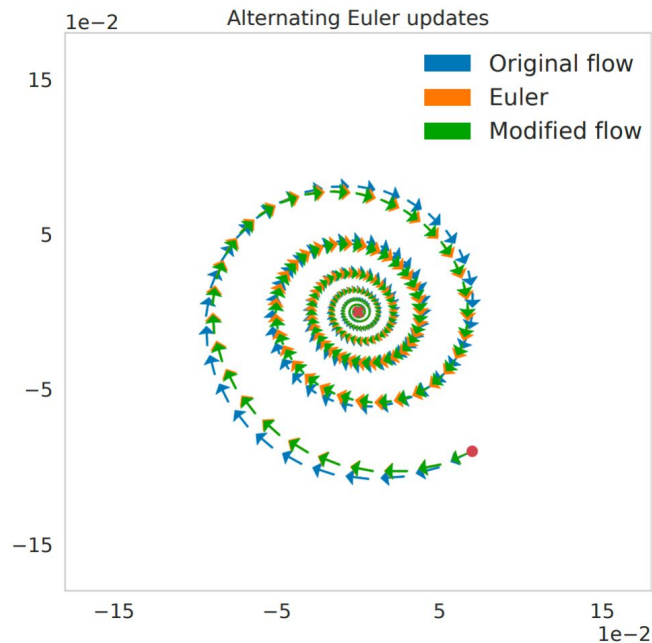
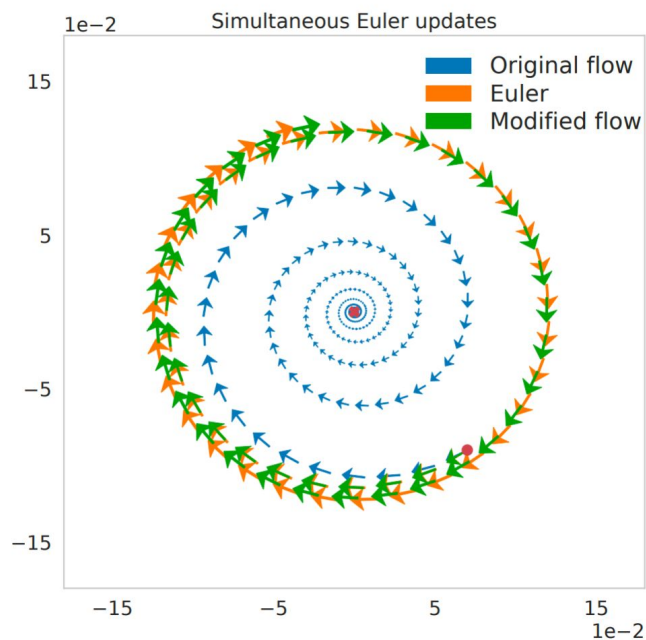
## Modified ODEs as tools for stability analysis

$$\tilde{J} = \begin{bmatrix} \nabla_{\phi} \tilde{f} & \nabla_{\theta} \tilde{f} \\ \nabla_{\phi} \tilde{g} & \nabla_{\theta} \tilde{g} \end{bmatrix} = J - \frac{h}{2} \Delta$$

Jacobian of the original game

$$\Delta = \begin{bmatrix} \alpha(\nabla_{\phi} f)^2 + \alpha \nabla_{\phi} g \nabla_{\theta} f & \alpha \nabla_{\theta} f \nabla_{\phi} f + \alpha \nabla_{\theta} g \nabla_{\theta} f \\ \lambda \nabla_{\phi} g \nabla_{\theta} g + \lambda \nabla_{\phi} f \nabla_{\phi} g & \lambda (\nabla_{\theta} g)^2 + \lambda \nabla_{\theta} f \nabla_{\phi} g \end{bmatrix}$$

Having access different ODEs for simultaneous and alternating updates, we can now analyse the stability of simultaneous and alternating gradient descent separately.



## Zero-sum games

$$L_f = -E$$

$$L_g = E$$

$$f = \nabla_{\phi} E$$

$$g = -\nabla_{\theta} E$$

## Simultaneous gradient descent in zero-sum games

$$\tilde{L}_\phi = -E + \frac{\alpha h}{4} \|\nabla_\phi E\|^2 - \frac{\alpha h}{4} \|\nabla_\theta E\|^2$$

$$\tilde{L}_\theta = E + \frac{\lambda h}{4} \|\nabla_\theta E\|^2 - \frac{\lambda h}{4} \|\nabla_\phi E\|^2$$

The *interaction terms* maximize the gradient norm of the other player.

