Neural Network Optimization:
Global Landscape, Local Structure, and Algorithms

Ruoyu Sun

The Chinese University of Hong Kong, Shenzhen

SIAM Conference on Optimization (OP26)

Edinburgh, UK, June 2-5, 2026

深圳市大数据研究院
Shenzhen Research Institute of Big Data

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

深圳国际工业与应用数学中心
SHENZHEN INTERNATIONAL CENTER FOR INDUSTRIAL AND APPLIED MATHEMATICS

The Edinburgh Connection

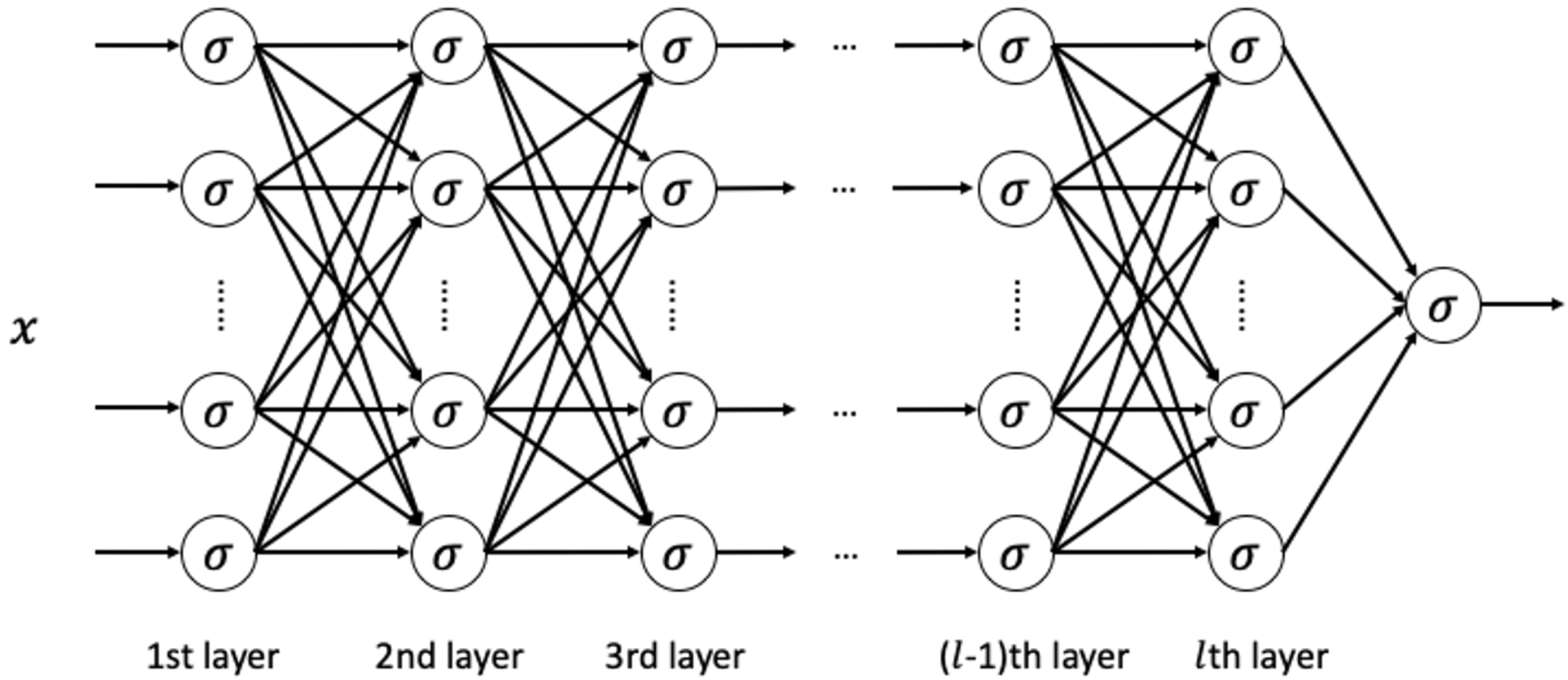
Hinton earned his PhD from the University of Edinburgh.

His advisor, Christopher Longuet-Higgins:
among the few working in **neural networks then**, while symbolic AI dominated.

So, special pleasure to talk about neural network optimization here at the University of Edinburgh.

Introduction:
Optimization for Neural Networks

Multi-layer Neural Network (NN)



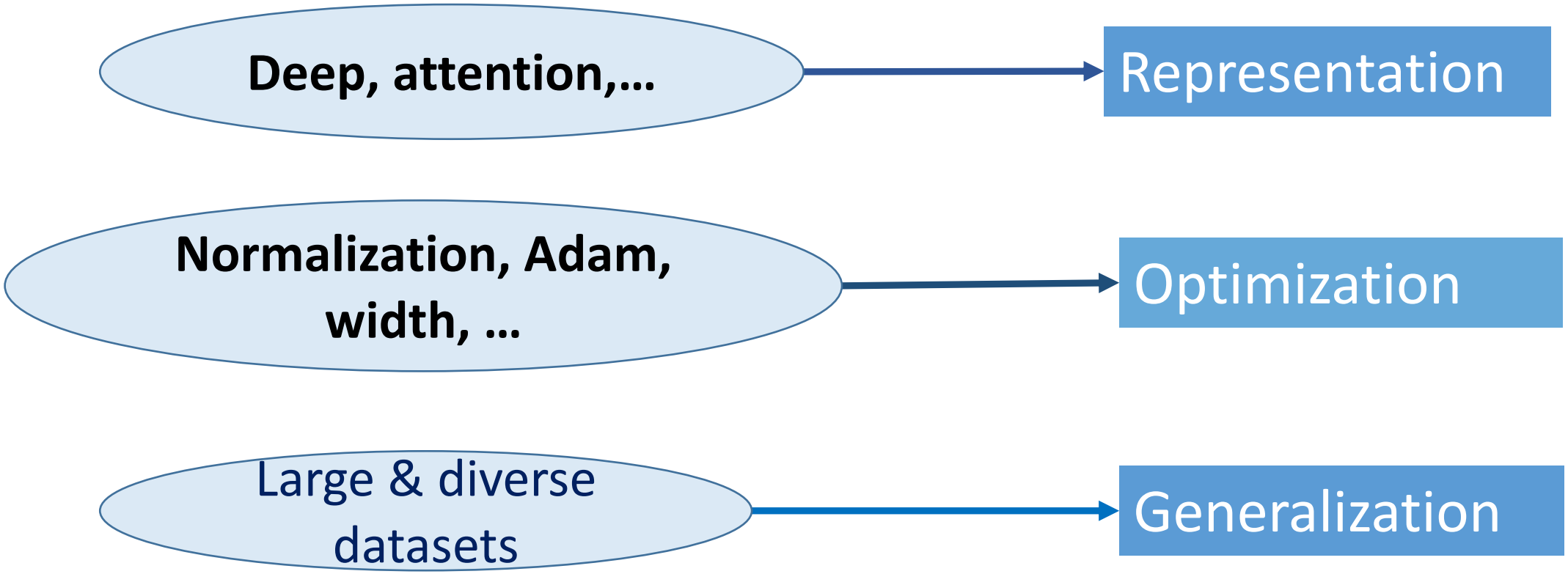
Basic Decomposition of Machine Learning Theory

1. **Representation:** Can a neural network express the target function?
2. **Optimization:** Can training find parameters with small empirical loss?
3. **Generalization:** Will the trained model perform well on unseen data?

Test error = representation error + optimization error + generalization gap

Training error = representation error + optimization error

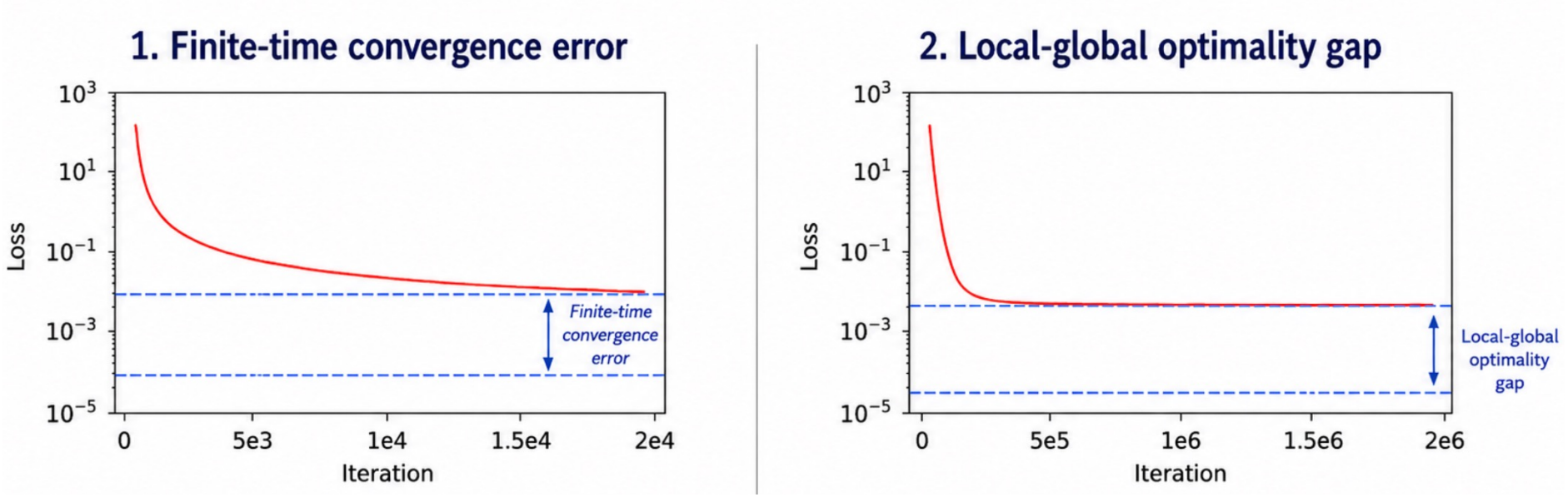
Key Ingredients of Good NN



Decomposition of Optimization Error

$$F(w_T) - F(w^*) = \underbrace{F(w_T) - F(\hat{w})}_{\text{finite-time convergence error}} + \underbrace{F(\hat{w}) - F(w^*)}_{\text{local-global optimality gap}}$$

w_T: iterate after T steps w^*: global minimum $\hat{w}$: stationary point approached by training



Two Simplifying Caveats

Caveat 1

Finite-time convergence error

It may come from:

--non-convergence

--insufficient training time.

For simplicity, we group these two effects together.

Caveat 2

“Local-global gap” is a simplification

Training may approach a stationary point, not necessarily a local minimum.

Strictly speaking, this part contains:

--stationary-to-local-min gap;

--local-to-global gap.

Closing “stationary-to-local-min” gap is related to **escaping saddle points.**

It is a rich area of study, but we omit for simplicity.

Outline of This Talk

Questions:

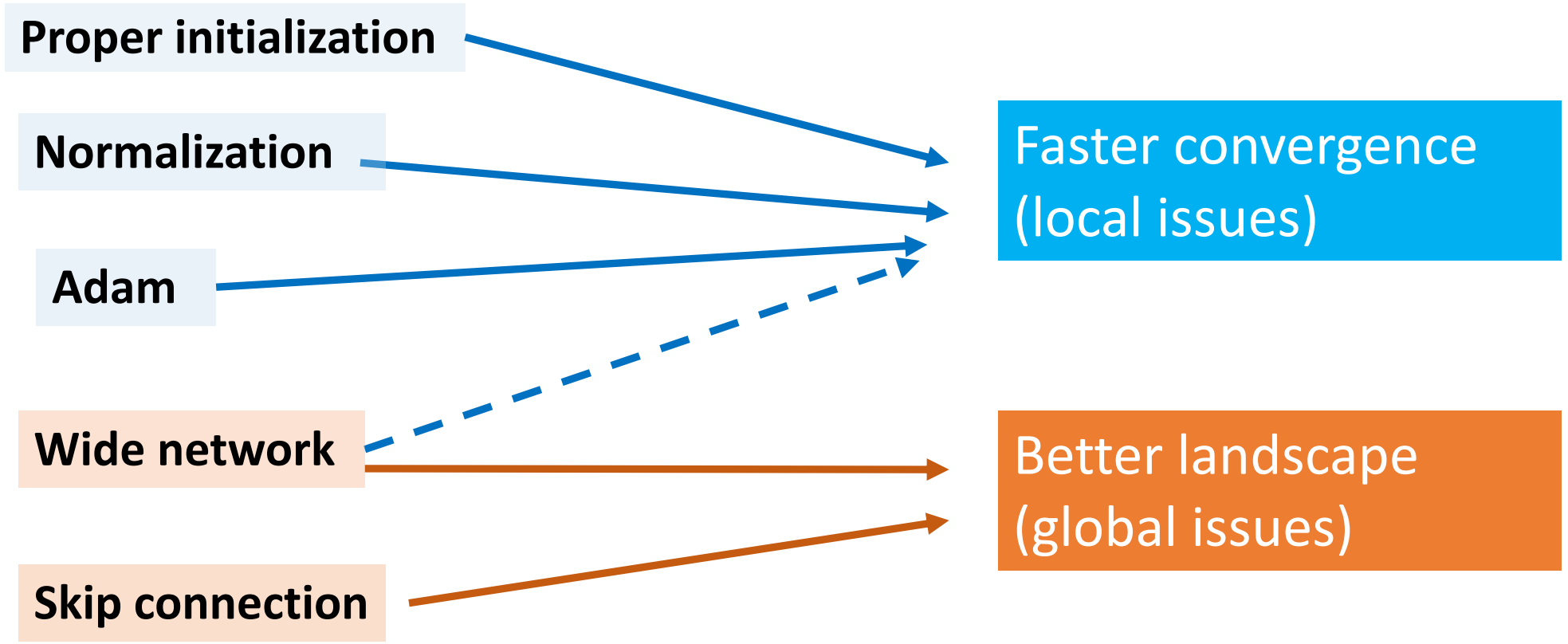
Why are NN optimization problems not as intractable as generic nonconvex problems?

What structures shape the behavior of training algorithms and the choice among them?

In this talk, we discuss these questions from two complementary views.

- **Global landscape**
- **Local Structure and Algorithms**

Linking Training Gadgets with Optimization Effects



Reference: Optimization for deep learning: an overview, Ruoyu Sun, JORSC, 2020.

