DeepMind

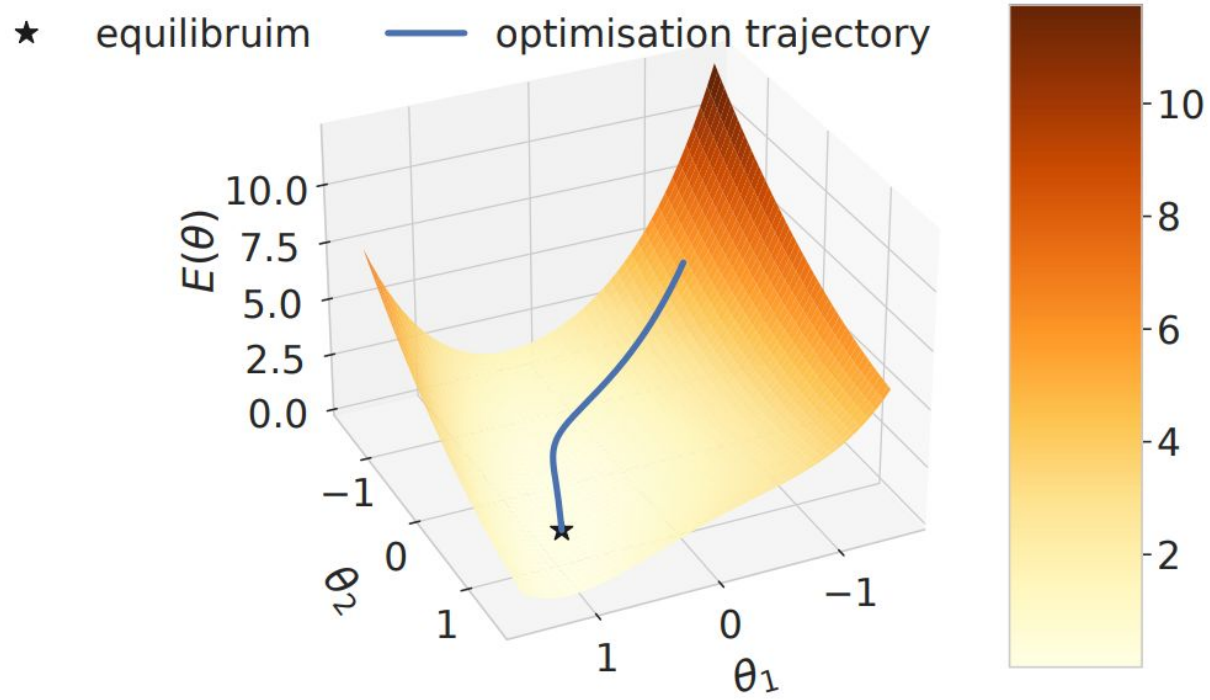
Optimisation in deep learning

Mihaela Rosca
9/2024

Disclaimer

This is a very broad overview of the field, there are many other interesting works that cannot be covered due to time constraints. If I missed anything, this is not a statement about the value of that respective work.

There are slides with additional references at the end of the talk.



Mihaela Rosca, 2024

Gradient descent:

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

parameter dimension: *D*
h denotes the learning rate throughout this talk.

A continuous time perspective

Negative gradient flow (NGF):

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

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

Continuous dynamics

Discrete dynamics

Euler integration

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

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

Gradient descent!

Continuous dynamics

Discrete dynamics

Euler integration

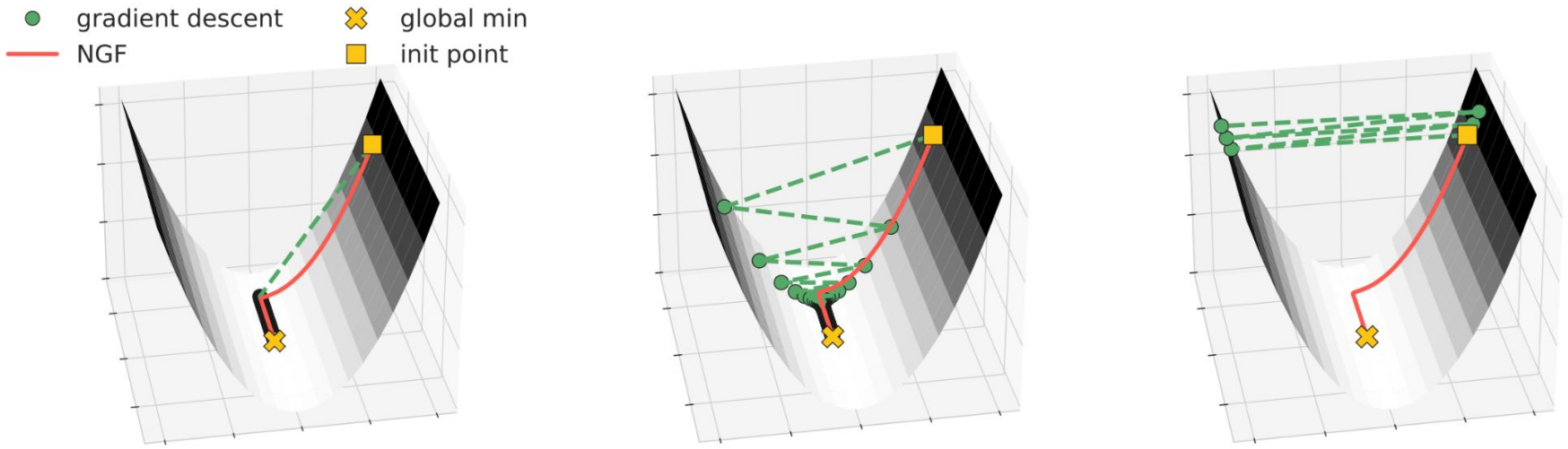
$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} E$$

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

$$\boldsymbol{\theta}(h; \boldsymbol{\theta}(0)) = \boldsymbol{\theta}(0) + \int_0^h \dot{\boldsymbol{\theta}}(t)dt = \boldsymbol{\theta}(0) - \int_0^h \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}(t))dt$$

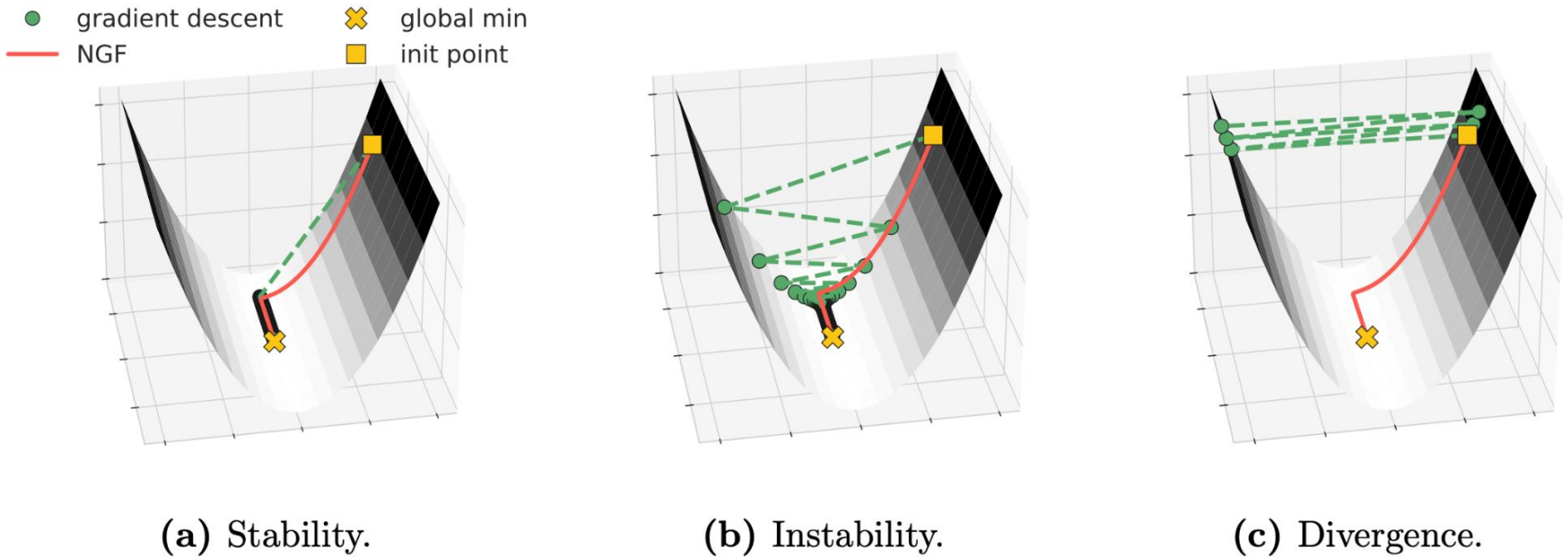
$$\approx \boldsymbol{\theta}(0) - \int_0^h \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}(0))dt = \boldsymbol{\theta}(0) - h\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}(0)).$$

Challenge: discretisation



(a) Stability. **(b)** Instability. **(c)** Divergence.

Challenge: discretisation



(a) Stability. **(b)** Instability. **(c)** Divergence.

For quadratic functions like this one, we know when instability/divergence happen. For NNs, it's more complex. More about this later!

Next step: momentum

Intuition:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \sum_{i=1}^{t} \beta^{i-1} \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-i}),$$

Next step: momentum

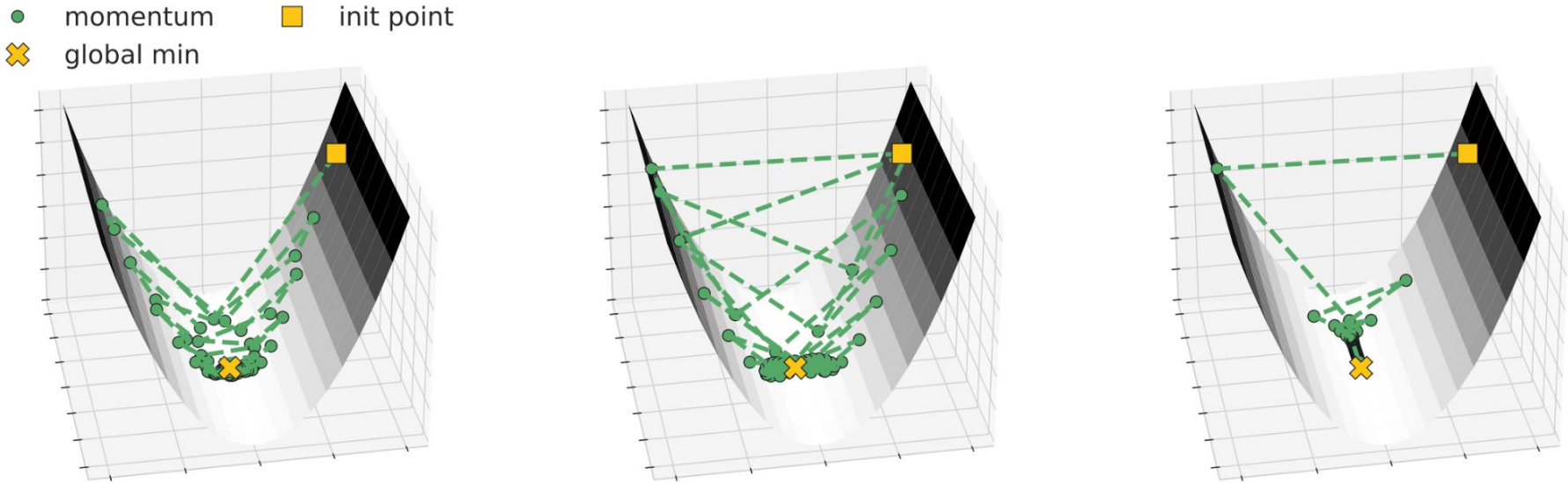
Intuition:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \sum_{i=1}^{t} \beta^{i-1} \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-i}).$$

We can rewrite this update as:

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} - h \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1}); \qquad \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{v}_t$$

Next step: momentum



(a) $\beta = 0.9$ and small h.

(b) $\beta = 0.9$ and large h.

(c) $\beta = 0.5$ and large h.

Next step: momentum as an ODE

$$\ddot{\boldsymbol{\theta}} + c_v \dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$$

Classical view: second order ODEs needed!

$$\dot{\boldsymbol{\theta}} = -\frac{1}{1-\beta} \nabla_{\boldsymbol{\theta}} E$$

Different discretisation: speeding up going into the direction of the negative gradient.