## Simultaneous gradient descent in zero-sum games

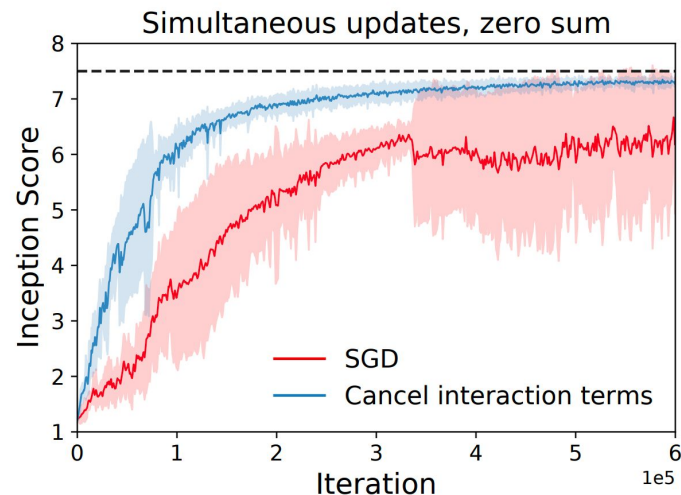
$$\tilde{L}_\phi = -E + \frac{\alpha h}{4} \|\nabla_\phi E\|^2 - \frac{\lambda h}{4} \|\nabla_\theta E\|^2$$

$$\tilde{L}_\theta = E + \frac{\lambda h}{4} \|\nabla_\theta E\|^2 - \frac{\alpha h}{4} \|\nabla_\phi E\|^2$$

# Simultaneous gradient descent in zero-sum games

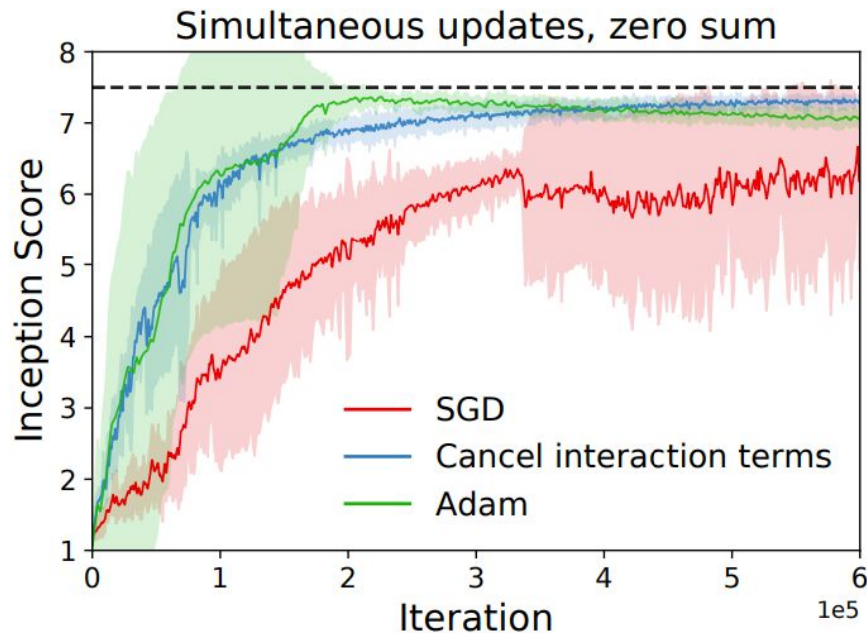
$$\tilde{L}_\phi = -E + \frac{\alpha h}{4} \|\nabla_\phi E\|^2 - \frac{\lambda h}{4} \|\nabla_\theta E\|^2$$
$$\tilde{L}_\theta = E + \frac{\lambda h}{4} \|\nabla_\theta E\|^2 - \frac{\alpha h}{4} \|\nabla_\phi E\|^2$$

GAN training  
Higher is better



Explicit regularization cancelling the *interaction terms* improves performance.

# Simultaneous gradient descent in zero-sum games



*GAN training  
Higher is better*

Using explicit regularization SGD reaches the same peak performance as Adam.



## Alternating gradient descent in zero sum games

$$\tilde{L}_\phi = -E + \frac{\alpha h}{4} \|\nabla_\phi E\|^2 - \frac{\alpha h}{4} \|\nabla_\theta E\|^2$$

$$\tilde{L}_\theta = E + \frac{\lambda h}{4} \|\nabla_\theta E\|^2 - \frac{\lambda h}{4} \left(1 - \frac{2\alpha}{\lambda}\right) \|\nabla_\phi E\|^2$$