Disclaimer: They may help generalization and representation, but we focus on optimization.

Part I

Global Landscape of Neural Net Optimization

Basic Formulation of Neural Net Optimization

- Consider a supervised learning problem with n samples (x_i, y_i), i=1,2,…,n
- A fully-connected neural network:

$$f_\theta(x) = W_L\sigma\big(W_{L-1}\dots W_2\sigma(W_1)\big)\dots)$$

Here $\sigma\colon \mathbb{R} \to \mathbb{R}$ is a scalar function, and W_l is a matrix.

- The optimization problem is (minimizing empirical loss):

$$(P1)\quad \min_\theta F(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

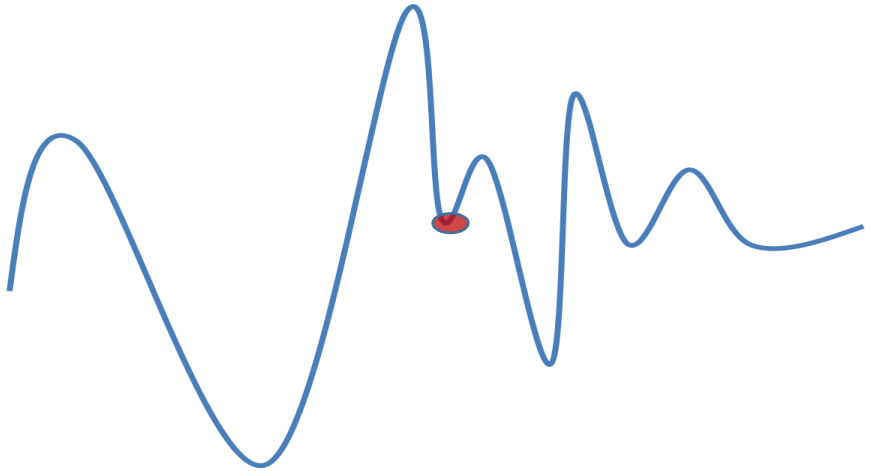
Features:
(F1) **Objective:** Composition of a (often convex) loss function and a nonlinear neural net;
(F2) NN is a composition of matrix multiplications with pointwise nonlinear functions.

Remark: Broadly, neural-net optimization can involves any optimization problem with neural-nets as part of it; e.g. GAN.
In this talk, we restrict to supervised learning, where the outer function is convex (i.e., assume F1).

Non-convexity of Neural Network Optimization

"*They speak as though becoming **trapped** on **local maxima** were rarely a serious problem*."

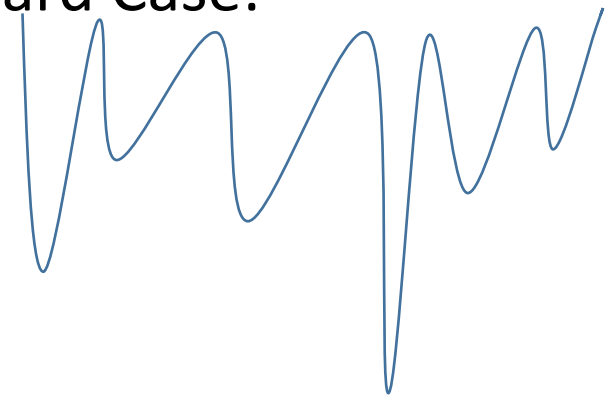
--Minsky and Peppert, "**Perceptrons**" (expanded), 1988



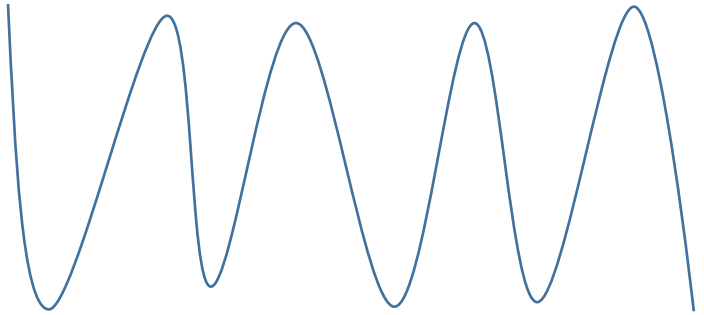
- non-convex, local optima

Neural Network Optimization: Easy or Hard?

- Hard Case:



Easy Case:



Non-convex ≠"hard"

Mathematically, $W_L\sigma\big(W_{L-1}\dots\sigma(W_1X)\big)$.
NOT arbitrary problem!

Initial intuition:
Special NN: Matrix factorization $\min_{X\in\mathbb{R}^{m\times r},Y\in\mathbb{R}^{n\times r}}||M-XY^T|||^2$
is 2-layer linear net, and it has no bad local minima

Is this intuition enough?
Is neural-net optimization easy or hard?

Dauphin's Work and Optimism

- [Dauphin et al. NIPS 14] claimed:
  --Local minima are NOT a challenge for training NNs.
  --Saddle points are the major challenges.

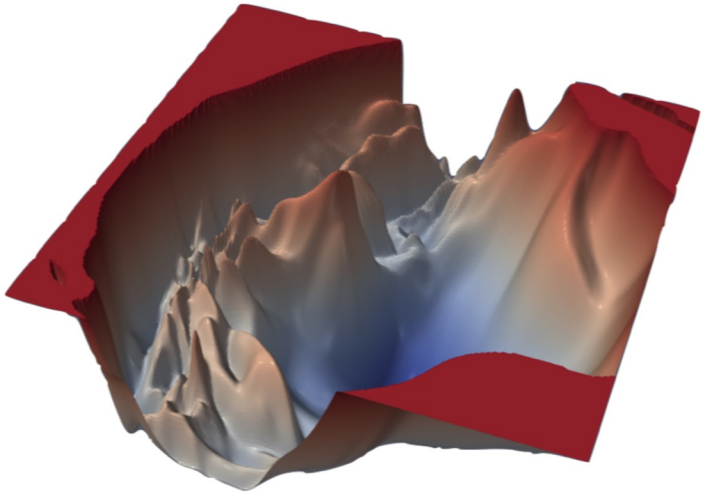
- Around 2014-2015, LeCun said in a few places that local minima are not a challenge for NNs,
  Thus in a theory paper [Choromanska-Henaff-Mathieu-Arous-LeCun, ICML 2015]
  they use spin glass theory (random matrix) to analyze the layers of saddle points,
  for a multi-layer linear networks.

- [Kawaguchi NIPS 16]: For mutli-layer linear networks, any local-min is a global-min.

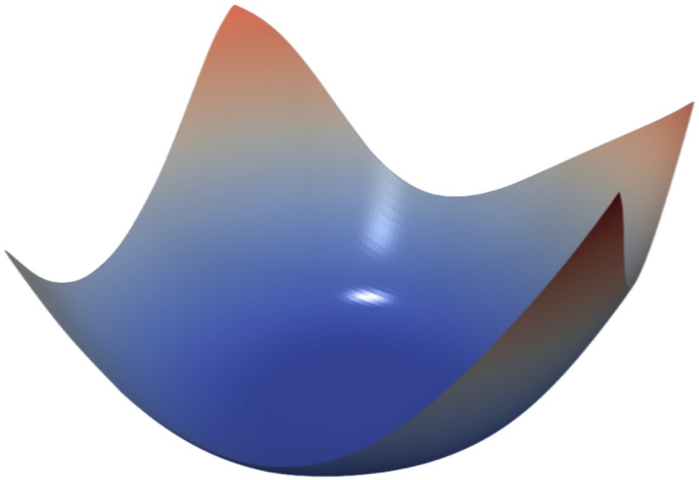
- **Unclear**: rigorous results for nonlinear nets?

Contents

“Smoother” landscape Requires Large Width



Narrow



Wide

Note:
These figures are the projection
of the true landscape onto low-dim space;
NOT the original high-dim landscape

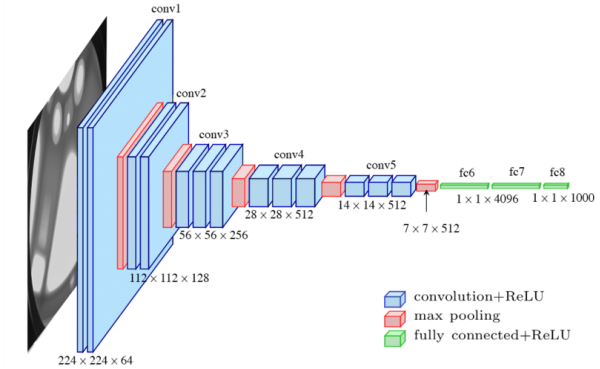
Figures from Li, Xu, Taylor, Studer and Goldstein, Visualizing the Loss Landscape of Neural Networks, NIPS 2018

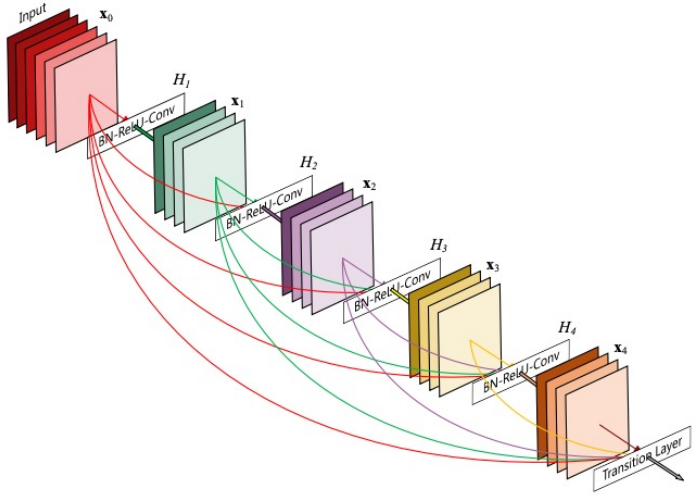
Empirical observations from Li et al.’s work:

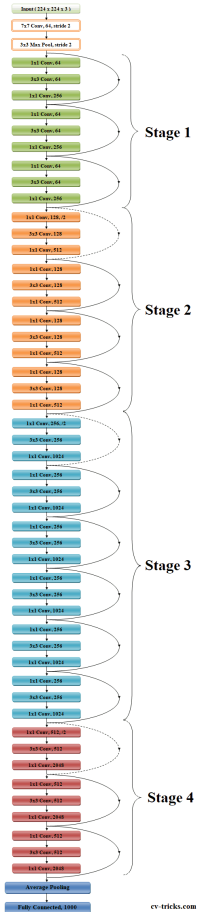
1) Not all neural-nets are easy to optimize; narrow nets have complicated landscape
2) Increasing **width** can make landscape smoother

Disclaimer: The two figures are originally for NNs without and with skip connections;
the narrow and wide nets figures have similar flavor but visually the comparison is not as striking.

Some Neural Nets are Wide



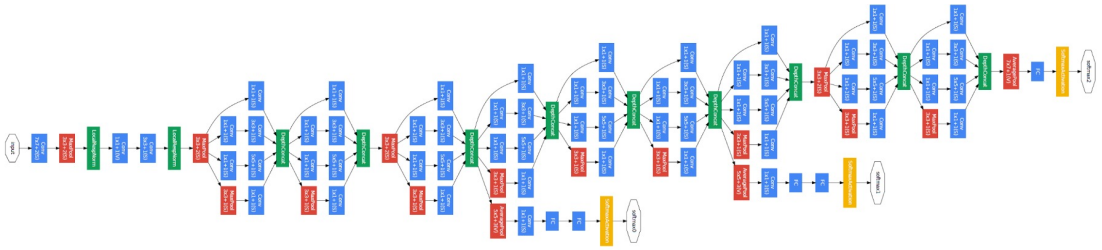




Some NNs are deep and wide.

On ImageNet, training size n = 1.3m.

	depth	width
VGG19	19	30m
InceptionV3	46	13m
ResNet52	52	0.8m



Towards Concrete Results

- **Observation**: width helps smooth landscape

  More **rigorous theory?**

  Q: What is the benefit of width in neural-nets?

  What does “smoother” mean, in math?

Results

Below, “bad” means “sub-optimal”

1) **Wide** networks can have bad local-min.
   [Venturi-Bandeira-Bruna, JMLR 19], [Li-Ding-Sun, MOR 21],…

2) Mildly **wide** networks have no bad “basin”
   [Nguyen-Mukkamala-Hein, ICLR 19], [Venturi-Bandeira-Bruna, JMLR 19], [Li-Ding-Sun, SIOPT 22]

3) **Narrow** networks can have bad basins [Li-Ding-Sun, SIOPT 22]

Implication: Width itself eliminates ***bad basins***, NOT bad local-min.

Remark: NTK type results (e.g. Jacot et al. 2018; Du et al. 2018; Allen-Zhu et al. 2019; Zou et al. 2018) require significantly higher width, often at least $\Omega(n^2)$ to show GD converges to global minima.

Examples of Wide Nets with Local Minima

For m, n > 2, examples of existence of sub-optimal local-min.

Assumption 1: d^2+3d/2 +1 $\leq$ n. (d: input dim; n: # of samples)
Assumption 2: There exists $a \in \mathbb{R}$ and $\delta > 0$, such that
a) σ is twice-differentiable on [a - δ, a + δ];
b) $\sigma(a), \sigma'(a), \sigma''(a) \neq 0$.

Assumption 2 covers standard smooth activation, i.e., Swish, GeLU, tanh.

Thm 1 [Ding-Li-Sun, MOR 21] Under Assumption 1 and 2, for any $\mathrm{m} \geq n \geq 3$, there exits output Y, s.t. for 1-hidden-layer neural-net with m neurons, (P1) has sub-optimal local-min.

For piecewise linear activation, we have a similar result [R1, Thm2]. (also in [Wang et al.'ICLR 20])

Existence of bad local-min: Different Settings

There are many papers that find bad local minima for neural nets; but many of them have restrictions.