Next step: signs of the gradient is what matters

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

$$c_i > 0$$

Next step: signs of the gradient is what matters

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

Going to discrete space: Rprop

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

$$sign(\nabla_{\boldsymbol{\theta}_i} E) = \frac{\nabla_{\theta_i} E}{\sqrt{\left(\nabla_{\boldsymbol{\theta}_i} E\right)^2}}$$

Next step: Rmsprop

Rprop + moving averages:

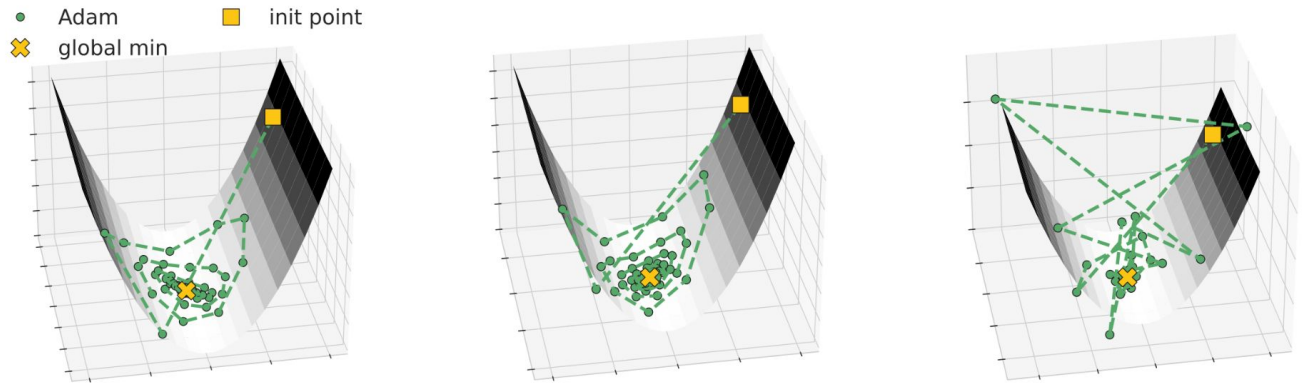
$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + (1-\beta) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})^2; \qquad \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \frac{\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})}{\sqrt{\mathbf{v}_t}}$$

Next step: Adam

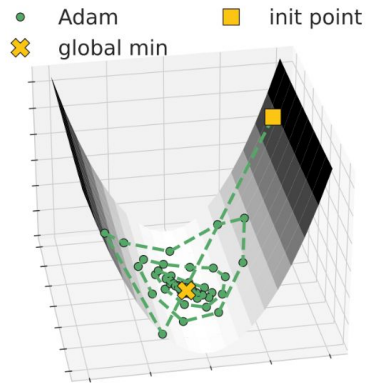
$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})$$

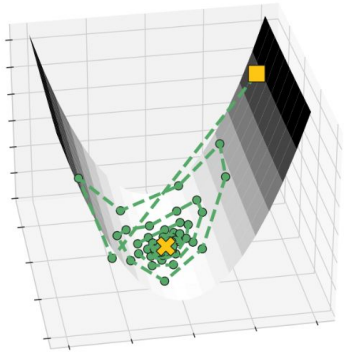
$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})^2$$

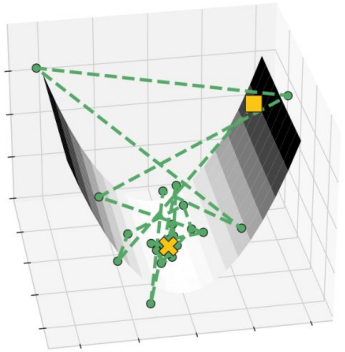
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h \frac{\frac{1}{1-\beta_1^t}\mathbf{m}_t}{\sqrt{\frac{1}{1-\beta_2^t}\mathbf{v}_t + \epsilon}},$$



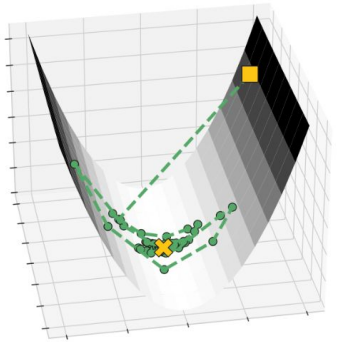
(a) $\beta_1 = 0.9$, small h, $\epsilon = 10^{-8}$. **(b)** $\beta_1 = 0.9$, large h, $\epsilon = 10^{-8}$. **(c)** $\beta_1 = 0.5$, large h, $\epsilon = 10^{-8}$.

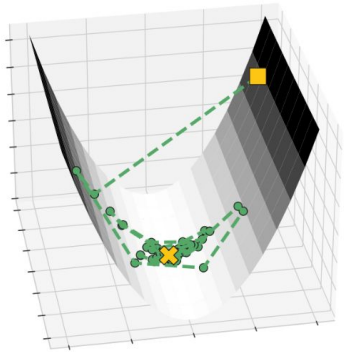


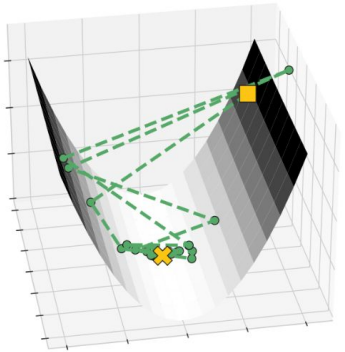




(a) $\beta_1 = 0.9$, small h, $\epsilon = 10^{-8}$. **(b)** $\beta_1 = 0.9$, large h, $\epsilon = 10^{-8}$. **(c)** $\beta_1 = 0.5$, large h, $\epsilon = 10^{-8}$.







(d) $\beta_1 = 0.9$, small h, $\epsilon = 10^{-2}$. **(e)** $\beta_1 = 0.9$, small h, $\epsilon = 10^{-2}$. **(f)** $\beta_1 = 0.5$, large h, $\epsilon = 10^{-2}$.

Summary

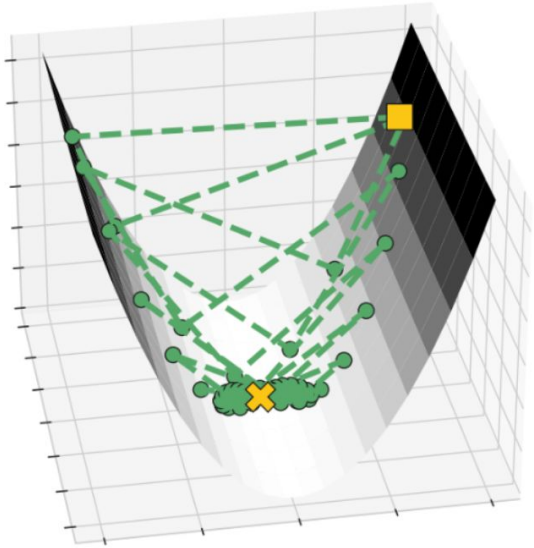
- It is useful to think what happens in continuous time to derive discrete time optimiser
- Sign of the gradient is that matters, but we often smooth over multiple steps
- Adam is a staple of deep learning optimisation, and hypers often ignored matter a lot

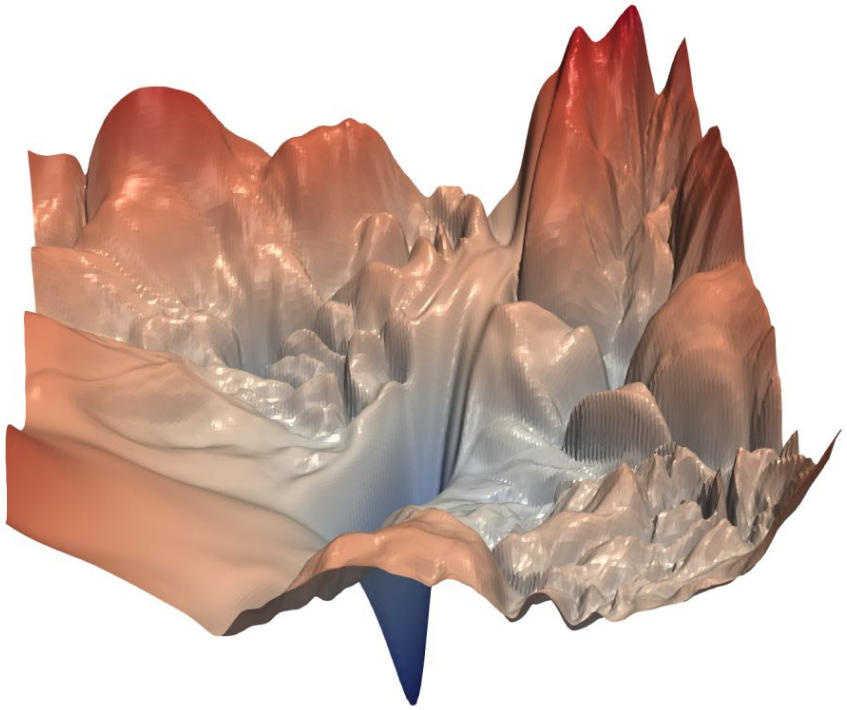
Deep learning optimisation

(with a continuous time flavour)

Mihaela Rosca, 2024

Deep learning optimisation





RHS image from: Visualizing the Loss Landscape of Neural Nets, Li et al, Neurips 2018

Deep learning optimisation

- High dimensional
- Need a lot of data: SGD
- Exhibits edge of stability phenomena
- Generalises
- Exhibits implicit regularisation phenomena

Deep learning optimisation

Questions to answer:

- why does it work so well?
- why are saddle points not a bigger issue?
- why do deep learning local optima generalise?

Deep learning optimisation

Concerns about saddle points:

- gradient descent iterations stop when the gradient is zero, but that does not mean we are at a local minima (where all Hessian eigenvalues are >=0).
- concern around high dimensionality: *"Intuitively, in high dimensions, the chance that all the directions around a critical point lead upward (positive curvature) is exponentially small w.r.t. the number of dimensions"* [Dauphin et al, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, 2014]
- Empirically seems less of a problem in practice
    - noise from stochastic gradient descent also seems to help

Generalisation

Why do deep learning models generalise?

The goal of optimisation: minimise the training loss.

The overarching goal of machine learning: generalise beyond training data.

Optimisers that are good at the former need not be good at the latter.

Flatness

Convergence to flat minima has been connected to generalisation, though this is debated. Some definitions of flatness:

- gradient norms
- Hessian eigenvalues or related measures

Note: due to reparametrization properties of NNs, we can construct equivalent neural networks with hugely varying sharpness properties[1].

[1] Sharp Minima Can Generalize For Deep Nets, Dhin et al, 2017

Flatness

Convergence to flat minima has been connected to generalisation, though this is debated. Some definitions of flatness:

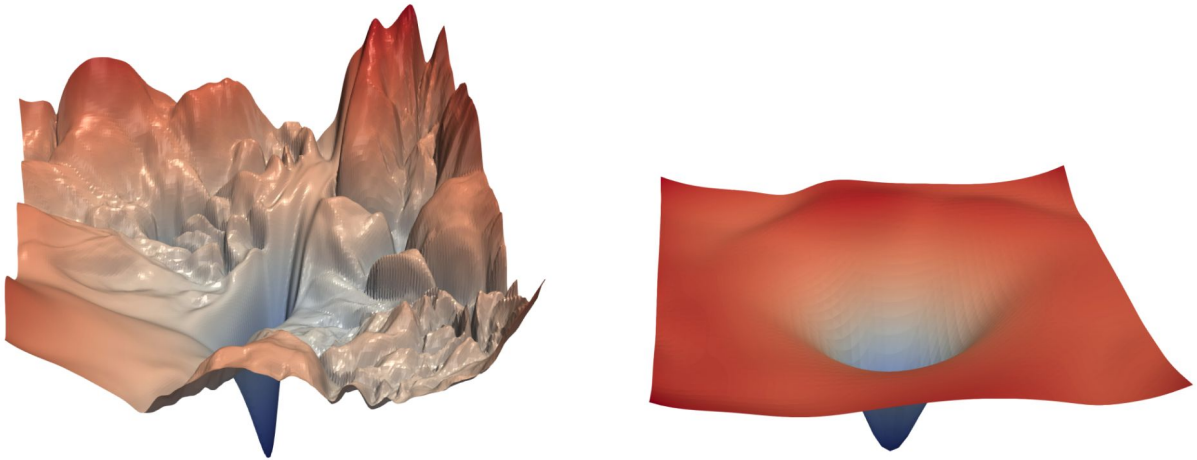
- gradient norms
- Hessian eigenvalues or related measures

Note: due to reparametrization properties of NNs, we can construct equivalent neural networks with hugely varying sharpness properties[1].