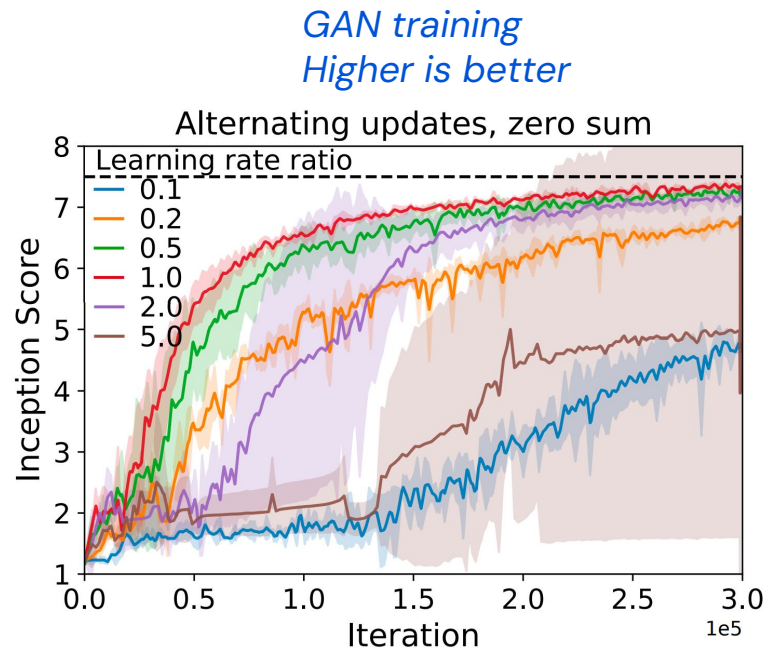


learning rate ratio

# Alternating gradient descent in zero sum games

We can predict which learning rate ratios will perform best!

The sign of the interaction term for the second player provides useful practical information.



# Comparison with other methods

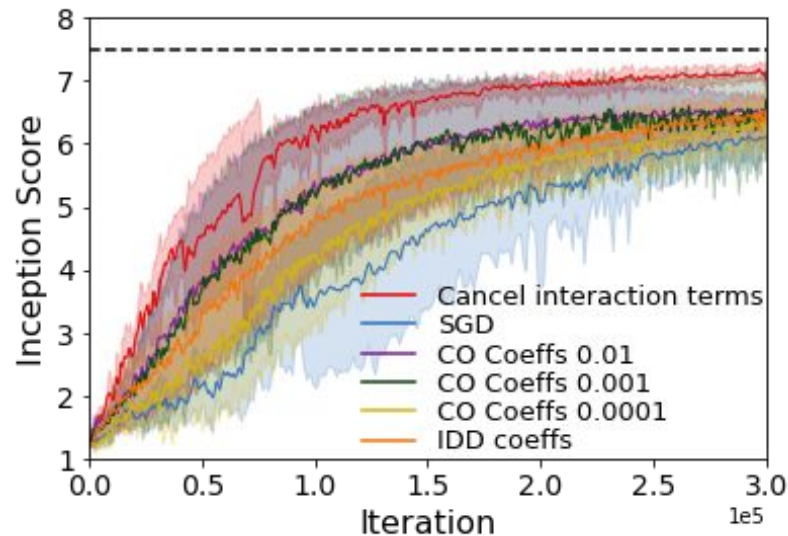
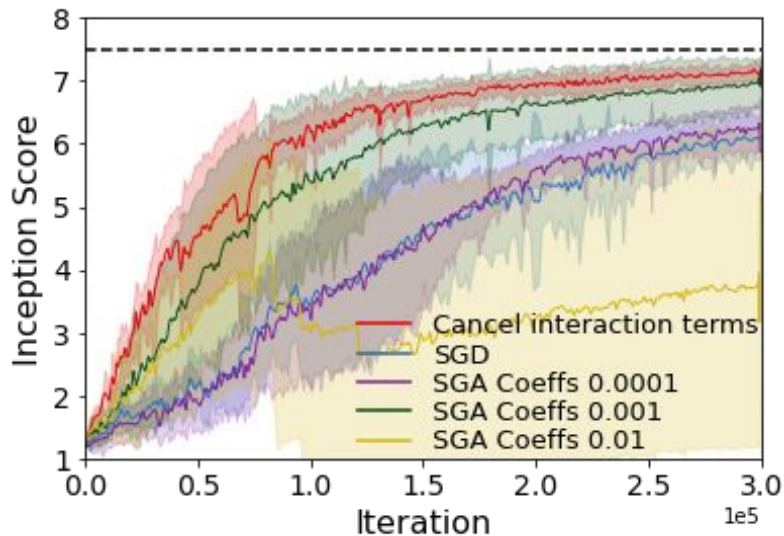
	Explicit	First player ( $\phi$ )	Second player ( $\theta$ )
DD (S)	✗	$\frac{\alpha h}{4} \ \nabla_{\phi} E\ ^2 - \frac{\alpha h}{4} \ \nabla_{\theta} E\ ^2$	$-\frac{\lambda h}{4} \ \nabla_{\phi} E\ ^2 + \frac{\lambda h}{4} \ \nabla_{\theta} E\ ^2$
DD (A)	✗	$\frac{\alpha h}{4m} \ \nabla_{\phi} E\ ^2 - \frac{\alpha h}{4} \ \nabla_{\theta} E\ ^2$	$\frac{(2\alpha - \lambda)h}{4} \ \nabla_{\phi} E\ ^2 + \frac{\lambda h}{4k} \ \nabla_{\theta} E\ ^2$
Cancel DD interaction terms (S)	✓	$\frac{\alpha h}{4} \ \nabla_{\theta} E\ ^2$	$\frac{\lambda h}{4} \ \nabla_{\phi} E\ ^2$
Cancel DD interaction terms (A)	✓	$\frac{\alpha h}{4} \ \nabla_{\theta} E\ ^2$	$-\frac{(2\alpha - \lambda)h}{4} \ \nabla_{\phi} E\ ^2$
SGA (S)	✓	$\frac{1}{2} \ \nabla_{\theta} E\ ^2$	$\frac{1}{2} \ \nabla_{\phi} E\ ^2$
Consensus optimization (S)	✓	$\eta \ \nabla_{\phi} E\ ^2 + \eta \ \nabla_{\theta} E\ ^2$	$\eta \ \nabla_{\phi} E\ ^2 + \eta \ \nabla_{\theta} E\ ^2$
Locally stable GAN (S)	✓	✗	$\eta \ \nabla_{\phi} E\ ^2$
ODE-GAN (S)	✓	$\eta \ \nabla_{\theta} E\ ^2$	✗