[Ding-Li-Sun, MOR 21] **provide direct strong examples for common activations.**

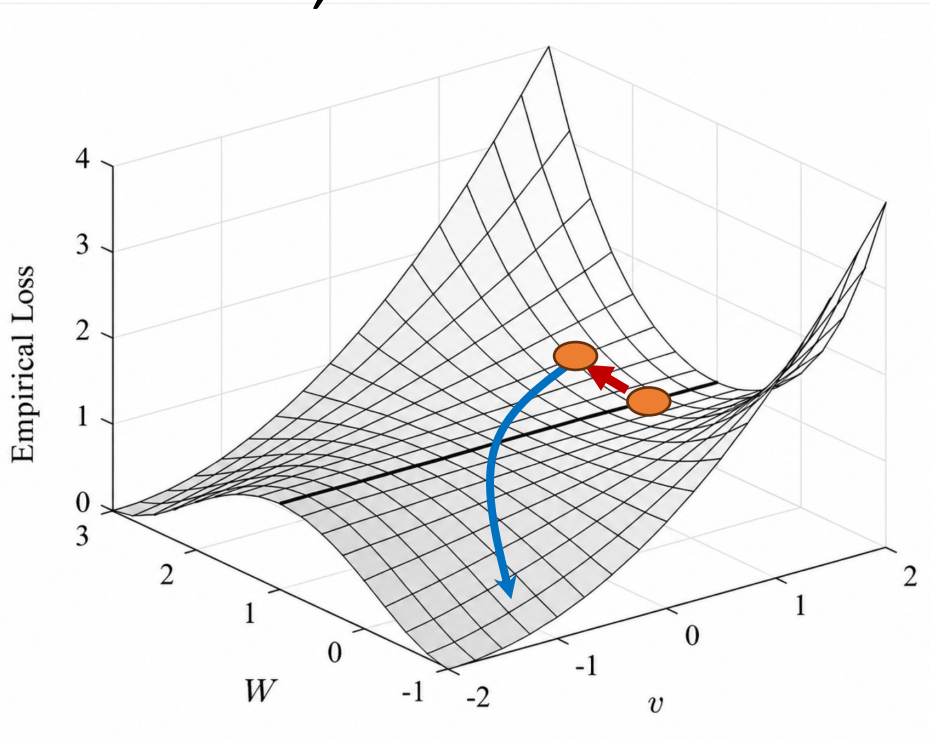
Reference	Width	Depth	Activation	Input data
Auer et al. [3]	One	One	Continuous, boundeda	Positive measure
Swirszcz et al. [61]	Two or three	Two	Sigmoid, ReLU	Fixed
Zhou et al. [72]	One	One	ReLU	Fixed
Safran et al. [52]	6 to 20	One	ReLU	Gaussian
Liang et al. [35]	Any	Any	$\sigma(t)+\sigma(-t)=c$	Fixed
Yun et al. [68], theorem 1	Any	One	Piecewise linear	Generic
Yun et al. [68], theorem 2	Two	One	Small nonlinearity	Fixed
He et al. [23]	Any	Any	Piecewise linear	Generic
Ours, Theorem 1	Any	Any	Generic smooth	Generic

Remark: We allow some freedom in **picking output Y**.

--If allowing label smoothing (manipulating Y), it is not impossible to get around local-min --- open question.

Prototype Landscape

- With a non-linear neuron, bad local-min can appear, e.g. $(vw^2 - 1)^2$



Is the landscape that bad?

GD with noise could escape this local-min via a decreasing path.

Escapable Local-min v.s. Non-escapable Basins

Non-basin local-min: river

∃ path to sea along riverbank

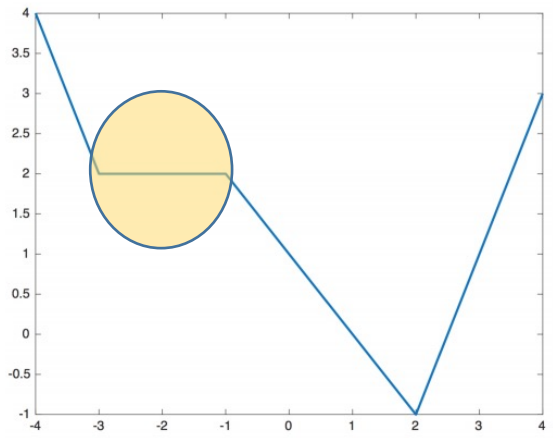
Basin: barrier lake

stuck!

No basin v.s. existence of basin

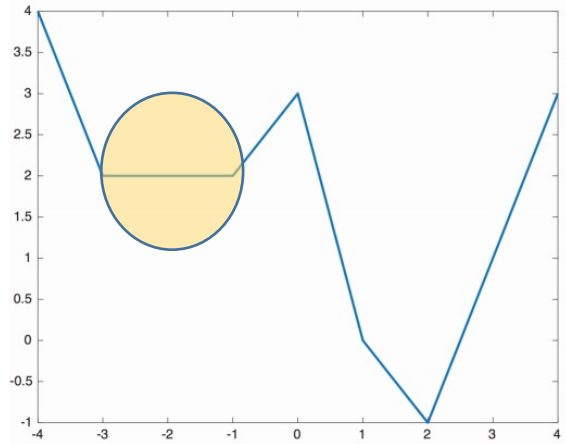
Set-wise strict local-min (SSLM; informally, "basin"): [Josz et al.'18] compact set S surrounded by "barriers".
Formally: if there exists $\varepsilon > 0$ s.t. any point w in ε-neighborhood of a compact set S
satisfies $f(w) > sup_{v \in S} f(v)$, then S is an SSLM.

Here the ε-neighborhood of S is $\{w \notin S: \exists\, v \in S, \text{s.t. } \| v - w \| \leq \varepsilon\}$

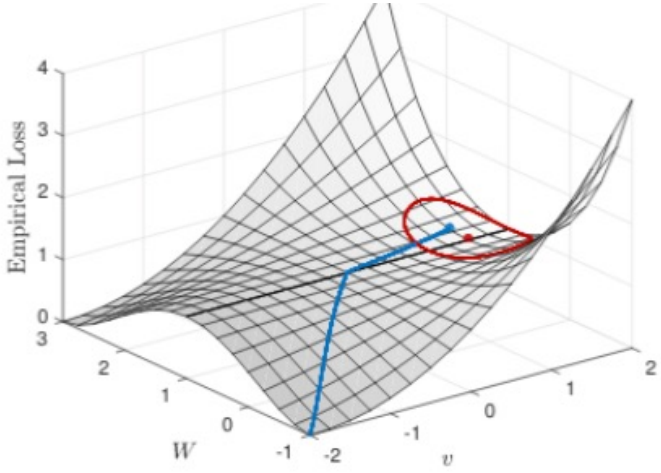


No basin

Has bad local-min.



Has bad basin.



No basin

Has bad local-min.

Results on Wide Nets and Narrow Nets

Assumption 1: loss function l is convex
Assumption 2 (distinct data): input data are distinct
Assumption 3 (neurons): activations are continuous
Assumption 4 (last wide layer): the last hidden layer has $\geq n$ neurons
We have a result with a wide middle layer; skip details here.

Thm 2 [Li-Ding-Sun, SIOPT 22] Under Assumptions 1-4, the objective function in (P1) has no sub-optimal basins.

This implies: width-n neural net has no sub-optimal basin.

Thm 3 [Li-Ding-Sun, SIOPT 22] For any n**,** for any non-zero distinct 1D input $x_1, x_2, \dots, x_n$, $\exists$ activation σ, $\exists$ output $\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_n}$, s.t. for the width-(n-1) 2-layer-neural-net with activation σ, the loss function has sub-optimal strict local minima.

Implication: Why is Training Small Nets Difficult?

Reason 1 seems simple, but it can help explain:
small networks are hard to train, partially due to bad basins.

Empirical evidence:

Mode connectivity (level set connected):
[Freeman–Bruna, ICLR'17; Garipov et al., NeurIPS'18;
Draxler et al., ICML'18]
For standard networks, different initialization converge to connected solutions.

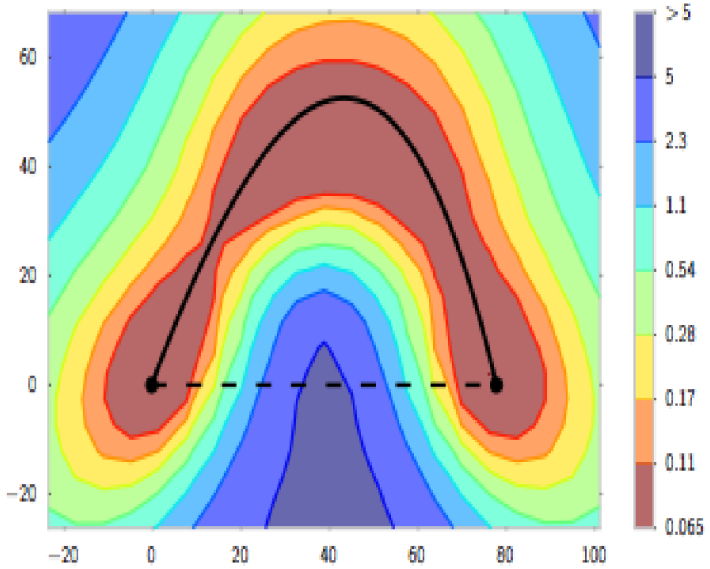


Figure: Two solutions connected by an equal-value curve searched in a large space.

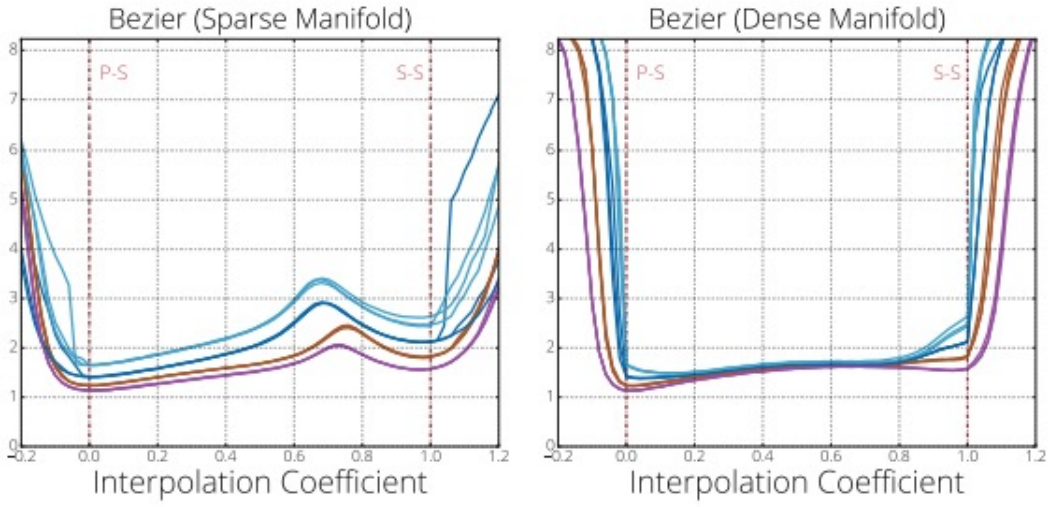
Implication: Why is Training Small Nets Difficult?

Reason 1 seems simple, but it can help explain:
small networks are hard to train, partially due to bad basins.

Empirical evidence:

Mode connectivity (level set connected):
[Freeman–Bruna, ICLR'17; Garipov et al., NeurIPS'18; Draxler et al., ICML'18]
For standard networks, different initialization converge to connected solutions.

"**Difficulty of Training Sparse Neural Networks**"
[Evci-Pedregosa-Gomez-Elsen 19] found:
For smaller networks, different initialization lead to different "basins" (barrier in between).



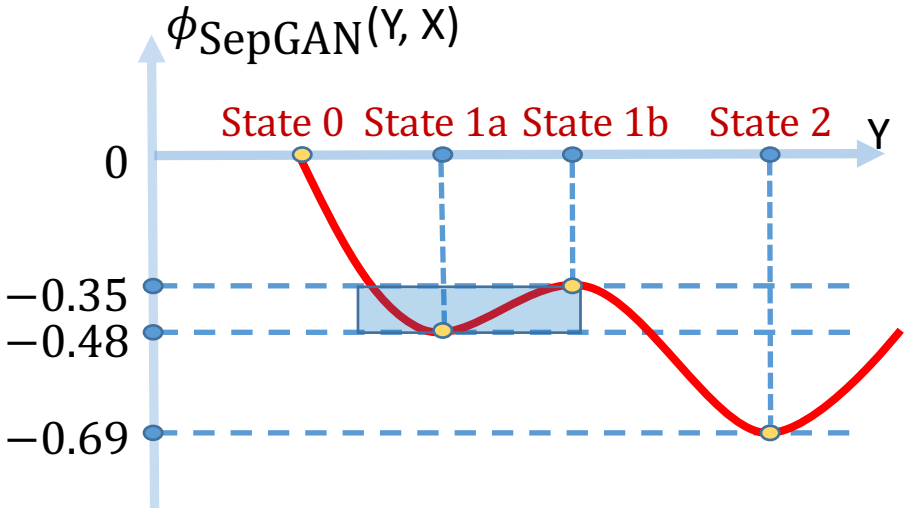
Left: On a smaller network, there is **"barrier"** between two converged solutions.

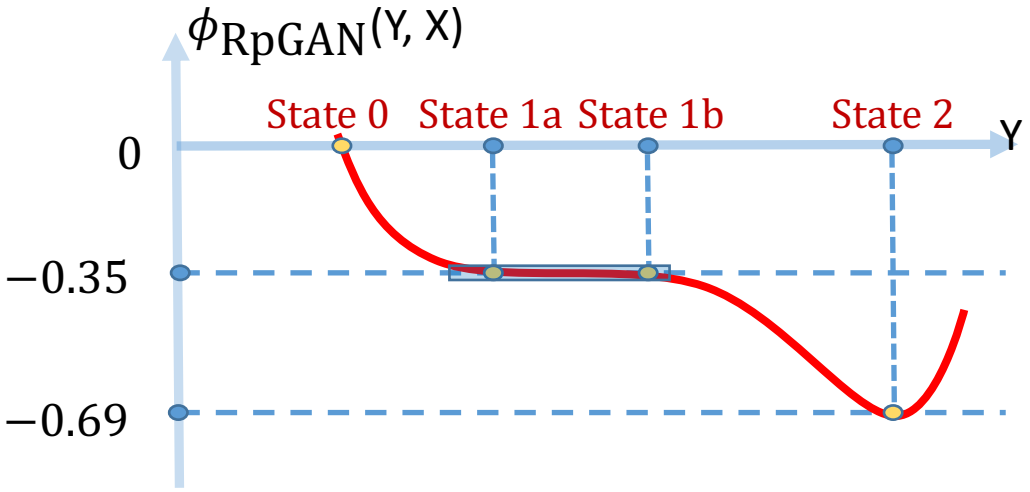
Right: On a standard network, there is no barrier between two converged solutions.