But, are these equally likely to be preferred by an optimisation procedure?

[1] Sharp Minima Can Generalize For Deep Nets, Dhin et al, 2017

Flatness changes with model architecture



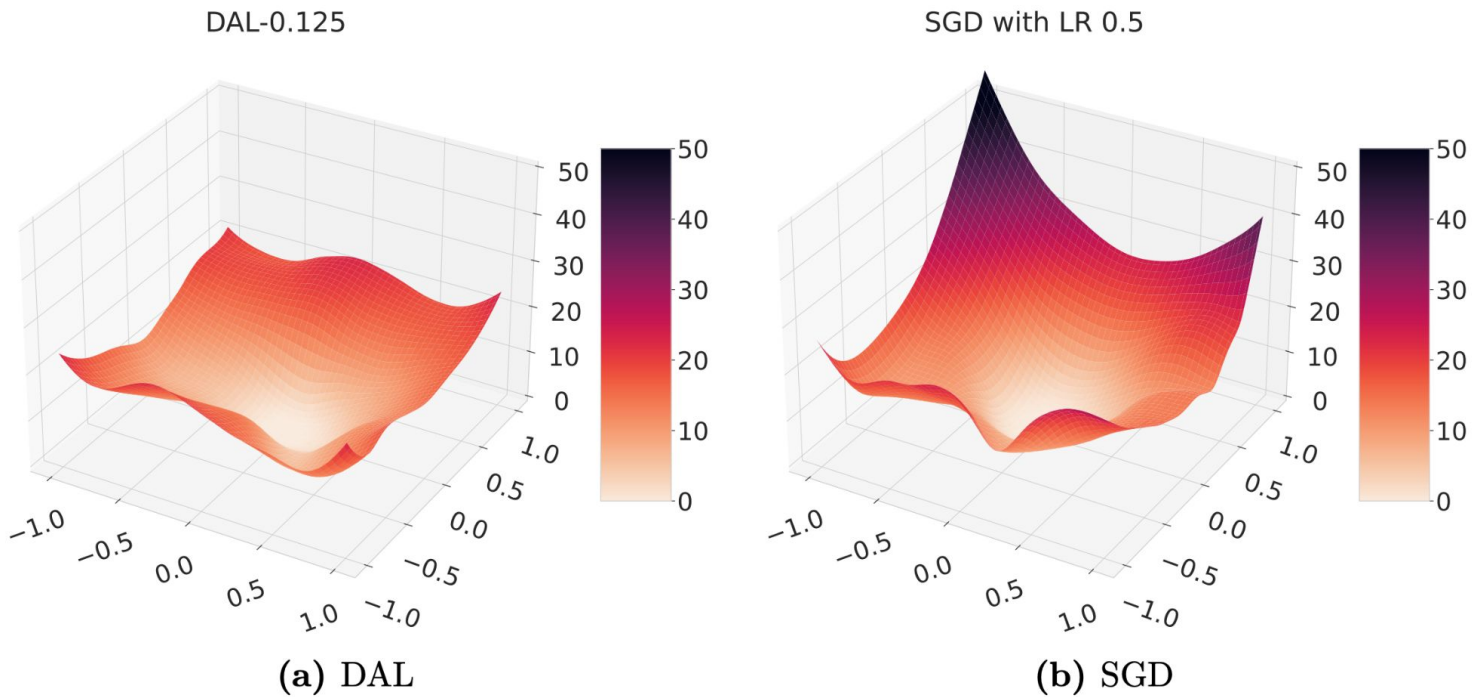
(a) without skip connections

(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Figure from: Visualizing the Loss Landscape of Neural Nets, Li et al, Neurips 2018

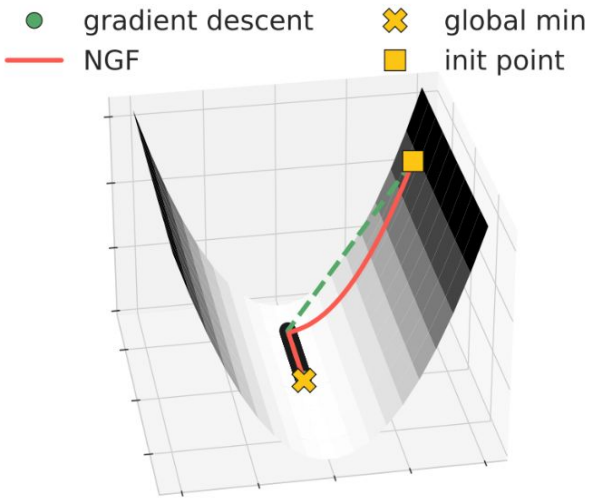
Flatness via optimisation changes



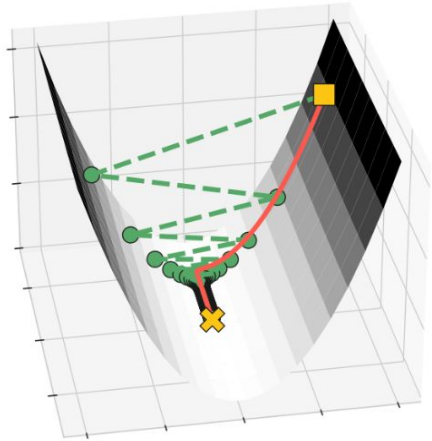
(a) DAL **(b)** SGD

Let’s start looking into some of these!

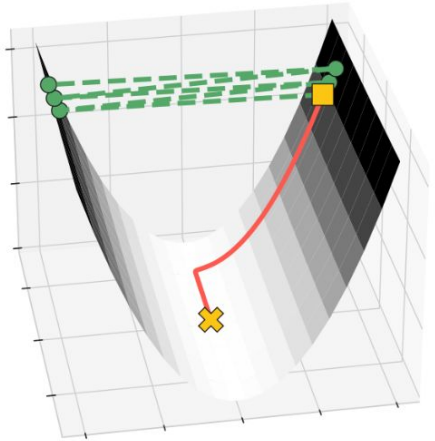
Remember this?



(a) Stability.

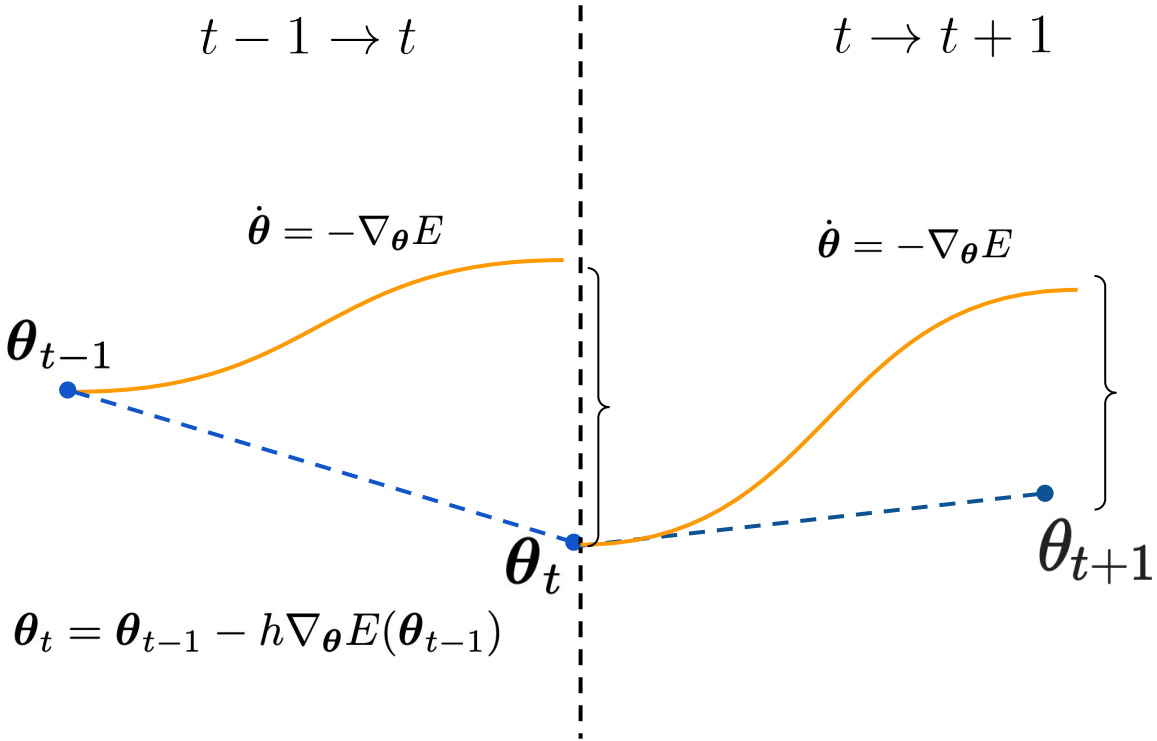


(b) Instability.



(c) Divergence.