Table 1. Comparing DD with explicit regularization methods in zero-sum games. SGA (without alignment, see the Supplementary Material and Balduzzi et al. 2018), Consensus Optimization (Mescheder et al., 2017), Locally Stable GAN (Nagarajan & Kolter, 2017), ODE-GAN (Qin et al., 2020). We assume a  $\min_{\theta} \min_{\phi}$  game with learning rates  $\alpha h$  and  $\lambda h$  and number of updates  $m$  and  $k$ , respectively. S and A denote simultaneous and alternating updates.



# Empirical comparison

GAN training  
Higher is better



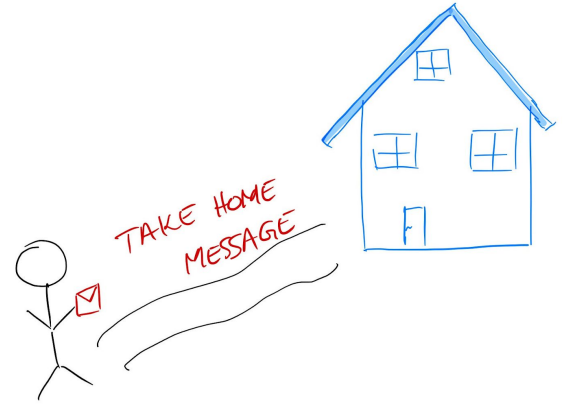
## Recent work: no need to only do zero-sum games

$$\tilde{E}_{\phi,t} = E_{\phi} + h \left( \underbrace{\frac{1}{2} \|\nabla_{\phi} E_{\phi}\|^2}_{\text{self term}} + \underbrace{\frac{1}{2} \nabla_{\theta} E_{\phi}^T \nabla_{\theta} E_{\theta}(\phi_{t-1}, \theta_{t-1})}_{\text{interaction term}} \right)$$
$$\tilde{E}_{\theta,t} = E_{\theta} + h \left( \underbrace{\frac{1}{2} \|\nabla_{\theta} E_{\theta}\|^2}_{\text{self term}} + \underbrace{\frac{1}{2} \nabla_{\phi} E_{\theta}^T \nabla_{\phi} E_{\phi}(\phi_{t-1}, \theta_{t-1})}_{\text{interaction term}} \right).$$



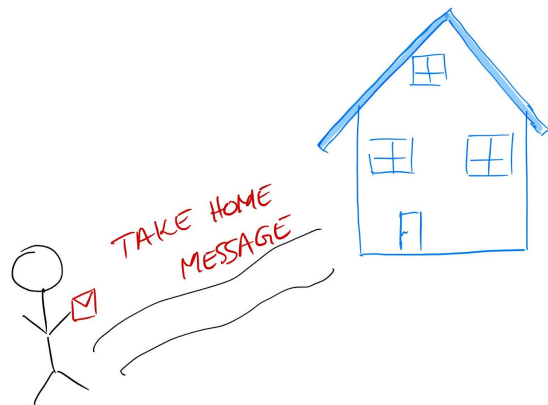
# Summary

- Discretisation drift has two terms in games: the self term and interaction term
- The interaction term can destabilise the game, and using explicit regularisation to cancel it can improve stability and performance



# Talk summary

- Continuous time approaches are a useful tool to understand optimisation
- We can use them to find implicit regularisers as well as stabilising training
- Different dynamics between one objective and two-player games



**Thank you!**