Implication: Modern Baseline of GAN Training

GANs (generative adversarial nets) are considered hard to train.

We prove: RpGAN has no bad basins & no bad local NE. [Sun-Fang-Schwing, 2020, 2026]





The GAN is dead; long live the GAN!
A Modern Baseline GAN

Yiwen Huang Brown University
Aaron Gokaslan Cornell University
Volodymyr Kuleshov Cornell University
James Tompkin Brown University

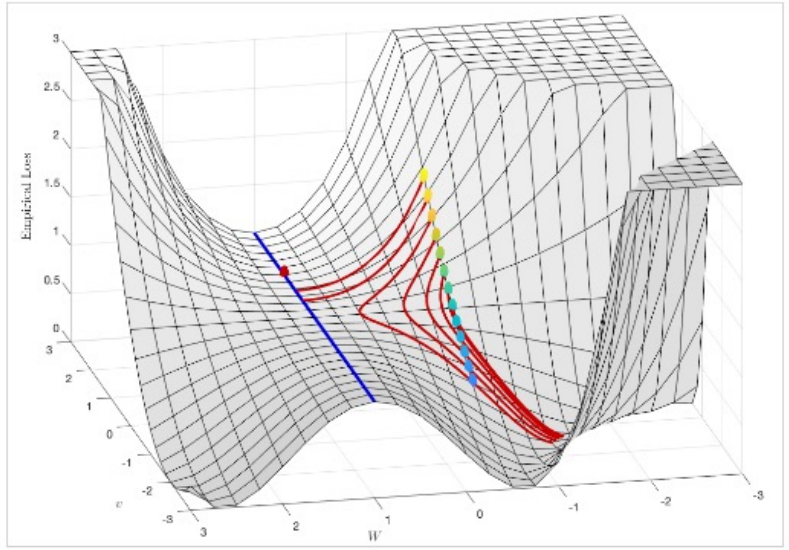
[Huang et al., NeurIPS 2024] empirically verify:
RpGAN with regularizers is easy to train;
can outperform diffusion models.

Contents

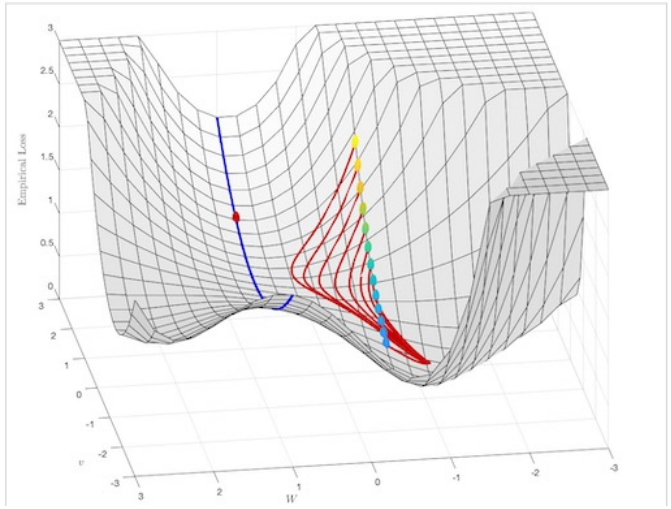
Tilting Landscape

For the original problem, eliminating local-min seems hard; what about modifying formulation?

unmodified landscape



"tilted" landscape: good



Intuition: "tilting" the left landscape can possibly eliminate all bad local-min.

Challenge: Perturbation may create a basin. How to tilt a landscape "well"?

Rigorous Theory?

Intuition: "tilting" the left landscape can possibly eliminate all bad local-min.

- What modification can achieve this goal?
  Answer: regularization.

Theorems (informal) [Liang-Sun-Srikant, SIOPT 21; arxiv]
For **1-hidden-layer** ReQU and ReLU width O(n) NN with regularizers, no bad local-min!

[R3] S. Liang, **R. Sun,** R. Srikant, **"**Revisiting landscape analysis in deep neural networks: eliminating decreasing paths to infinity", **SIAM Journal on Optimization 2021**.
[R4] S. Liang, **R. Sun** and R. Srikant. Achieving small test error in mildly wide ReLU Neural Networks. Arxiv.

Width + Regularizers ➔ No Bad Local-min!

High-level Proof Sketch

Step 1: Local-min with a zero neuron is a global-min.

[Burer–Monteiro MP 2005, Prop. 4] (informal version):
If a rank-r local minimizer remains locally optimal after appending a zero column, then it is a global minimizer. (for SDP)

Our lemma for the two-layer neural-nets has a bit similar flavor:

Lemma **[Liang-Sun-Srikant, 2021]**
A stationary point of a one-hidden-layer neural-net problem with one neuron's connections being all zero is global minimizer.

Step 2 (next page): Any stationary point has at least one zero neuron.

Proof Key: Solvability of Systems of Equations

Step 2 (next page): Any stationary point has at least one zero neuron.

Underlying math problem:

Solvability of "over-determined" systems of nonlinear equations.

- At least one zero neuron $\Longleftarrow$ $\boldsymbol{F(v) = \lambda}$ **has no solution** for generic λ

  ---F is a mapping from v to the eigenvalues of a matrix parameterized by v.

Major trick: Proper dim counting: # of samples = dim(v) < dim(λ) = # of neurons
(after splitting into finitely many systems)

Lemma 1: If $F: \Re^n \to \Re^m, m > n$ is Lipschitz, then $F(\Re^n)$ has zero measure in $\Re^m$.

Lemma 2 (Lidskii 1950): The mapping from the elements of a symmetric matrix to its ordered eigenvalues is Lipschitz.

Side: Comparison with Low-Rank Matrix Completion

Low-rank Matrix Completion Problem: $\min_{X\in\mathbb{R}^{n\times r}, Y\in\mathbb{R}^{n\times r}} \frac{1}{2}\|\mathcal{P}_\Omega(M - XY^T)\|_F^2$

can be viewed as a nonlinear two-layer network.

People say "matrix completion problem has no bad local-min" (under certain conditions).
The precise version is "regularized matrix completion problem has no bad local-min" .
For a certain variant of matrix completion, bad local-min exist.

regularization seems necessary to eliminate local-min, in NNs and matrix completion (both nonlinear).

	Original Objective	Regularized Objective
Low-rank Matrix Completion	Local min can exist [Zhang-Sojoudi-Lavaei, JMLR 19]	No sub-optimal local-min [Sun-Luo, FOCS 15]: [Ge-Lee-Ma, NIPS 15], [Ge-Jin-Zheng, ICML 17]
1-Hidden-Layer Neural Net (ReQU neuron)	Local min can exist [Ding-Li-Sun, MOR 20]	No sub-optimal local-min [Liang-Sun-Srikant, SIOPT 21]

Optimism on Training NNs

Some Reviewers around 2016-2020: Why analyzing NN's optima?
It is well known that over-parameterized NNs can be easily trained to optima.

What matters is to prove over-parameterized networks can generalize well.

Reflect the opinions of many ML theory researchers at that time.

Ruoyu: We still need some rigorous results to better understand NN optimization.
Plus, it helps us understand why training small nets is soooo hard.

Reviewers: I don't care.

Some Reviewers around 2016-2020: Why need a landscape result?
NTK results already prove GD converges to global minima for wide networks.

Ruoyu: They are nice results, but:
i) They require a huge width that is not practical; that's for ultra-wide nets.
ii) They require iterates to move little, which do not reflect the true trajectory.

Reviewers: I don't care.

Are NNs Trained to Global Minima?

I gave a talk at a CV conference in May 2026.

A professor asked: Why do you think NNs are trained to global optima?

Yesterday, I asked a bunch of LLM company pre-training team young authors:

do you think NNs are trained to global optima?

Their first reflection: Why do you think NNs are trained to global optima?

Let's pause a bit:

Why there was an optimism that NNs are trained to global optima, in, say, 2015-2021? (before LLMs become popular)

Why (Many) People Thought NNs were Trained to Global Optima?

My guess (correct me if wrong):

Reason 1: LeCun, Bengio, and many ML researchers said so; and… with "theoretical evidence".

Reason 2: On ImageNet classification (main task of discussion),
top-5 accuracy > 98%;
top-1 accuracy > 90% (after, say, 2018), close to absolute lower bound.
Thus, it seems fair to say: achieved approximate global minima.

Reason 3: On ImageNet classification (main task of discussion),
different initialization lead to similar error.
[mode connectivity is built on this]

Do People Still Believe NNs were Trained to Global Optima?

- Nowdays, the main task of discussion is NO longer ImageNet classification with a size-50m ResNet.

- Instead, the main task of discussion: pre-training LLMs.
- To be more precise: not just 1B LLM, but also **SoTA LLMs (1T-10T size)**

whose architecture is NOT even known to public.

Are these LLMs trained to global optima?

I don't know. I guess nobody (at least nobody in this auditorium) knows.

---perhaps they are NOT trained to fully converge yet.

Are these LLMs over-parameterized? Hard to say…

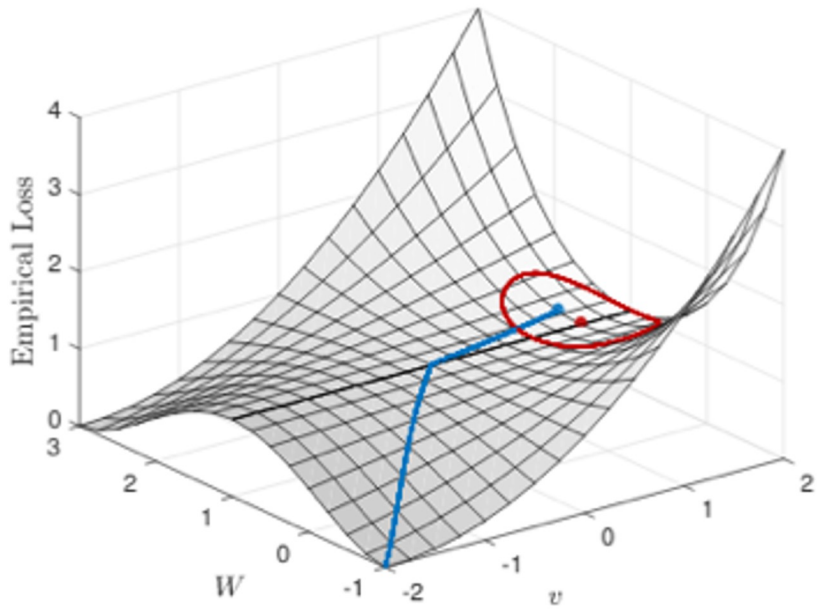
One day, if the LLMs can be easily trained to converge,
then the question of "NNs can be trained to global optima" can be tested again.

Do People Still Believe NNs were Trained to Global Optima?

- So…
- The previous counter-arguments largely no longer hold
- so… I believe, this line of fundamental research is still of interest.

Part 1.3 Symmetry and Landscape

A Basic Geometric Unit Rooted in Symmetry



This geometrical “unit” is helpful for thinking about the landscape

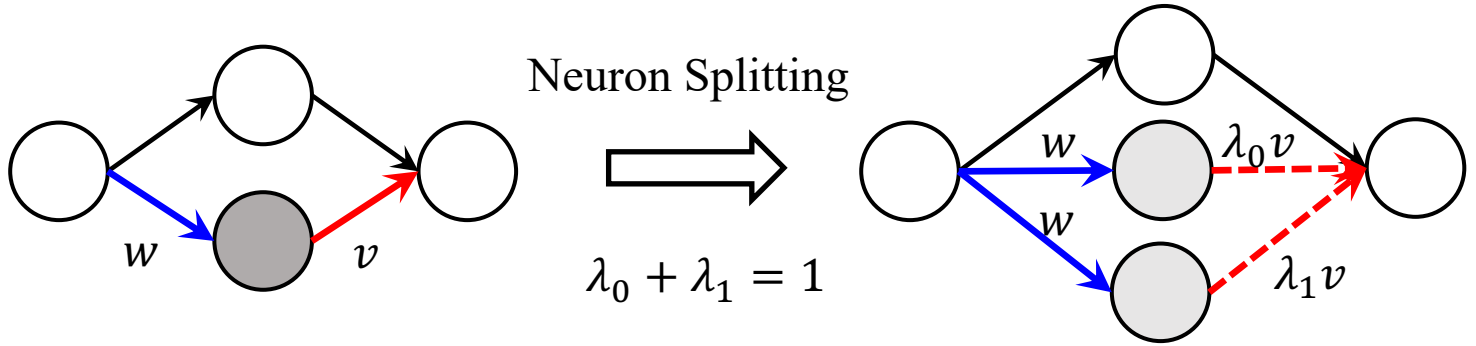
- Bad local minima exist
- Bad local minima are “non-attracting” plateaus
- perturbed decreasing path still exist

Next: This “unit” is not just a toy example.
It is a common component in NN landscape, rooted in symmetry.

Understanding its root and structure can deepen our understanding of landscape.

Evolution Mechanism

Neuron splitting: "Split" a hidden neuron into multiple copies

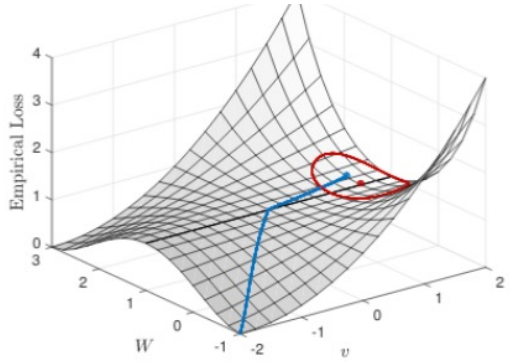


- **Fan-in weights**: copy from origin fan-in weights
- **Fan-out weights**: spread over all new neurons (but sum up to the original fan-out weights)
- **Preserve the input-output mapping**

[Fukumizu-Amari 00], [Zhang et al. 21], [Simsek et al.'21]:

Neuron splitting maps a stationary point to a stationary plateau.