Discretisation drift

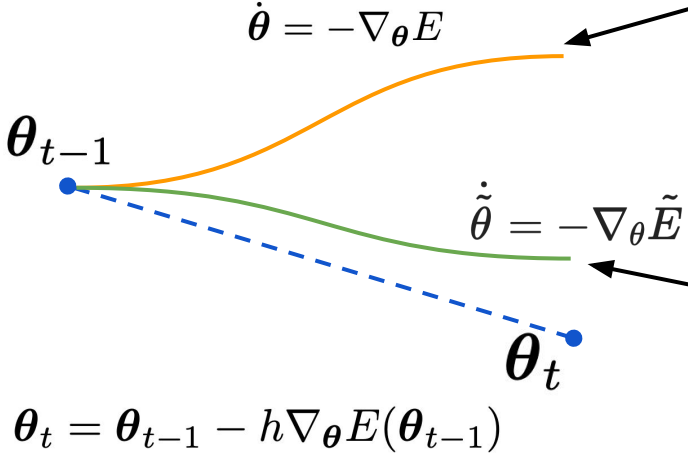


Discretization drift

$\mathcal{O}(h^2)$

Modified loss functions

$t-1 \rightarrow t$

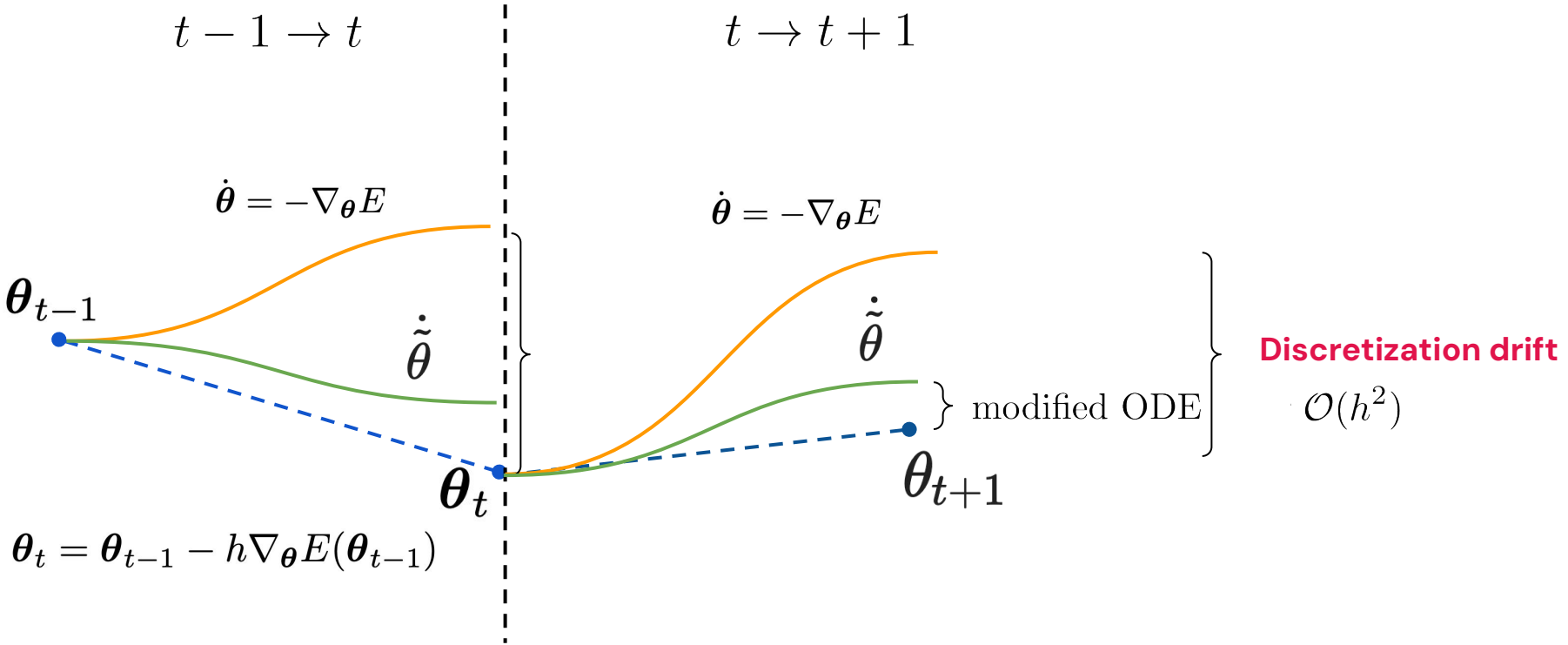


$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})$

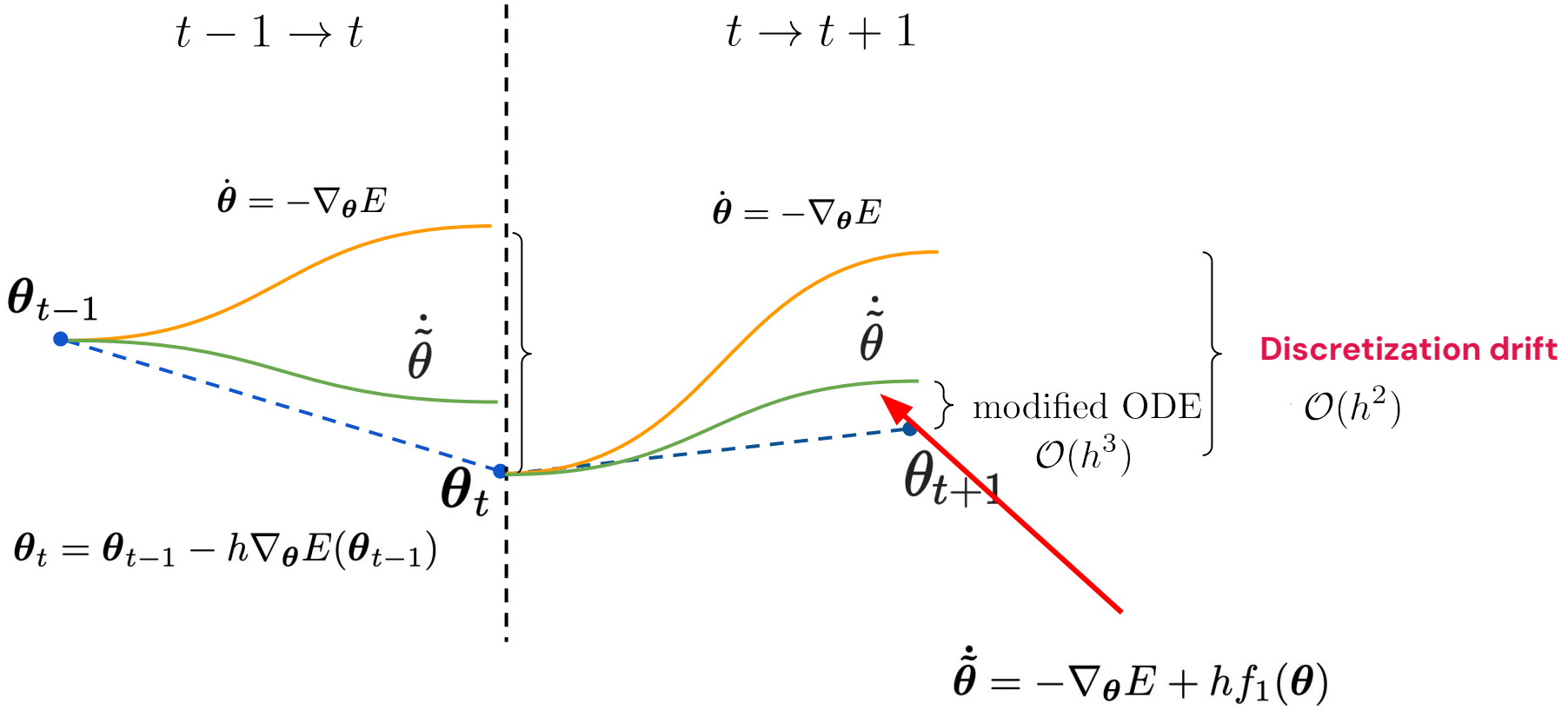
Minimises original loss function E.

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.

Discretization drift for Euler updates

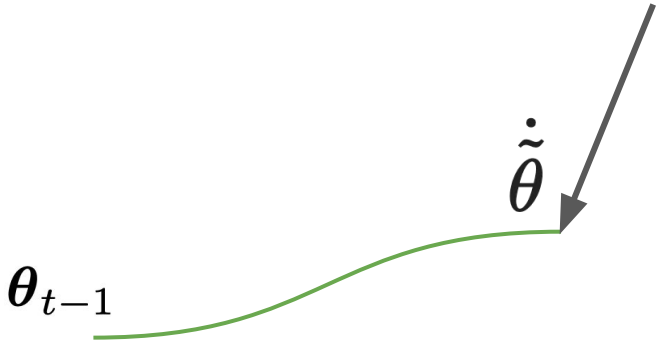


Backward error analysis



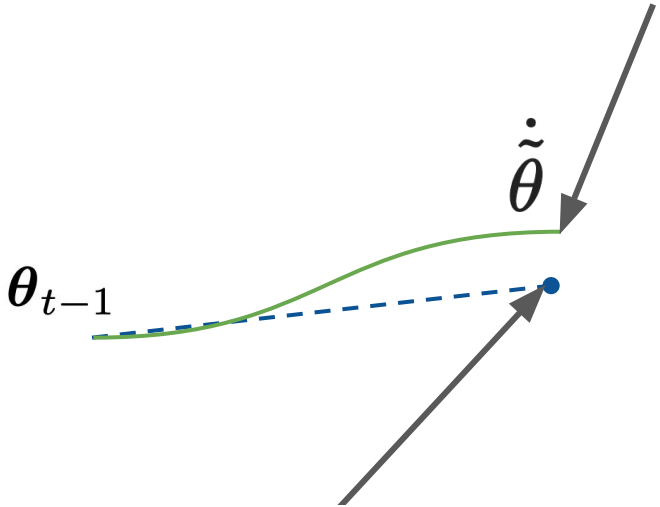
BEA proof visualisation

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



BEA proof visualisation

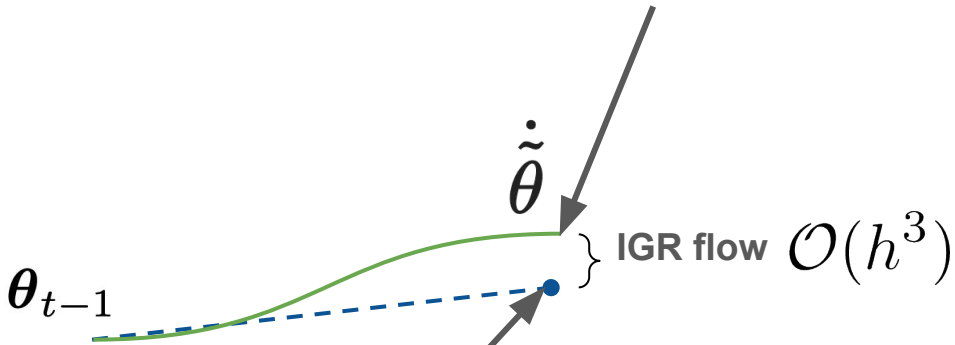
$$\tilde{\theta}(h) = \theta_{t-1} - h\nabla_\theta E(\theta_{t-1}) + h^2 \left[f_1(\theta_{t-1}) + \frac{1}{2}\nabla^2_\theta E(\theta_{t-1})\nabla_\theta E(\theta_{t-1}) \right] + \mathcal{O}(h^3)$$



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

BEA proof visualisation

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

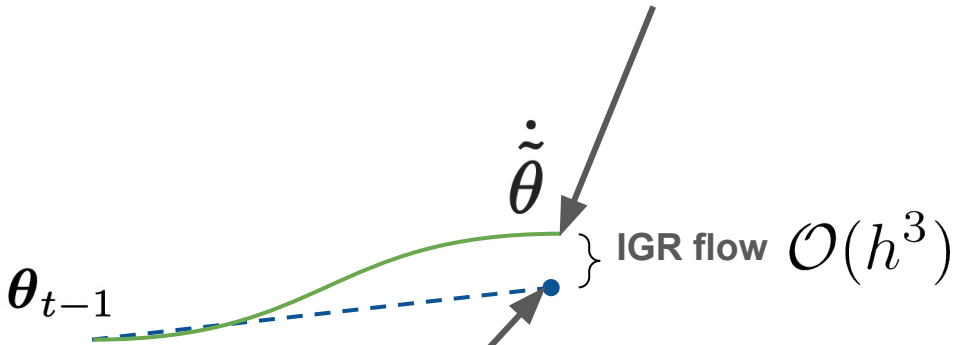


$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - h\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{t-1})$$

BEA proof visualisation

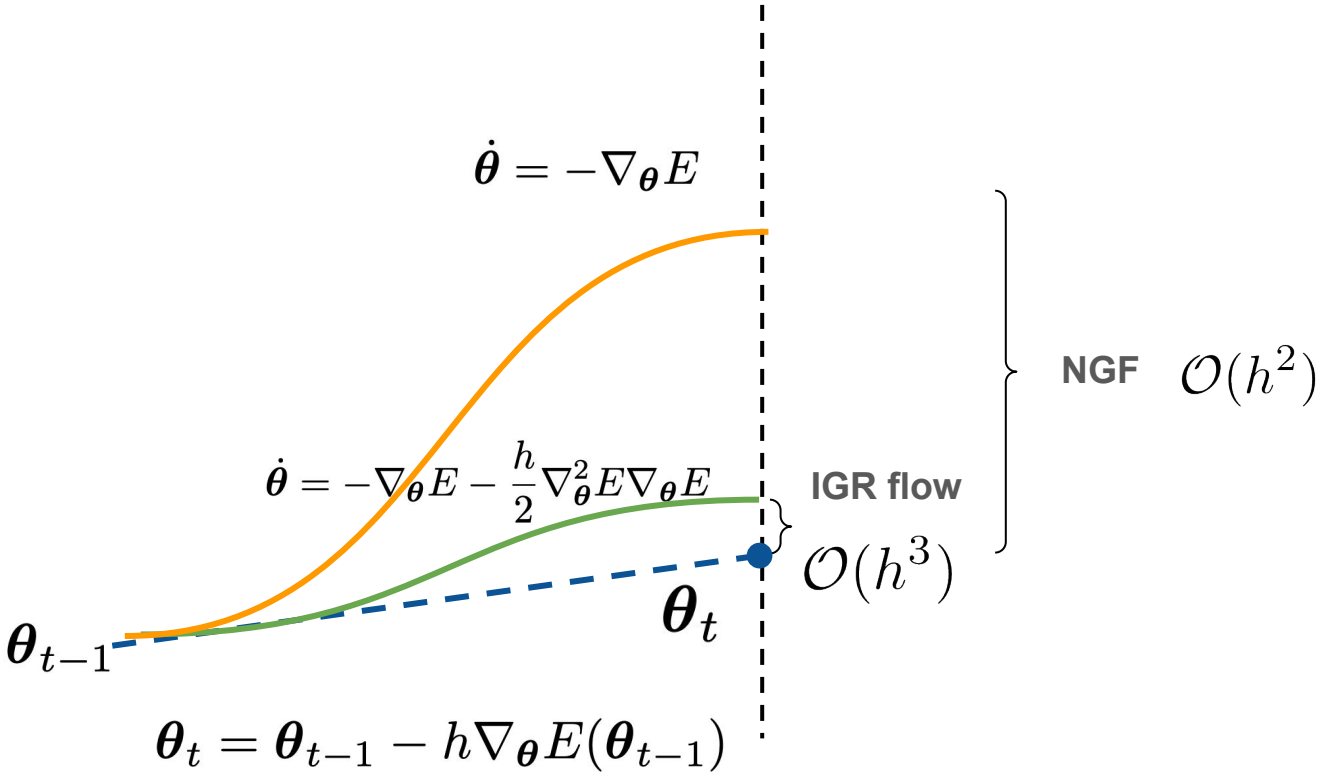
=0

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



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

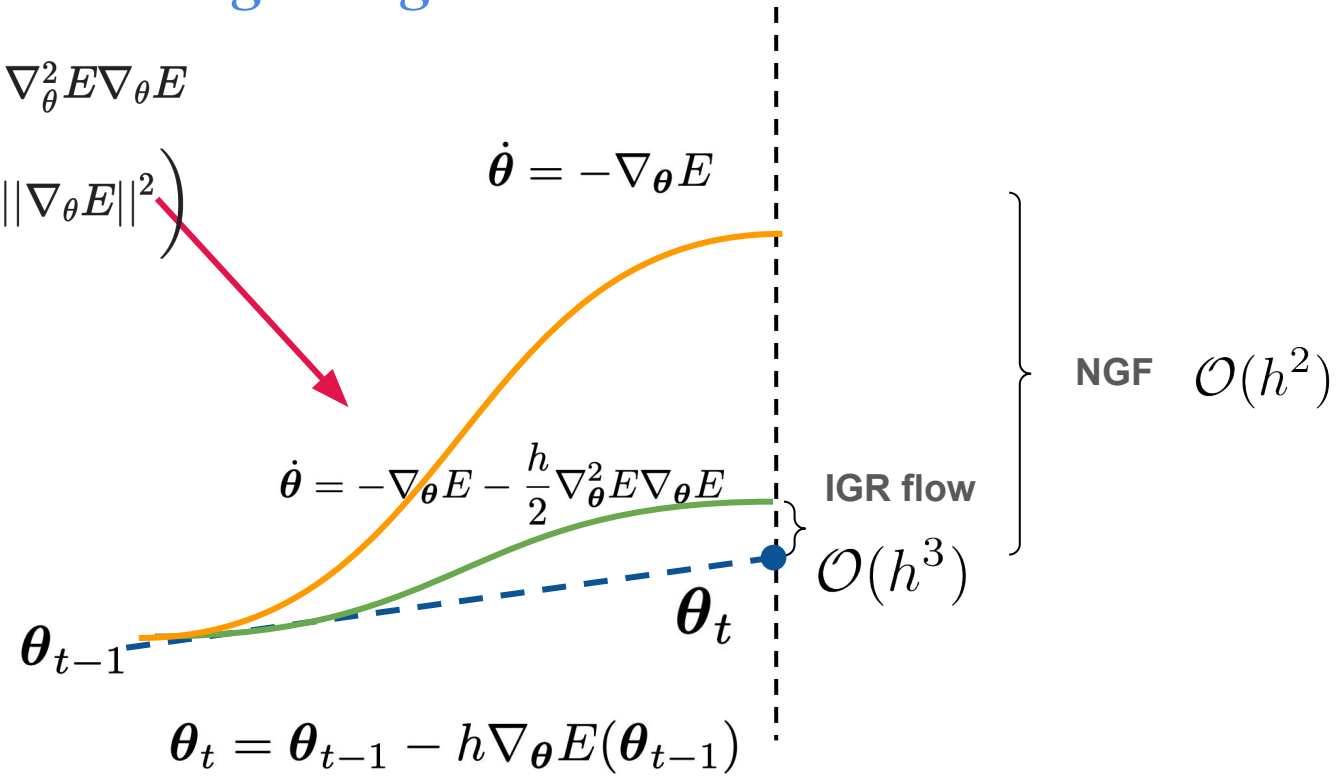
Implicit gradient regularisation



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

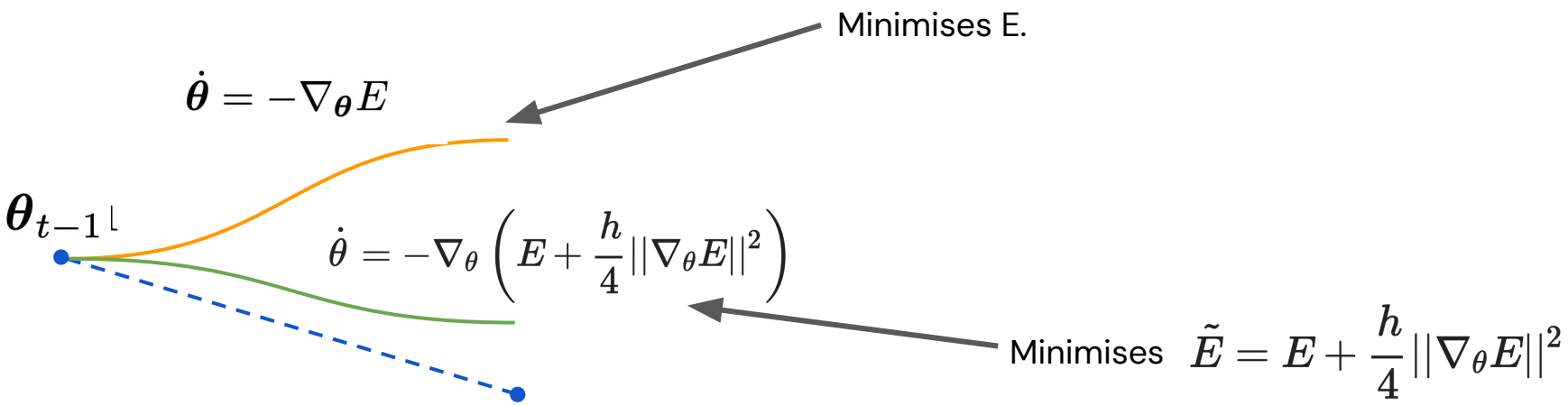
$$\dot{\theta} = -\nabla_\theta E - \frac{h}{2}\nabla^2_\theta E \nabla_\theta E$$

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



Modified loss functions

$$t-1 \to t$$



Implicit gradient regularisation

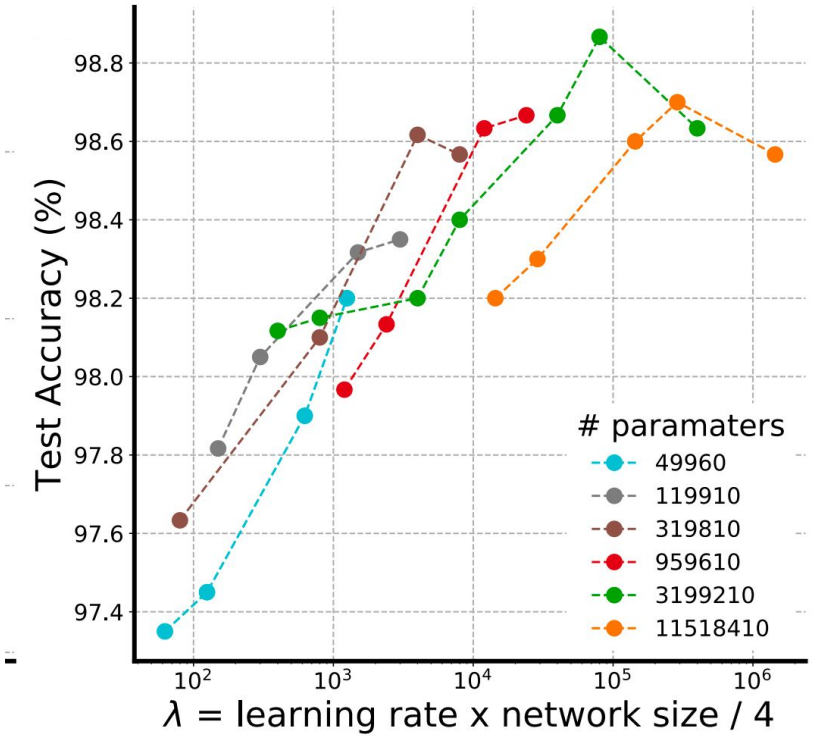
$$\tilde{E} = E + \underbrace{\frac{h}{4}||\nabla_\theta E||^2}$$

Flatness penalty: implicit in when using gradient descent.

Strength is proportional to the learning rate.

Implicit gradient regularisation

Implicit gradient regularisation aids generalization.



Summary

- We can use continuous time tools to find implicit regularisers of optimisers
- Our optimisers induce a bias in our optimisation procedures
- This bias can aid generalisation

Stochasticity

Stochastic gradient descent was introduced to:

- To deal with large datasets
- To avoid the slow training that often happens with full batch GD
    - early in training you do not need a full gradient update to learn, you can learn much quicker with smaller batches

What has happened since?

SGD is a main workhorse of deep learning and has been attributed with:

- improved generalisation performance
- leading to flatter minima
- avoiding bad local minima and saddle points

Note: There is **a critical mini-batch size** after which SGD behaves similarly to full-batch gradient descent (GD) and converges slowly.

Stochasticity and implicit regularisation

Can we use some continuous-time tools to understand the effect of SGD?

Turns out we can, at least in expectation...

Stochasticity and implicit regularisation

For random shuffling SGD:

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

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

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

Is stochasticity then required to obtain good results?

Geiping et al, 2022 take the implicit regularisation from SGD and to full batch gradient descent. Their hypothesis:

"We can modify and tune optimization hyperparameters for GD and also add an explicit regularizer in order to recover SGD's generalization performance without injecting any noise into training."

Stochastic Training is Not Necessary for Generalization, Geiping et al, 2022

Is stochasticity then required to obtain good results?

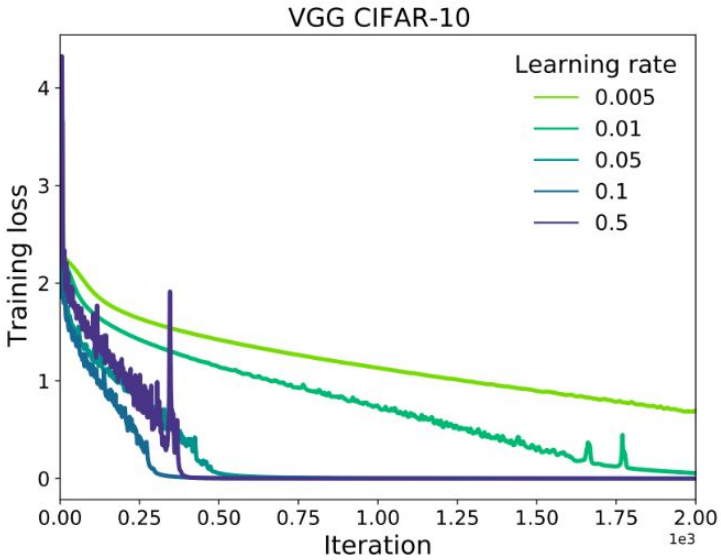
Experiment	Mini-batching	Epochs	Steps	Modifications	Val. Accuracy %
Baseline SGD	✓	300	117,000	-	95.70(±0.11)
Baseline FB	✗	300	300	-	75.42(±0.13)
FB train longer	✗	3000	3000	-	87.36(±1.23)
FB clipped	✗	3000	3000	clip	93.85(±0.10)
FB regularized	✗	3000	3000	clip+reg	95.36(±0.07)
FB strong reg.	✗	3000	3000	clip+reg+bs32	95.67(±0.08)
FB in practice	✗	3000	3000	clip+reg+bs32+shuffle	95.91(±0.14)

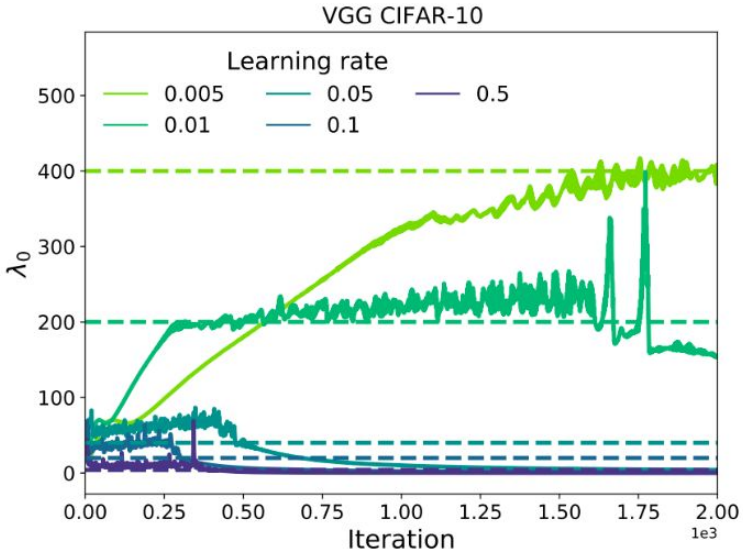
Table 2: Summary of validation accuracies in percent on the CIFAR-10 validation dataset for each of the experiments with data augmentations considered in Section 3. All validation accuracies are averaged over 5 runs.