[Fukumizu-Amari 00]: Under certain conditions, the stationary point is mapped to a local-min-saddle line.

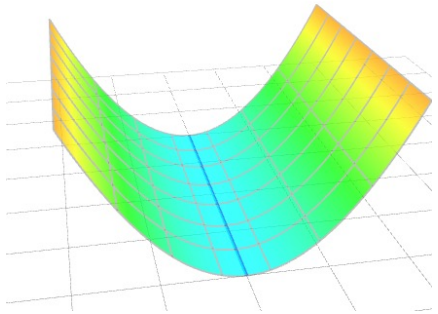


The black line is a local-min-saddle line

Three Possible Stationary Plateaus

A stationary point

?

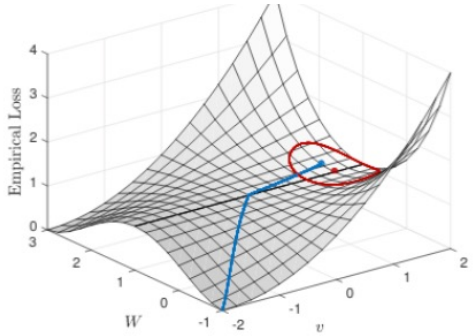


All-local-min plateau

Strong attractors

Very bad

?

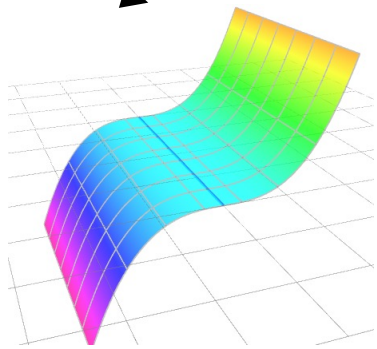


Local-min-saddle plateau

Weak attractors

Somewhat good

?



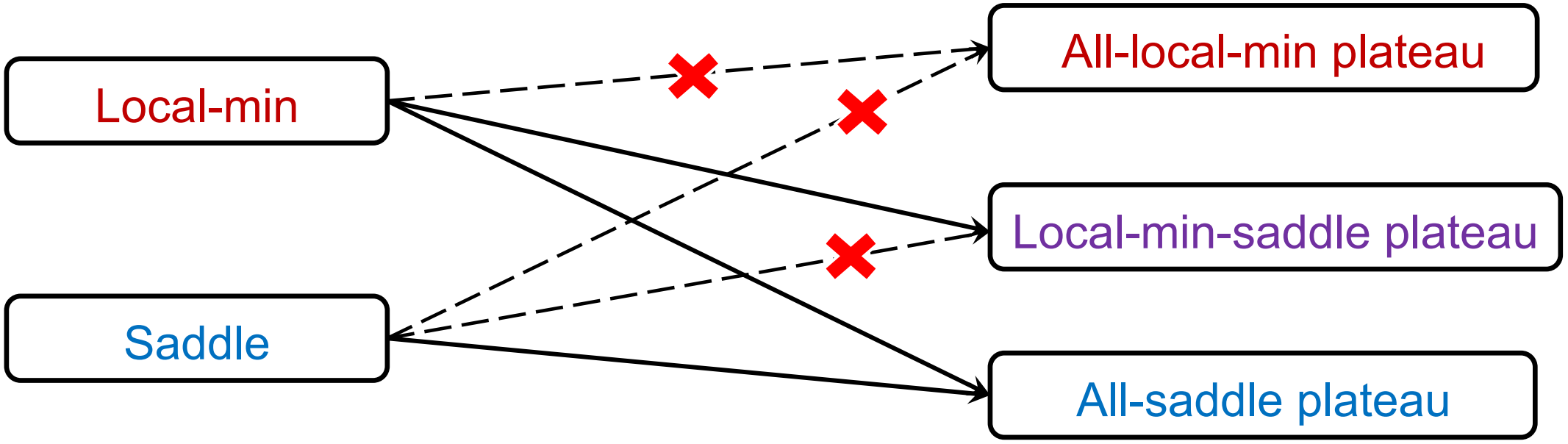
All-saddle plateau

Non-attractors

Good

Evolution of Stationary Points

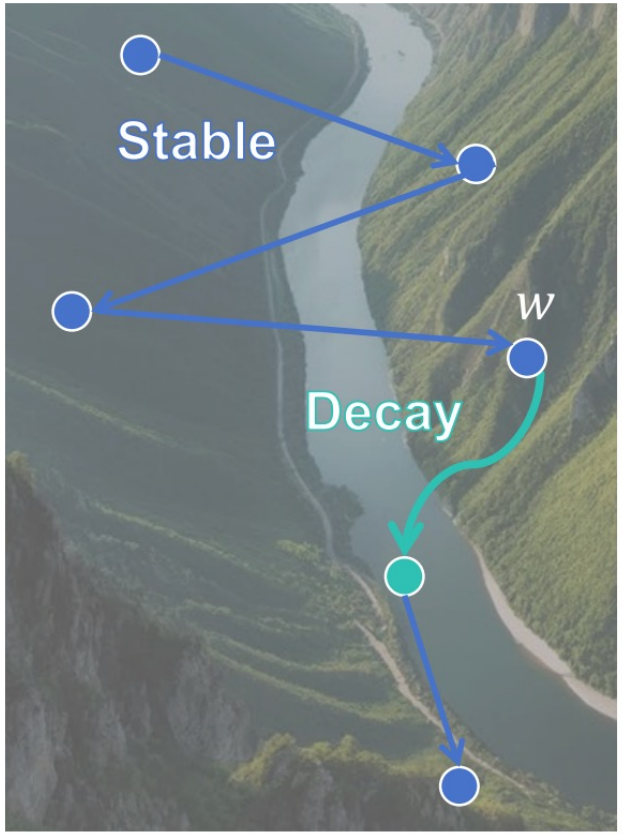
- [Ding-Li-Sun 26]: Classify the stationary points on the plateau obtained by neuron splitting.
  - Based on: Inner Hessian (PD, ND, ID) & fan-out coefficients $\boldsymbol{\lambda}$ (local-min region in Λ_{PD} or Λ_{ND}).



Reference: A Geometric Characterization of the Stationary Plateau for Two-Layer Neural Networks. Tian Ding, Dawei Li, Ruoyu Sun. Arxiv.

Part 1.4 Discussions

“River-Valley Landscape”

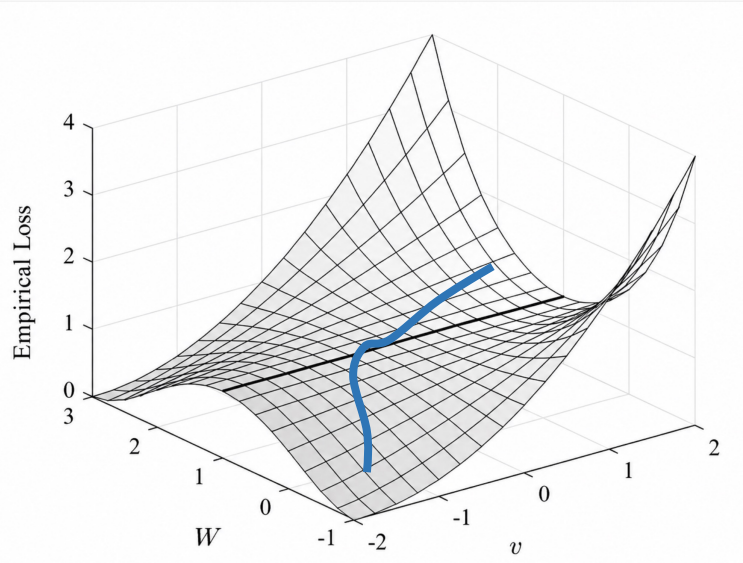


(a) River Valley Landscape

[Wen et al., ICLR 2025 Oral] made a bold conjecture:
LLM pretraining loss has a river valley landscape,
where the loss is like a “deep valley with a river at its bottom”.

The “river valley” resembles
our geometrical unit.

Its proposed trajectory somewhat
resembles the trajectory on the right.



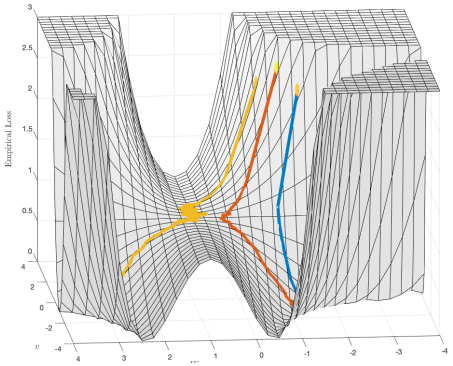
Will the trajectory be that simple?
Unlikely.

But why does this paper explain some LLM phenomenon?
Mysterious.

A Story on Why NNs ”Easy” to Train

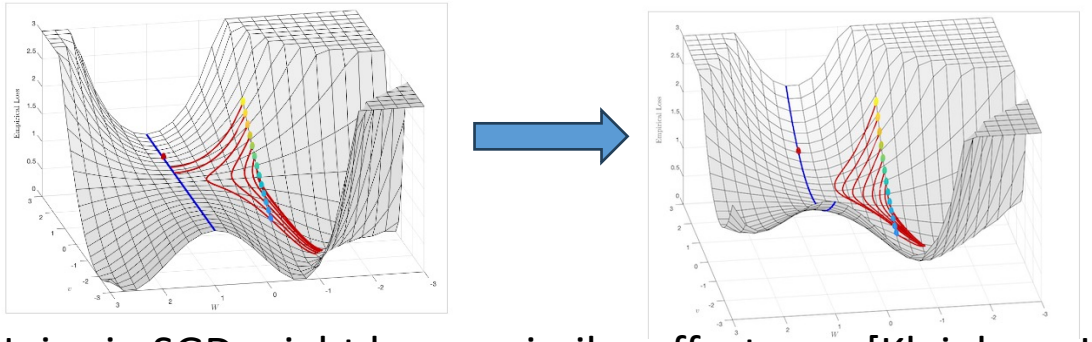
Reason 1 (landscape)

Wide NN’s landscape does NOT contain basins.

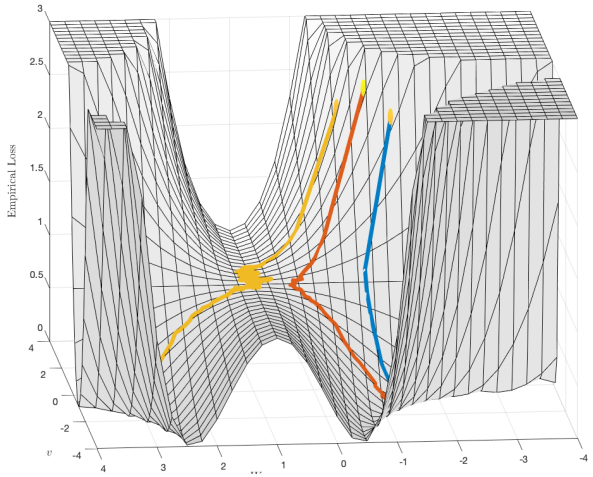


Reason 2 (landscape):

Regularization tilted the landscape to a nicer one.



Remark: Noise in SGD might have a similar effect; e.g. [Kleinberg-Li-Yuan, ICML 18]



Reason 3 (algorithmic & landscape):

Proper choice of initialization (avoids being close to plateaus).

Reason 4 (algorithmic and/or landscape): Training can avoid plateaus.

--Highly open!

--Result with somewhat related flavor: [Cohen et al. ICLR’25] ”region-avoiding” behavior of GD

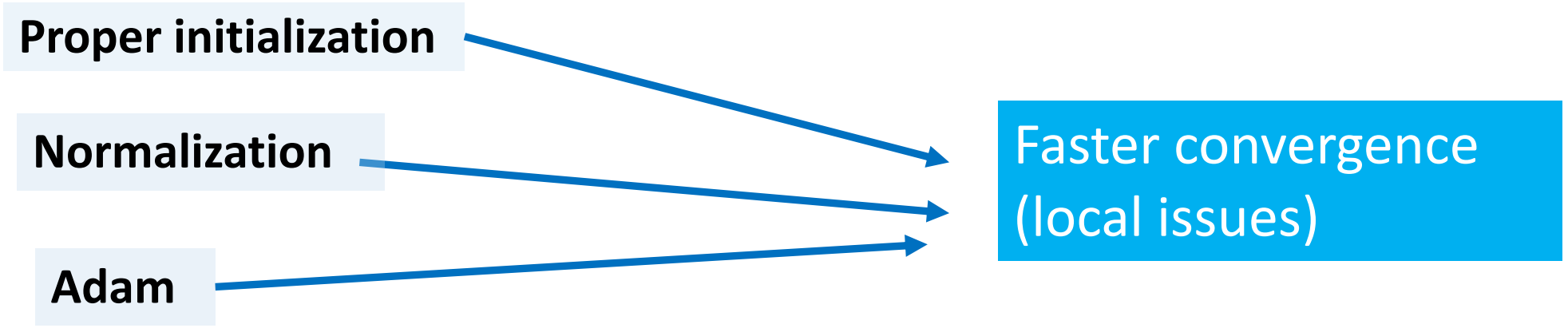
Part 2
Local Structure and Algorithms

Faster Convergence

For most algorithm designers, global landscape is not a big issue.
[for pioneers like Minsky, LeCun, Bottou, it was be a big issue]

Survival bias: If local minima present significant threat,
then the problem may not be on the radar of most algorithm designers.
e.g. min-max was rarely used with NNs, until GAN worked in 2014.

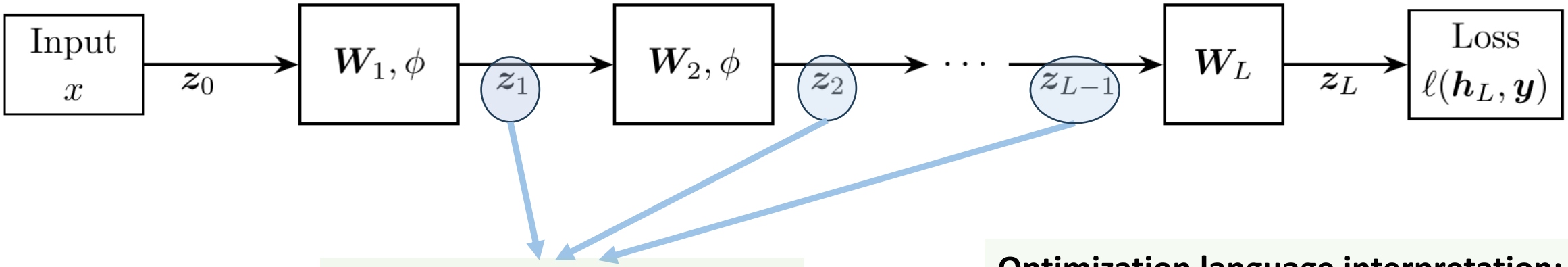
Next, we discuss faster convergence in NN optimization.



Reference: Optimization for deep learning: an overview, Ruoyu Sun, JORSC, 2020.

Disclaimer: They may help generalization and representation, but we focus on optimization.

Initialization & Normalization



"Signal Propagation" (SP)

Hope $\| z_j \| \approx 1$

Optimization language interpretation:

smaller Lipschitz constants

e.g. [Santurkar et al. NeurIPS 18]

or smaller condition number

e.g. [Pleser-Widmann-Bach, NeurIPS 20]

Initialization methods:

Bottou-LeCun initialization

Xavier initialization

Kaiming initialization

…

Normalization methods:

Layer normalization (transformers)

Batch normalization (image classification)

Group normalization (object classification)

…

Adam

- **Adam** has been the most popular algorithms in deep learning (DL).
  (>240k citations, by May 2026)
- **Popular in** RL, LLM (large language models), Diffusion models, VLA,…

```
optimizer = optim.Adam(net.parameters(), lr=args.lr, betas=(args.beta1, args.beta2), eps=1e-08,
                       weight_decay=args.weightdecay, amsgrad=False)
```

(for LLMs, recently challenged by muon)

- Understanding Adam is not easy.
- It requires algorithmic analysis & NN structure analysis.

Criticism on Adam

- [Wilson et al.'2017] "Marginal values of adaptive gradient methods" claimed:

well-tuned SGD (with momentum) outperforms Adam

- Criticism of Adam [Wilson2017] did not stop Adam. Why?
- They only run experiments on Image Classification!

("main task of discussion", for theoreticians)

-- I asked: who used Adam? Check top papers citing Adam paper.

10 top cited papers in 2017:
4 GANs,3 NLP,1 RL,1 GNN.

Finding:
people use Adam in GAN, RL, NLP, etc;
not in image classification

Part 2.1: Does Adam converge

Definition of SGD and Adam

Consider a finite-sum optimization problem

$$\min_{\theta} F(\theta) \coloneqq \frac{1}{n}\sum_{i=1}^{n} F_i(\theta)$$

SGD (Stochastic Gradient Descent): at t-th iteration, randomly pick index i, then update

$$\theta_{t+1} = \theta_t - \alpha_t \nabla F_i(\theta_t).$$

Adam: At t-th iteration:

randomly pick index i_t, compute $g_t = \nabla F_{i_t}(\theta_t)$, then update

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad \text{[update 1st order momentum]}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t \circ g_t, \qquad \text{[update “2nd order” momentum]}$$

$$\theta_{t+1} = \theta_t - \alpha_t \frac{m_t}{\sqrt{v_t} + \epsilon} \qquad \text{[update parameter]}$$

Question: Does Adam Converge?

- [Reddi et al.'2018] (ICLR'18 outstanding paper) shows "Adam can be divergent"

- **Conceptual Question**:

  Why does Adam work well, though possibly "divergent"?

- Maybe NN problems are special?

  Or is something missing in the theory?

Strong Assumptions in Many Adam Results

Commonly used assumptions in literature: A1 – A4

A1 (L-smooth): assume $\nabla \ell_i(\theta)$ are L-Lipschitz continuous

A2 (bounded Variance): $\frac{1}{n}\sum_{i=1}^{n} || \nabla \ell_i(\theta) - \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(\theta) ||_2^2 \leq D_0$

A3 (bounded gradient): $|| \nabla \ell(\theta)|| < C$

A4 (bounded 2nd order momentum): $v_k \in [C_l, C_u]$

[Reddi et al.18, Luo et al.19, Chen et al.18, Zou et al. 19, Chen et al. 22…]: proposed and studied variants of Adam
[Defossez et al. 2022] studied Adam: but A3 and A4 **avoid divergence a priori**, **reduced to AdaBound**

Standard Assumptions as SGD

We **removed A3 & A4** and **relaxed A2**

A1 (L-smooth): assume $\nabla \ell_i(\theta)$ are L-Lipschitz continuous

A2 (Affine Variance): $\frac{1}{n}\sum_{i=1}^{n} || \nabla \ell_i(\theta) - \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(\theta) ||_2^2 \leq D_1 || \nabla \ell(\theta)||_2^2 + D_0$

~~**A3 (bounded gradient):** $|| \nabla \ell(\theta)|| < C$~~

~~**A4 (bounded 2nd order momentum):** $v_k \in [C_l, C_u]$~~

[Reddi et al.18, Luo et al.19, Chen et al.18, Zou et al. 19, Chen et al. 22…]: proposed and studied variants of Adam
[Defossez et al. 2022] studied Adam: but A3 and A4 **avoid divergence a priori**, **reduced to AdaBound**

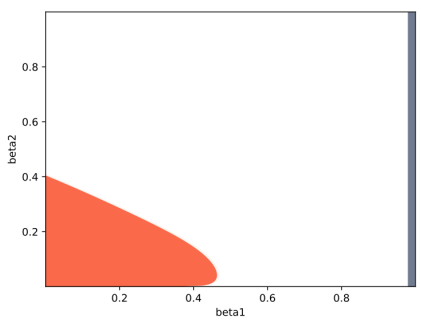
- A1 and A2 do not exclude Reddi's counterexample a priori
- A2 is weaker than bounded variance condition ($D_1 = 0$), mildest variance condition so far
- For Adam, $|| \nabla \ell(\theta)||$ can diverge as θ diverges: **it is important to remove $|| \nabla \ell(\theta)|| < C$.**

Phase Transition: Converge → Diverge

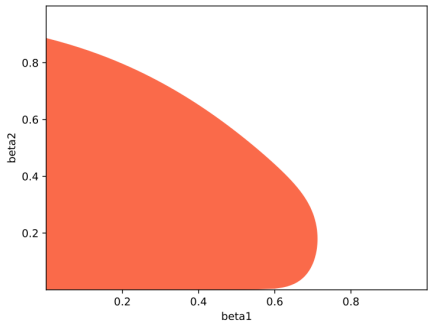
Thm 2.1: when $\beta_2 \geq 1 - O\left(\frac{1-\beta_1^n}{n^{3.5}}\right)$, $\beta_1 < \sqrt{\beta_2} < 1$, Adam converges with rate $O(\frac{\log k}{\sqrt{k}})$

Remark: When $D_0 > 0$: converges to a neighborhood of stationary points with size proportional to D_0.
When $D_0 = 0$ (strong growth condition holds), converges to stationary points.

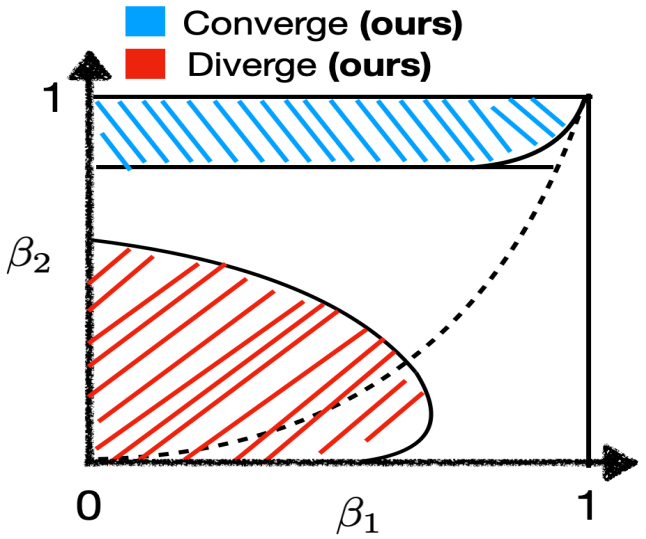
Thm 2.2: for $\forall n \geq 3$, $\exists \ell(\theta)$ in the same problem class as above, when (β_1, β_2) lies in the red region (also dependent on n), sequence of $\{\theta_k\}$ and $\{\ell(\theta_k)\}$ of Adam diverges to ∞



(b) $n = 10$



(d) $n = 100$



Reconcile Our Result and Divergence Result

[Reddi et al.'18]: for any $\beta_2 \in (0,1)$, ∃ a problem that Adam diverges.

Our work: for any problem, for large enough β_2, Adam converges

One sentence message:

Adam can converge with proper hyperparameters

Empirical Advice on Tuning Adam

- **Empirical advice**:

  Tune β_2 larger if performance bad.

  Tune β_2 larger for larger noise (e.g. smaller batch size)

- Although simple, many practitioners do NOT know this!

- Try GAN with Pytorch default β_2=0.9, very bad performance!

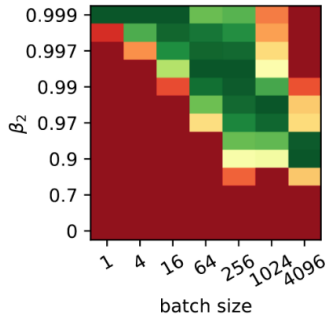
  --With the advice: try $\beta_2 = 0.99$ (not 0.8)

Feedback from LLM developers

Recent findings by Harvard & Amazon:
"larger β_2 (increase from 0.95 to 0.99, 0.999, or 0.9995) substantially improves small batch size training… aligning with findings in [**Zhang et al., 2022]**"

Recent findings by Yair Carmon and coauthors:
" …we find that tuning (enlarging) the β_2 (from 0.95 to 0.99 and 0.999) is essential at lower batch sizes… This matches the theoretical work by **[Zhang et al. 2022]** that showed that when the batch size is small, higher values of β2 are crucial"

Recent findings by Max Planck Institute: "**Zhang et al. [2022]** shows that higher β_2 substantially improve small-batch training, and Marek et al. [2025] highlights the importance of scaling β_2 in this regime."



[Figure from Marek et al. 25]

H. Zhang, et al. HOW DOES CRITICAL BATCH SIZE SCALE IN PRETRAINING?, ICLR 2025
Srećković, et al. Is your batch size the problem? revisiting the adam-sgd gap in language modeling, preprint
Porian, T et al., Resolving discrepancies in compute-optimal scaling of language models, NeurIPS 2024

References for this part

Adam Can Converge Without Any Modification On Update Rules

Yushun Zhang[13], **Congliang Chen**[1], **Naichen Shi**[2], **Ruoyu Sun**[13]*, **Zhi-Quan Luo**[13]
[1]The Chinese University of Hong Kong, Shenzhen, China [2]University of Michigan, US
[3]Shenzhen Research Institute of Big Data
{yushunzhang,congliangchen}@link.cuhk.edu.cn, naichens@umich.edu
{sunruoyu,luozq}@cuhk.edu.cn

Adam Converges Without Any Modification On Update Rules

Yushun Zhang[12], Bingran Li[1], Congliang Chen[1], Zhi-Quan Luo[12], Ruoyu Sun[12†]

[1]The Chinese University of Hong Kong, Shenzhen, China
[2]Shenzhen Research Institute of Big Data
{yushunzhang,bingranli,congliangchen}@link.cuhk.edu.cn,
{sunruoyu,luozq}@cuhk.edu.cn

Conference version:

- Appeared online in Aug 2022
- NeurIPS 2022 **(Spotlight)**

Extended journal version:

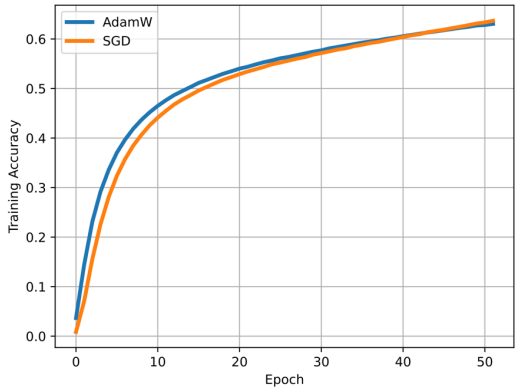
- Appeared online in March 2026
- Under review

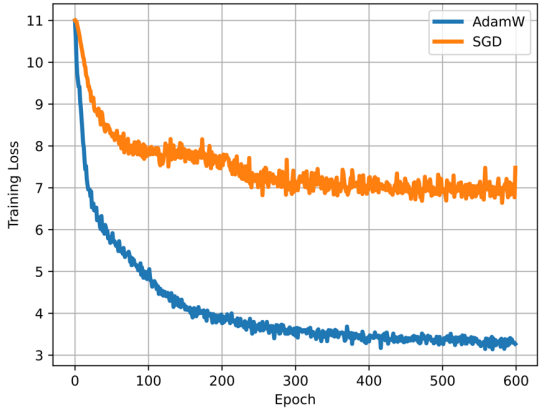
New contents:

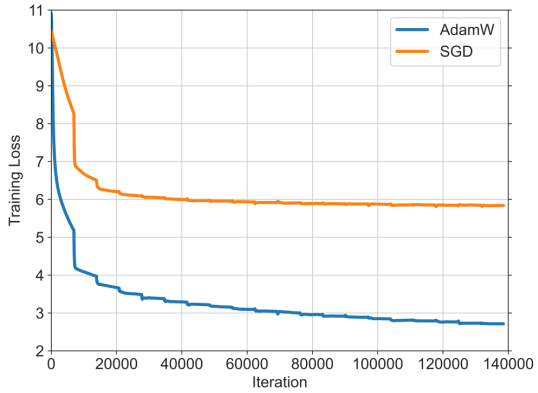
- A simplified proof under random shuffling
- A new proof under with-replacement sampling