Summary

- There is an implicit bias coming from stochasticity in gradient descent.
- SGD can have positive regularisation effect that can help with escaping saddle points.
- We can recover the performance of SGD with full-batch GD with explicit regularisation.

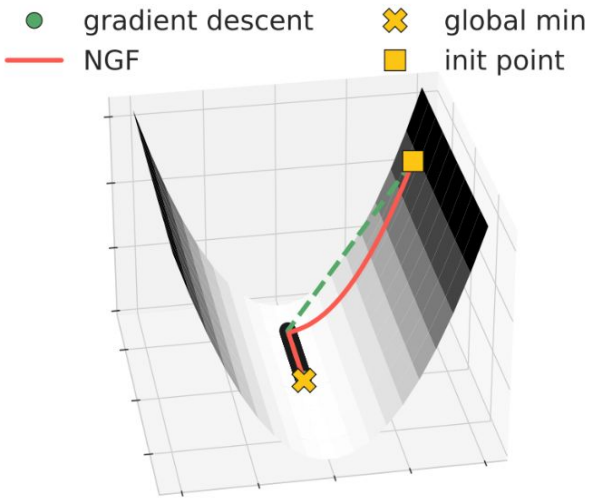
Edge of stability results in NNs



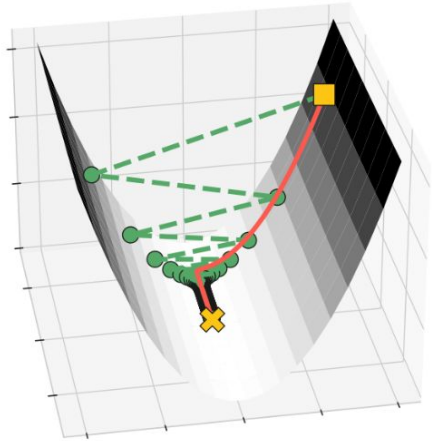


The edge of stability phenomenon in deep learning has been coined by Cohen et al, 2021.

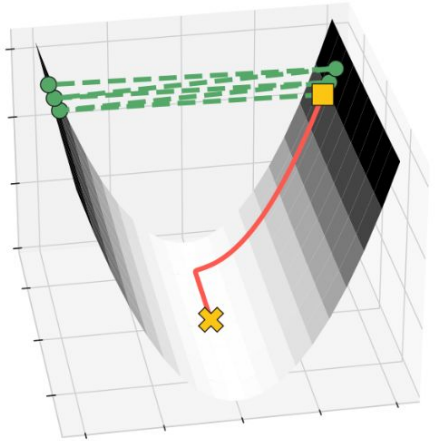
Remember this?



(a) Stability.



(b) Instability.



(c) Divergence.

Edge of stability results in NNs

- Originally seen for full batch gradient descent, but seen been expanded to SGD and adaptive optimisers
  - e.g. Adaptive Gradient Methods at the Edge of Stability by Cohen et al.
- Questions:
  - why does it not diverge, when second order models do
  - implicit regularisation from higher order terms seems to have an effect
    - Self-stabilization: The implicit bias of gradient descent at the edge of stability. Damian et al. 2022
    - On discretisation drift and smoothness regularisation in neural network training, Rosca, 2023

The Principal Flow

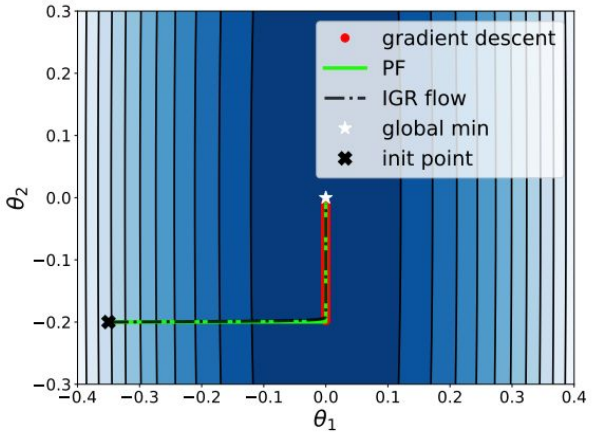
Hessian $\nabla^2_{\boldsymbol{\theta}} E$ eigenvalues and eigenvectors

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

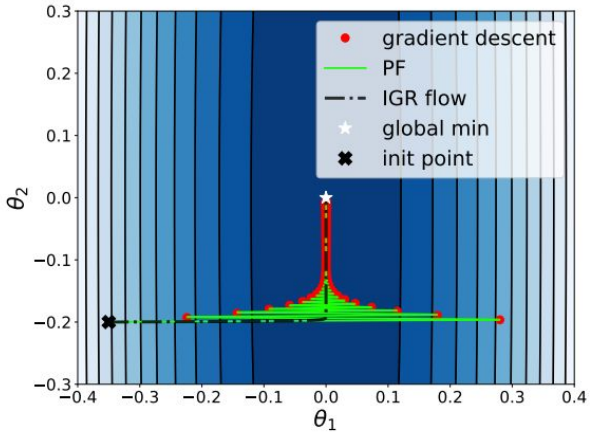
Properties of the Principal Flow (PF):

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

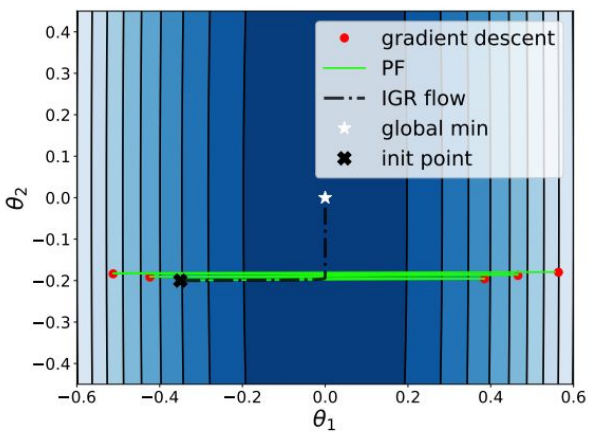
The Principal Flow: exact for quadratic case



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



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



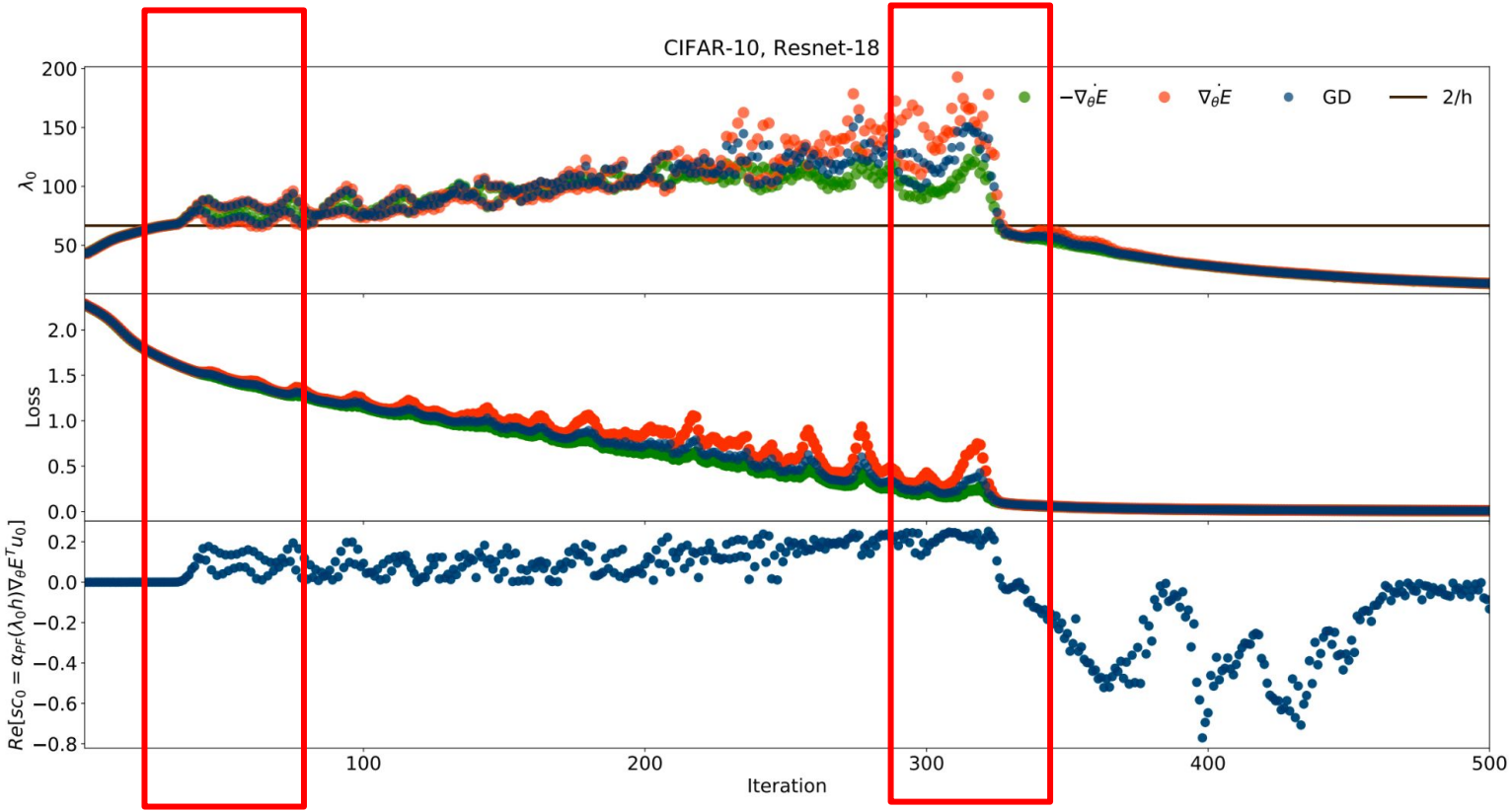
(c) $\lambda_0 > 2/h$ (divergence)

Stability coefficient is real and negative.

Stability coefficient is complex, with negative real part.

Stability coefficient is complex, with positive real part.

The Principal Flow and edge of stability



Summary

- Edge of stability results show that there is a significant role of quadratic dynamics in neural network training, but these cannot fully explain the behaviour we see, and higher order terms matter.

But Mihaela, I just want to train generative models!

Point of view

A lot of progress in generative models has been made not only due to changes in models and objectives, but a lot due to optimisation progress.

Let’s discuss a few case studies and provide historical context.

Generative Adversarial Nets

How to answer “*real or generated?*”

Based on divergences (though not exact!)

- Original GAN: JSD
- Wasserstein GAN
- MMD-GAN
- f-GANs
  - f-divergences like the KL and JSD
- many more works!

Generative Adversarial Nets

?