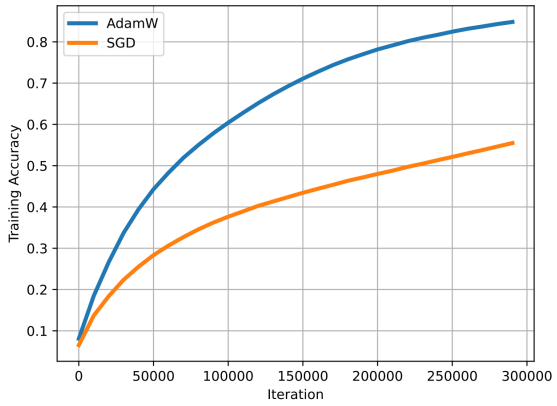
Part 2.2 Why Transformers need Adam: explanation based on NN's Hessian Structure

SGD works well on CNN, largely underperforms Adam on Transformers









VGG (CNN)

GPT2

BERT

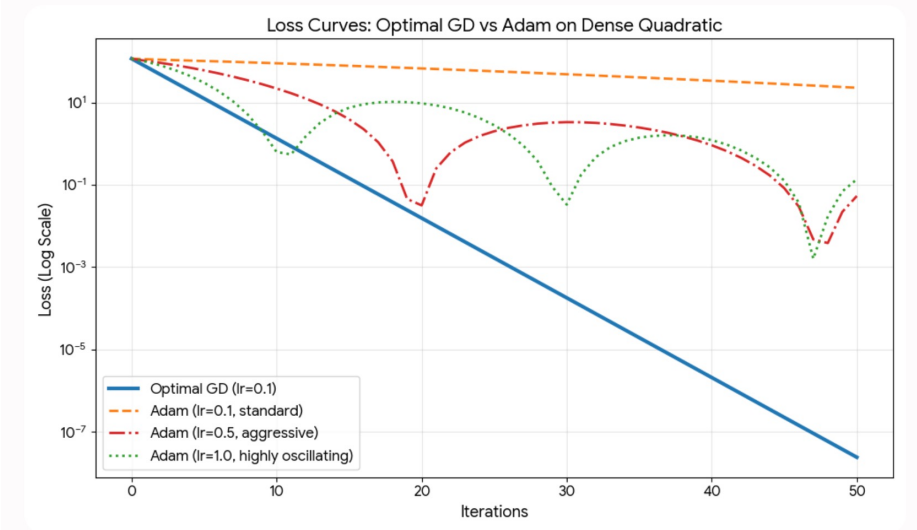
ViT

Transformers Need Adam, but why?

[Zhang et al. 19] attributed to heavy-tail noise;
but [Chen-Kunstner-Schmidt'21] shows the gap exists in full-batch setting too, so noise is not enough.

GD Beats Adam on Random Quadratic Functions

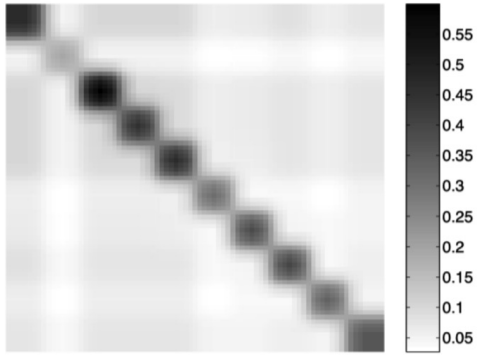
Even for convex quadratic minimization $min_w\ w^T Aw,$ GD can easily beat Adam.
Here is an example, where A is a randomly generated symmetric matrix.



This seems mysterious:
If Adam cannot beat GD on quadratics, why can it beat GD on complicated NNs?

Answer: NNs have **special structure** that "**random**" quadratic problems do not have: **near block diagonal Hessian**.

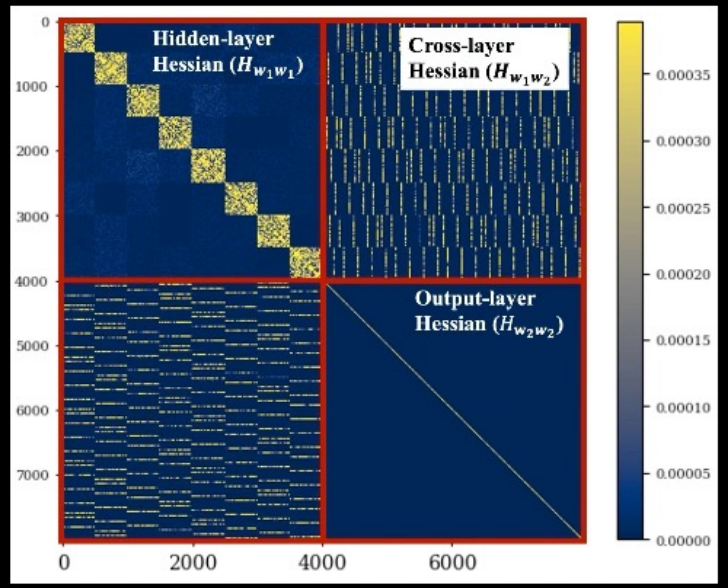
Structure: Hessian of NN is Nearly Block Diagonal

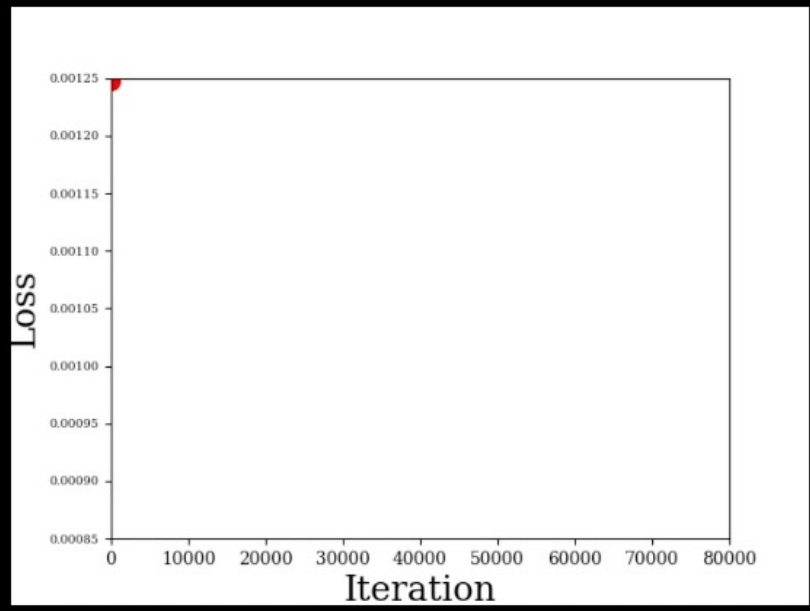


(a) Hessian of an MLP [18] after 1 step

[Fig. from: Collobert 2004]

Hessian of a 2-layer relu NN, input dim = # classes = 500, width = 8, CE loss +Adam, Gaussian data + random label, sample size = 5000

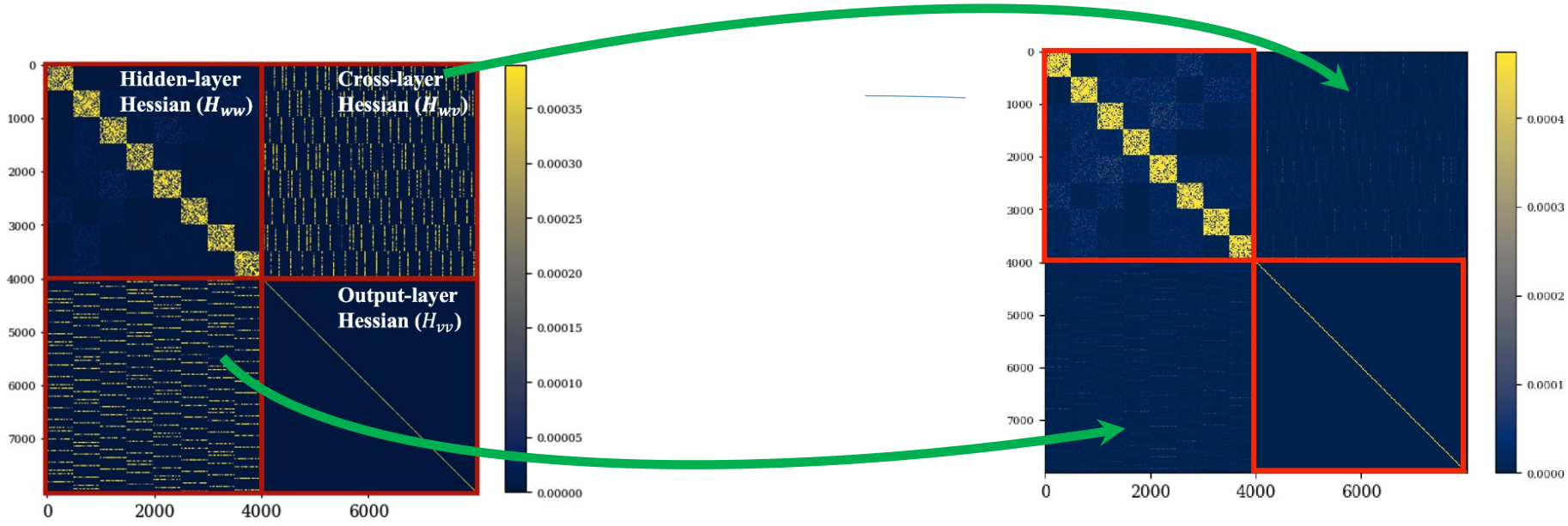




In an early stage of training, the Hessian becomes nearly block diagonal.

One block = one output neuron (or one row in a weight matrix)

Two forces that shape the Hessian structure



(a) Hessian at initialization

(f) Hessian at 100% steps

Force I: a ``**dynamic force**'' arisen from training;
off-diagonal parts decreases as prediction error decreases.
Force II: a ``**static force**'' rooted in the architecture design (e.g., large # Class C);
next page.

Dong*, Zhang*, Yao, **Sun**; Towards Quantifying the Hessian Structure of Neural Networks, arxiv, 2025.

Theoretical Results: Two-Layer NN at Initialization

Dong*, Zhang*, Yao, **Sun**; Towards Quantifying the Hessian Structure of Neural Networks, arxiv, 2025.

Assumption 1 *The entries of the data matrix* $X_N = (x_1, \cdots, x_N) \in \mathbb{R}^{d\times N}$ *are i.i.d.* $\mathcal{N}(0,1)$.

Assumption 2 *The model weights in* W *and* V *are initialized by LeCun initialization. That is: for the linear model,* $V_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{d})$, $i \in [C], j \in [d]$; *for 1-hidden-layer network,* $W_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{d})$, $i \in [m], j \in [d]$, $V_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{m})$, $i \in [C], j \in [m]$.

- A1 is restricted
- A2 is widely used

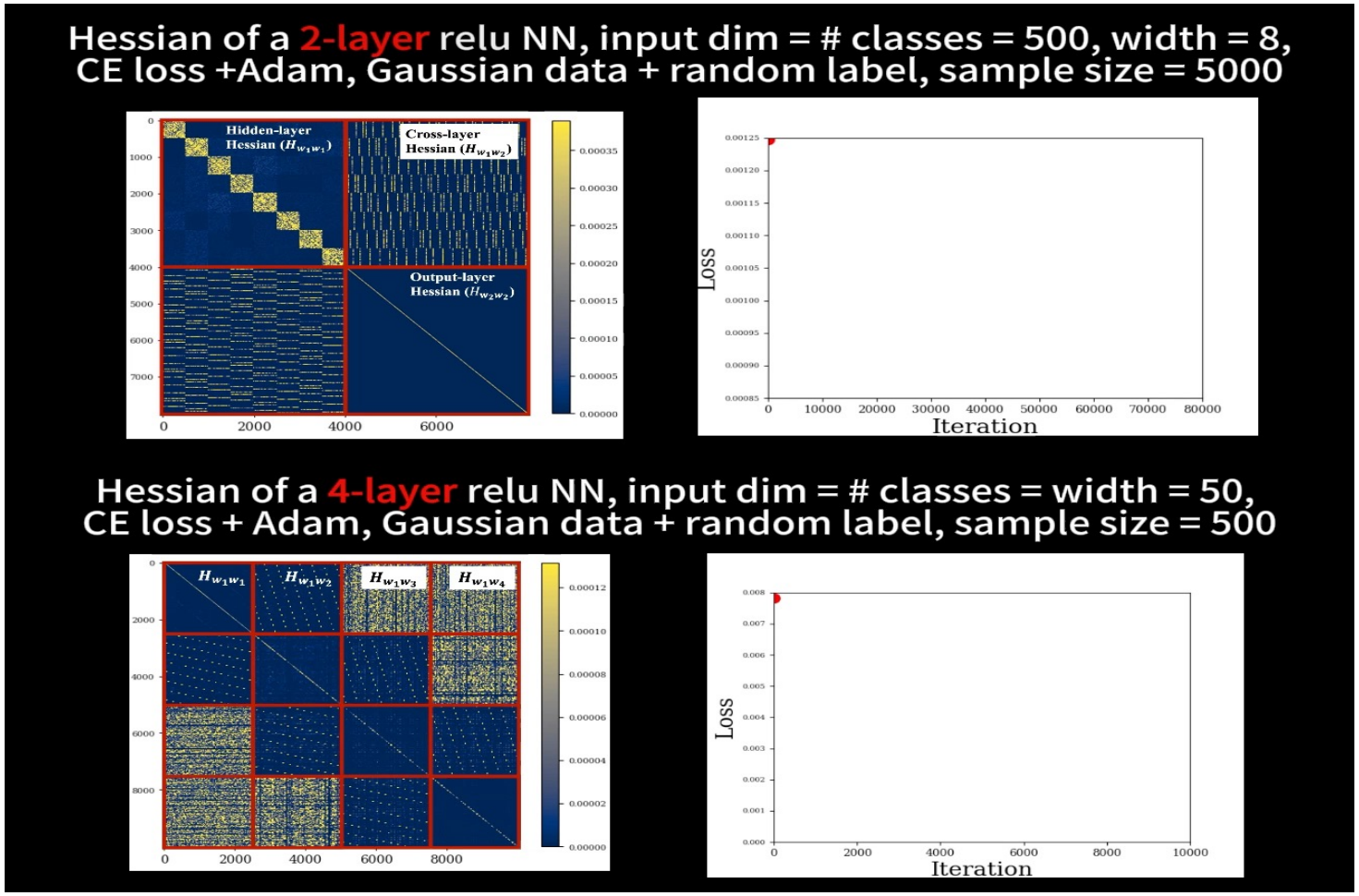
Theorem 2 **(1-hidden-layer networks.)** *Consider the Hessian expressions in (8) to (13), and assume Assumptions 1 and 2 hold. Then for any fixed* $m \geq 3$, *suppose* $d, N \to \infty$, $\frac{d}{N} \to \gamma \in (0, +\infty)$, *it holds that*

$$\lim_{d,N\to\infty} \frac{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{MSE}}(W,V)}{\partial w_i \partial w_j^\top}\right\|_{\text{F}}^2\right]}{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{MSE}}(W,V)}{\partial w_i \partial w_i^\top}\right\|_{\text{F}}^2\right]}, \quad \lim_{d,N\to\infty} \frac{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{CE}}(W,V)}{\partial v_i \partial v_j^\top}\right\|_{\text{F}}^2\right]}{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{CE}}(W,V)}{\partial v_i \partial v_i^\top}\right\|_{\text{F}}^2\right]} \tag{28}$$

vanish at the rate $\mathcal{O}(1/C)$, $\mathcal{O}(1/C^2)$, respectively, and the block-diagonal structure also emerges as C increases.