Goodfellow, et al. **Generative adversarial networks.** NIPS **(2014)**

Karras et al. **A Style-Based Generator Architecture for Generative Adversarial Networks** CVPR **(2019)**

Generative Adversarial Nets

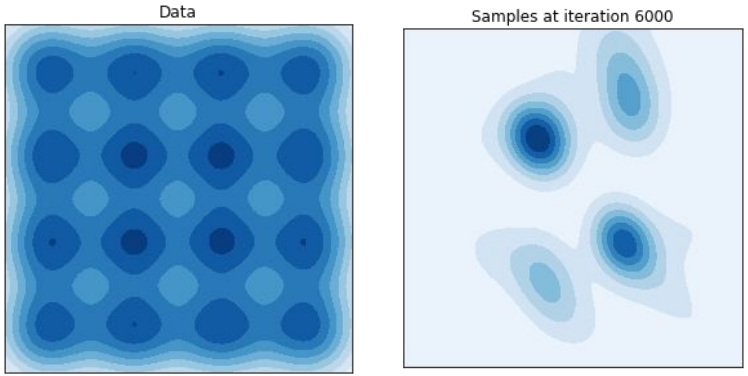
A lot of progress in GAN samples was made with:

- changes to architecture
- increasing the importance of conditioning
- optimization

Generative Adversarial Nets: Optimisation

Tricks to scale GANs:

- optimization unstable
  - large batch sizes help



Generative Adversarial Nets: Optimisation

Tricks to scale GANs:

- Adam:
  - important
  - $beta_1$ (momentum) low: 0 or 0.5

Adam: A Method for Stochastic Optimization, Kingma et al, ICLR 2015

Generative Adversarial Nets: Optimization

Tricks to scale GANs:

- Use Spectral Normalization
  - decreases learning rate sensitivity
  - note: model normalization technique but affects optimization
    - (not only for GANs, also in RL)

$$W \rightarrow W/\sigma(W)$$

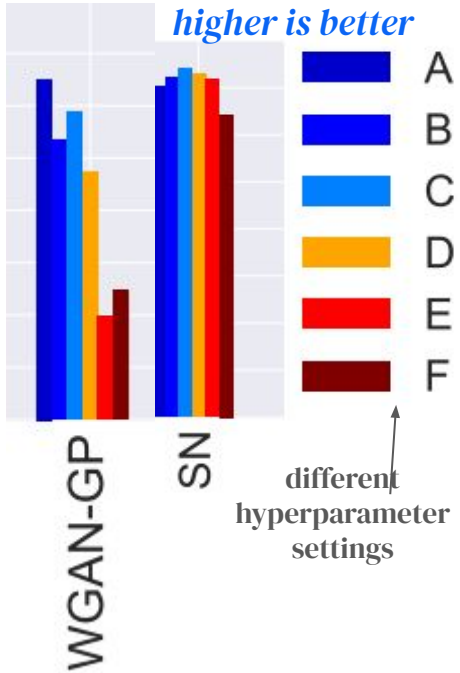
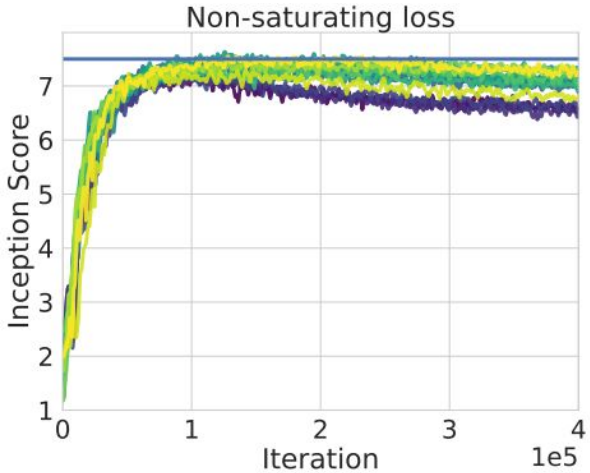


Image credit: Spectral Normalization for Generative Adversarial Networks, Miyato et al, ICLR 2018

Generative Adversarial Nets: Optimisation

Tricks to scale GANs:

- Adam:
  - important
  - $beta_1$ (momentum) low: 0 or 0.5



GAN training
Higher is better

still, performance degrades later in training!

Generative Adversarial Nets: Optimisation

Understanding optimization in generative models can be theoretical!

Observed problem:

gradient descent and its variants (Adam) struggle to train GANs.

Idea:

use continuous time (ODEs) to understand the training dynamics of the model!

ODE-GAN

Theory:

Continuous dynamics converge to a local Nash equilibrium.

Practical implications:

Do not use gradient descent/Adam, but higher order integrators (RK4) to better follow continuous dynamics.

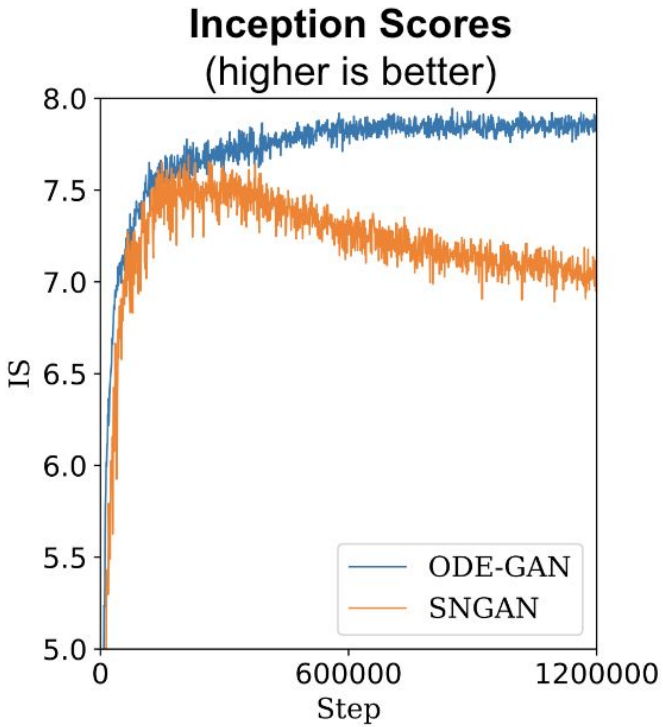


Image credit: Training Generative Adversarial Networks by Solving Ordinary Differential Equations, Qin et al, Neurips 2020

Discretisation Drift in Two-Player games

Question:

What makes gradient descent be unstable, when its continuous counterpart is not?

Quantify mathematically and use it to improve the performance of SGD in GANs!

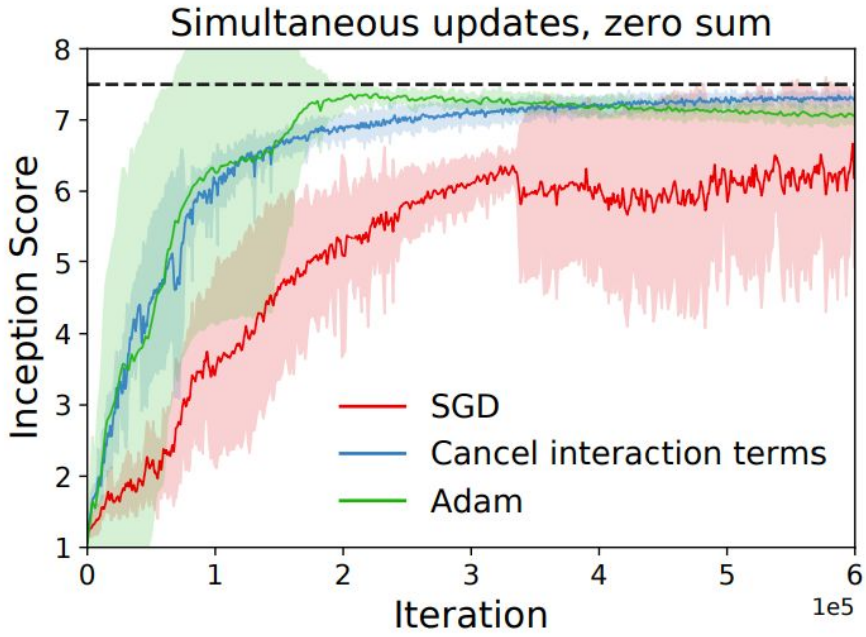
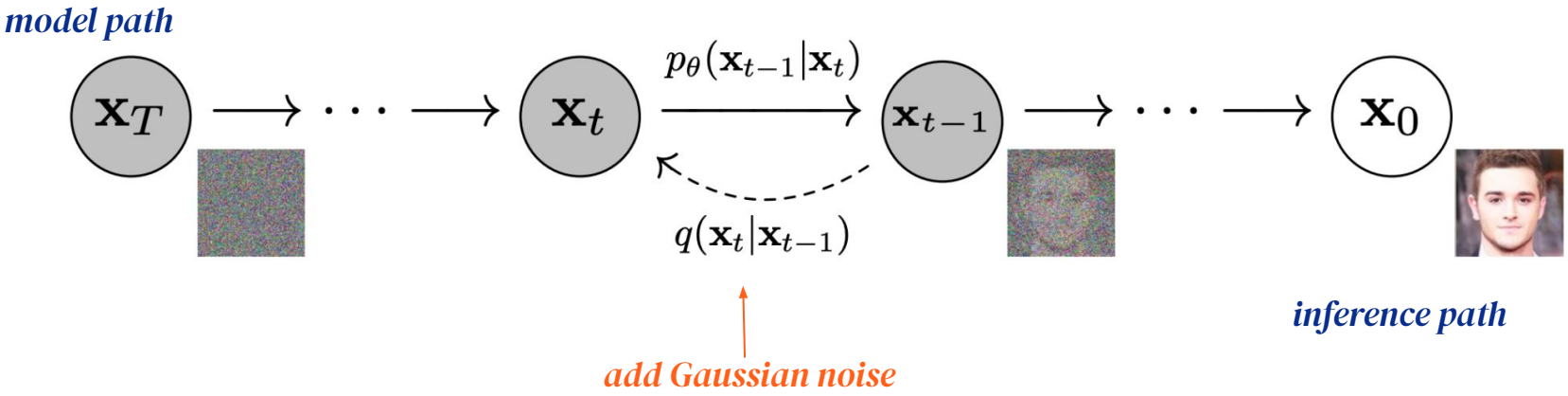


Image credit: Discretization Drift in Two-Player games, Rosca et al, ICML 2021

Diffusion models



You can think about diffusion models as a particular formulation of hierarchical VAEs (though latents have same dimension as data)!

Image credit: Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Diffusion models

$$\mathbf{KL}(p(x_{t-1}|x_t)||q(x_{t-1}|x_t, x_0)$$

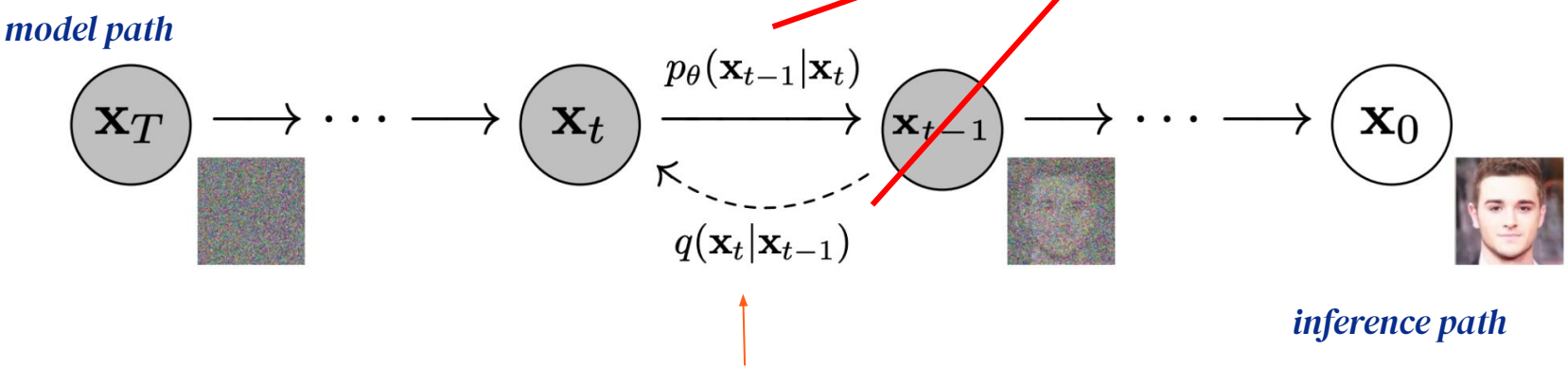


Image credit: Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Diffusion models

$$\mathbf{KL}(p(x_{t-1}|x_t)||q(x_{t-1}|x_t, x_0)$$

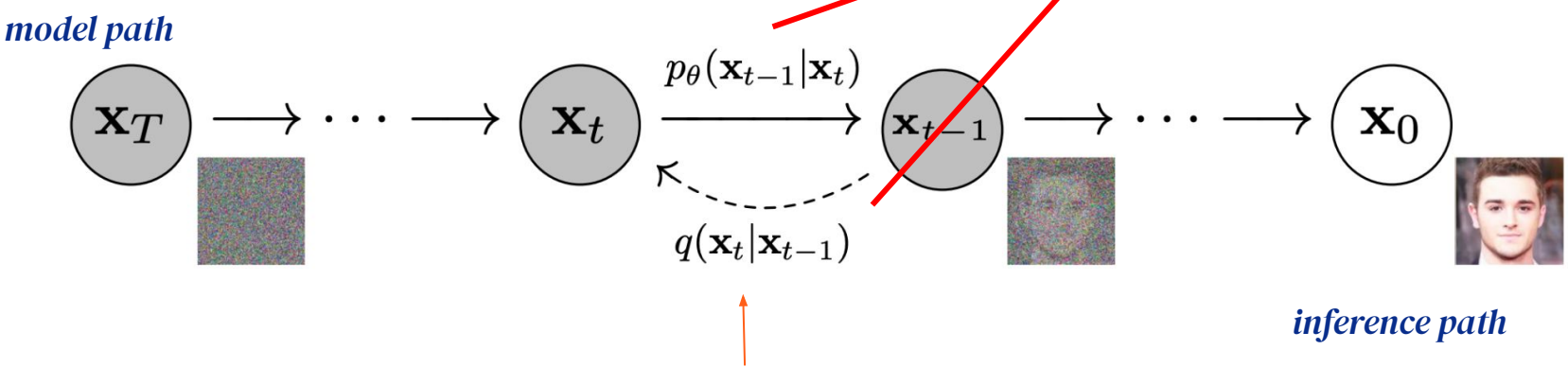


Image credit: Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Diffusion models - optimisation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

regression loss to unit gaussian sample
dense learning signal due to specific parametrisation of Gaussian mean

Algorithm credit: Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Aside:

Continuous-time is also crucial for understanding diffusion!

So is understanding discretisation errors!

Maximum likelihood text LLMs

Challenge: how to stabilize transformers at larger scale (entropy collapse of attention layers).

Unlike other models, transformers cannot easily by trained with SGD with momentum, and need Adam. Learning rate schedules are also very important.

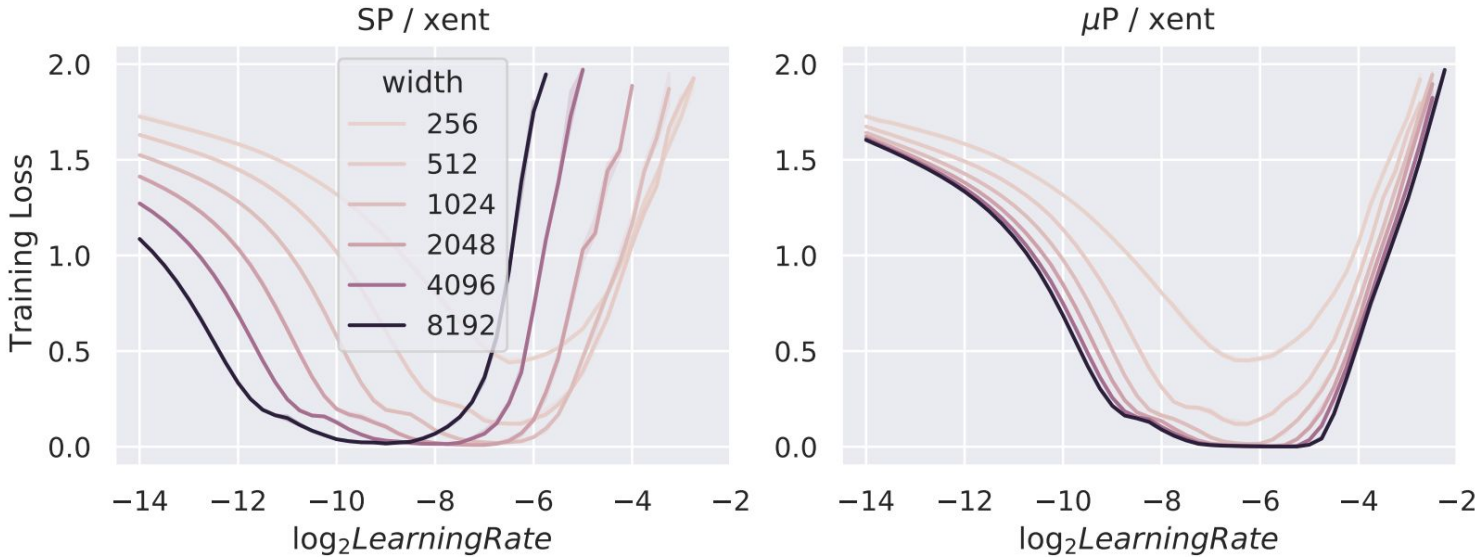
Early evidence suggests this could also be due to curvature properties!

Large language models

We train smaller models for which we can do hyperparameter sweeps, then we ask: “how should I train a bigger model?”

μ-p: Maximal Update Parametrization

If you parametrise a network in a certain way, hyperparameters can be easier to transfer:



Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer, Yang et al, 2022

μ-p: Maximal Update Parametrization

What can transfer:

Table 2: **Examples of μTransferable Hyperparameters.** All of the below can also be specialized to per-layer hyperparameters.

Optimizer Related	Initialization	Parameter Multipliers
learning rate (LR), momentum, Adam beta, LR schedule, etc	per-layer init. variance	multiplicative constants after weight/biases, etc

Key insight: think about the infinite width limit to derive theoretical insights, push empirical results beyond that.

Summary

Optimisation has been important for generative models progress, and still is a key component for driving large model success.

Conclusion

We have seen how to derive optimisers, and how to reason about their implicit regularisation effects.

Optimisation is a key component of the success of deep learning!

Think of it when training your models!

Thanks!

References

Flatness and generalisation

On large-batch training for deep learning: Generalization gap and sharp minima. Keskar, et al. ICLR, 2017

The large learning rate phase of deep learning: the catapult mechanism. Lewkowycz, et al. 2020.

On the relation between the sharpest directions of dnn loss and the sgd step length. Jastrzebski, et al. ICLR, 2019

Flat minima. Hochreiter et al. Neural computation, 1997.

Empirical analysis of the hessian of over-parametrized neural networks. Sagun et al. 2017.

The effect of network width on stochastic gradient descent and generalization: an empirical study, Park et al. ICML 2019.
On the maximum hessian eigenvalue and generalization. Kaur, et al. 2022

Sharp Minima Can Generalize For Deep Nets, Dhin et al, 2017

Effects of noise in SGD

Escaping From Saddle Points — Online Stochastic Gradient for Tensor Decomposition, Ge et al. COLT, 2015.

Sgd implicitly regularizes generalization error, Roberts et al. NIPS Workshop. 2018.

Gradient Descent Only Converges to Minimizers, Lee, et al. Conference on Learning Theory, 2016.

Width of Minima Reached by Stochastic Gradient Descent is Influenced by Learning Rate to Batch Size Ratio, Stanisław Jastrzębski, et al. Artificial Neural Networks and Machine Learning 2018

Stochastic Modified Equations and Adaptive Stochastic Gradient Algorithms, Li et al. ICML 2017.

A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks, Umut Şimşekli, et al. ICML 2019.

Towards Theoretical Understanding of Large Batch Training in Stochastic Gradient Descent. Dai et al. 2018

Never Go Full Batch (in Stochastic Convex Optimization). Idan Amir et al. Neurips 2021.

SGD Generalizes Better Than GD (And Regularization Doesn't Help), Amir et al. COLT 2021

Edge of stability results

Gradient descent on neural networks typically occurs at the edge of stability, Cohen et al. ICLR, 2021.

On the relation between the sharpest directions of dnn loss and the sgd step length, Jastrzebski et al. ICLR, 2019.

Understanding Gradient Descent on Edge of Stability in Deep Learning, Arora et al, 2022

Investigating the Edge of Stability Phenomenon in Reinforcement Learning, Iordan et al, 2023

Self-stabilization: The implicit bias of gradient descent at the edge of stability. Damian et al. 2022

Understanding the unstable convergence of gradient descent, Ahn, et al. ICML, 2022.

On gradient descent convergence beyond the edge of stability, Chen et al. 2023

On discretisation drift and smoothness regularisation in neural network training, Rosca, 2023

Adaptive Gradient Methods at the Edge of Stability, Cohen et al. 2022.

Continuous time modelling in optimisation

Continuous vs. discrete optimization of deep neural networks, Elkabetz et al. Neurips, 2021.

Implicit gradient regularization, Barrett et al. ICLR, 2021.

On the origin of implicit regularization in stochastic gradient descent, Smith et al. ICLR, 2021

Conformal symplectic and relativistic optimization. Franca, et al, Neurips , 2020

Understanding the acceleration phenomenon via high-resolution differential equations, Shi et al, 2021

Continuous time analysis of momentum methods, Kovachki et al. JMLR, 2021.

Stochastic modified equations and adaptive stochastic gradient algorithms, Li, et al. ICML 2017.

On a continuous time model of gradient descent dynamics and instability in deep learning, Rosca et al. TMLR 2023

Discretization Drift in Two-Player Games, Rosca et al. ICML 2021

Implicit regularisation in stochastic gradient descent: from single-objective to two-player games, Rosca et al, 2023.

Citations for the question around modelling momentum with a first order ODE:

Symmetry, conservation laws, and learning dynamics in neural networks. Kunin, et al. ICLR, 2021.

Implicit regularization in Heavy-ball momentum accelerated stochastic gradient descent, Ghosh et al, ICLR 2023

Implicit regularisation in deep learning

The implicit bias of gradient descent on separable data, Soudry et al. JMLR, 2018.

In search of the real inductive bias: On the role of implicit regularization in deep learning, Neyshabur et al. ICLR, 2015.

Understanding the difficulty of training deep feedforward neural networks, Glorot et al. JMLR, 2010.

The implicit regularization of stochastic gradient flow for least squares, Ali et al. ICML, 2020.

Implicit regularization in deep learning may not be explainable by norms, Razin et al. Neurips, 2020.

Sgd implicitly regularizes generalization error. In NIPS 2018 Workshop, Roberts et al. 2018.

Implicit Regularization in Deep Learning, Neyshabur et al. 2017.

Batch size and learning rate connections

Don't Decay the Learning Rate, Increase the Batch Size, Smith et al, ICLR 2018

Train longer, generalize better: closing the generalization gap in large batch training of neural networks. Hoffer et al, 2018

An Empirical Model of Large-Batch Training. McCandlish, et al. 2018

Learning rates as a function of batch size: A random matrix theory approach to neural network training. Granziol, et al. JMLR 2022

Large batch optimization for deep learning: Training BERT in 76 minutes, You, et al. ICLR 2020.

On the SDEs and Scaling Rules for Adaptive Gradient Algorithms, Malladi et al, Neurips 2022

GANs

GANs introduced based on divergences and distances:

Generative adversarial nets, Goodfellow et al, Neurips 2014

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, Nowozin, et al, Neurips (2016)

Wasserstein GAN, Arjovsky, et al, ICML 2017

MMD GAN: Towards Deeper Understanding of Moment Matching Network, Li, et al, Neurips 2017

Improved Training of Wasserstein GANs, Gulrajani et al, Neurips 2017

Demystifying mmd gans, Bińkowski et al, ICL 2018

Learning in implicit generative models, Mohamed, et al arxiv (2016)

GANs

Understanding the effect of discretisation

ODE-GAN: Training Generative Adversarial Networks by Solving Ordinary Differential Equations, Qin et al, Neurips 2020

Implicit competitive regularization in gans, Schäfer et al, ICML 2021

Discretization Drift in Two-Player Games, Rosca et al, ICML 2021

The limit points of (optimistic) gradient descent in min-max optimization, Daskalakis et al, Neurips 2018

Which divergence and distance to use?

Towards Principled Methods for Training Generative Adversarial Networks, Arjovsky et al, ICLR 2017

Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step, Fedus et al, ICLR 2018

Which divergence and distance to use?

Large Scale GAN Training for High Fidelity Natural Image Synthesis, Brock et al, ICLR 2018

A Style-Based Generator Architecture for Generative Adversarial Networks, Karras et al, CVPR 2019

Diffusion models

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein et al, ICML 2015

Generative Modeling by Estimating Gradients of the Data Distribution, Song et al, Neurips 2019

Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Improved Denoising Diffusion Probabilistic Models, Nichol et al, 2021

High-Resolution Image Synthesis with Latent Diffusion Models, Rombach et al, 2022

Diffusion Models Beat GANs on Image Synthesis, Dhariwal et al, Neurips 2021

Classifier-Free Diffusion Guidance, Ho et al, NeurIPS workshop on DGMs and Applications", 2021.

Maximum Likelihood Training of Score-Based Diffusion Models, Song et al, 2021

Hierarchical Text-Conditional Image Generation with CLIP Latents, Dhariwal et al, 2021

Blog posts:
What are Diffusion Models? by Lilian Weng
Guidance: a cheat code for diffusion models, Sander Dieleman

Transformer LLMs and optimisation

Large language models can suffer from optimisation and training related challenges

A Theory on Adam Instability in Large-Scale Machine Learning, Molybog et al, 2023

Stabilizing Transformer Training by Preventing Attention Entropy Collapse, Zhai et al, 2023

Palm: Scaling language modeling with pathways, Chowdhery et al, 2022

Toward Understanding Why Adam Converges Faster Than SGD for Transformers, Pan et al, 2023