Key message:
block-diagonal structure arises when
NN output dim C → ∞

Hessian for Deep NNs?



2-layer NN:
output dimension needs to be large

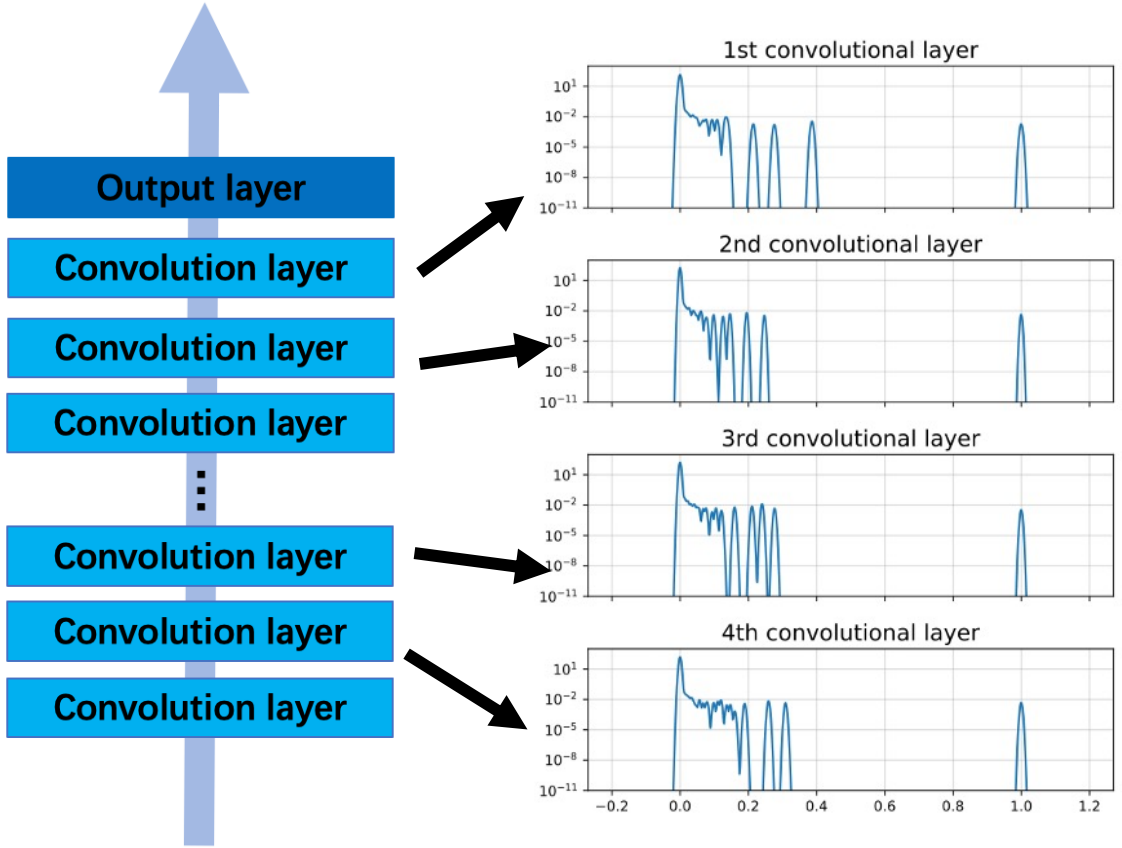
Proof done.

4-layer NN:
output dim of each layer is large;
i.e.
each layer shall be **relatively wide.**

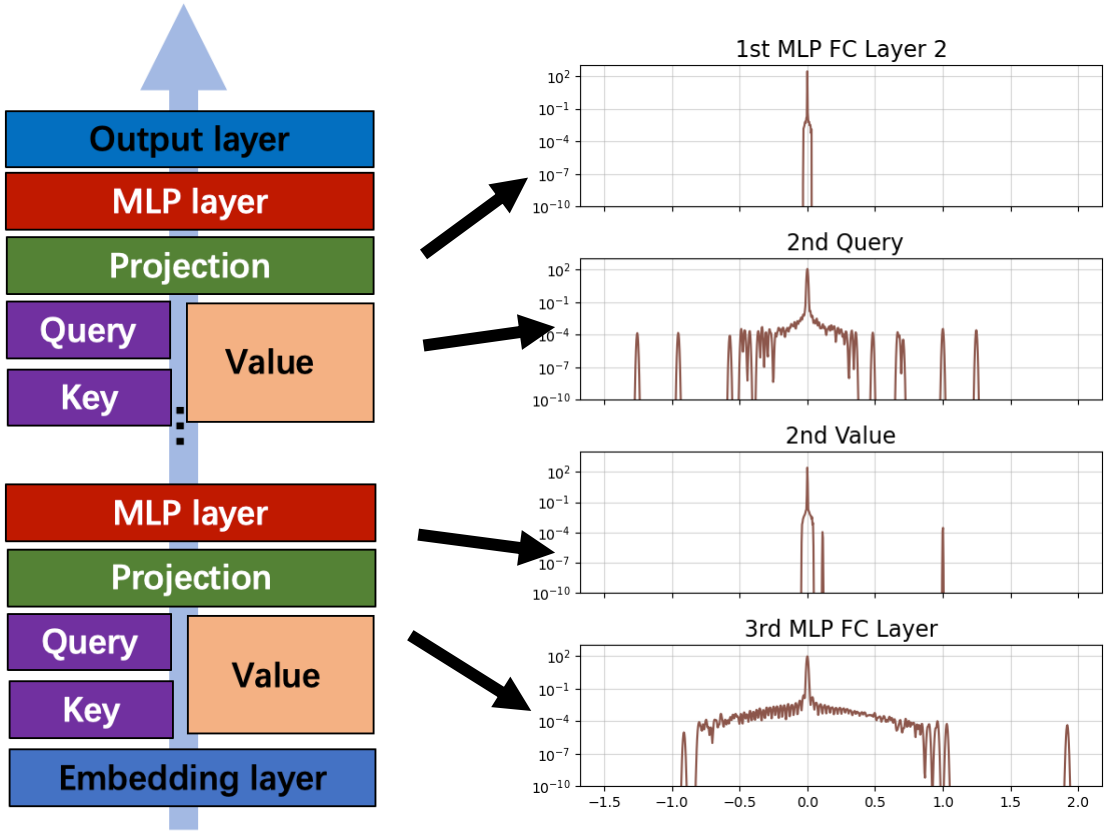
Proof left as future work.

Wide nets help convergence too.

Blockwise Hessian spectrum: CNN and Transformers



ResNet18



BERT

CNNs: blockwise spectrum are quite **similar**
We call it ``**homogeneity**”

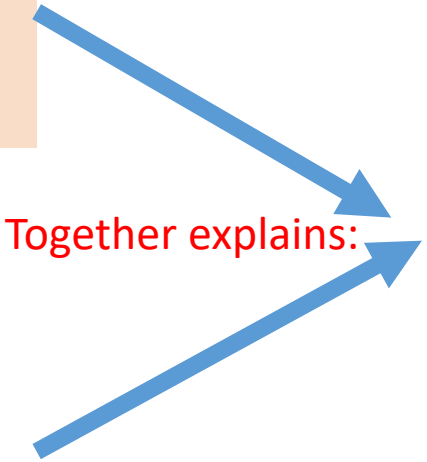
Transformers: blockwise spectrum are largely **different**
We call it ``**heterogeneity**”

Our Explanation: Why Transformer Needs Adam

Zhang, Chen, Ding, Li, **Sun***, & Luo, Why Transformers Need Adam: A Hessian Perspective, NeurIPS 24

Observation 1:
Transformer's block Hessian are heterogeneous.

Observation 2:
If block Hessian are heterogeneous,
then SGD worse than Adam.



Original claim:
Transformer:
SGD worse than Adam.

Remark: [Kunstner et al., NeurIPS 24] attributed to imbalanced data.
Our follow-up experiments indicate that imbalanced data may lead to heterogeneous block Hessian.

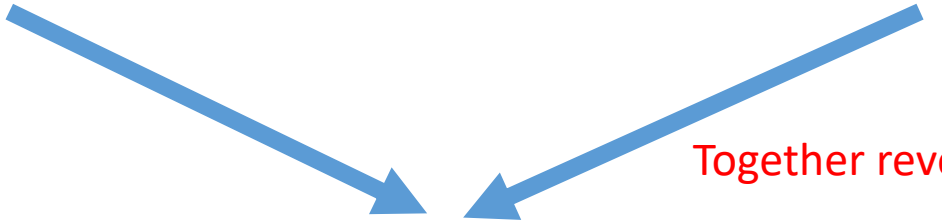
Contents

Part 2.3 Improving Adam and Muon based on Hessian Structure

Further Analysis: Wasted Parameters of Adam

Observation 1:
Adam is good for block-Hessian.

Observation 2:
Adam is bad for dense Hessian



Together reveals:

Implication:
Adam mainly handles "cross-block" challenge,
NOT "within-block" challenge.

Further Implication:
We may replace Adam's coordinate-wise lr by block-wise lr

Adam-mini (Framework)

- **Adam-mini:**

  - **Step 1**: partition the gradient g into B sub-vectors according to the dense Hessian sub-block $g_b, b = [B]$

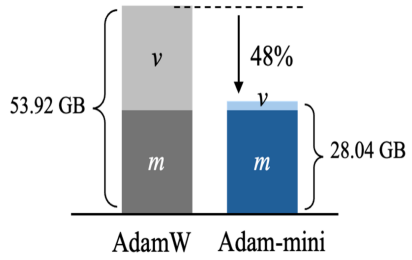
  - **Step 2:** for each g_b, calculate:

$$v_b = (1-\beta_2) * \mathbf{mean}(g_b \circ g_b) + \beta_2 * v_b, \quad b = 1, \cdots B$$

This removes **99.9%** v in Adam, so reduces optimizer state memory by ~ 50%.

- **Step 3:** then use $\frac{\eta}{\sqrt{v_b}}$ as the lr for the parameters associated with g_b

- **Step 1 is important, and will be discussed next**

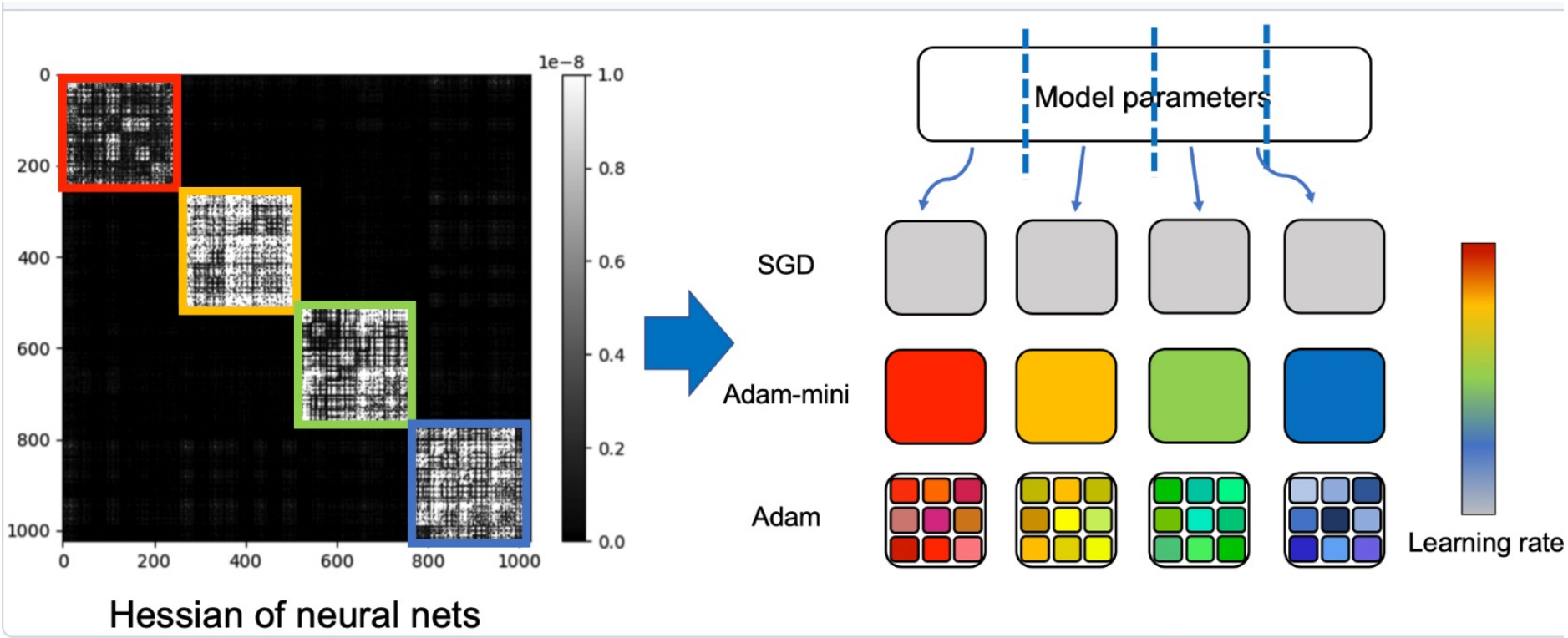


(a) Memory (↓)

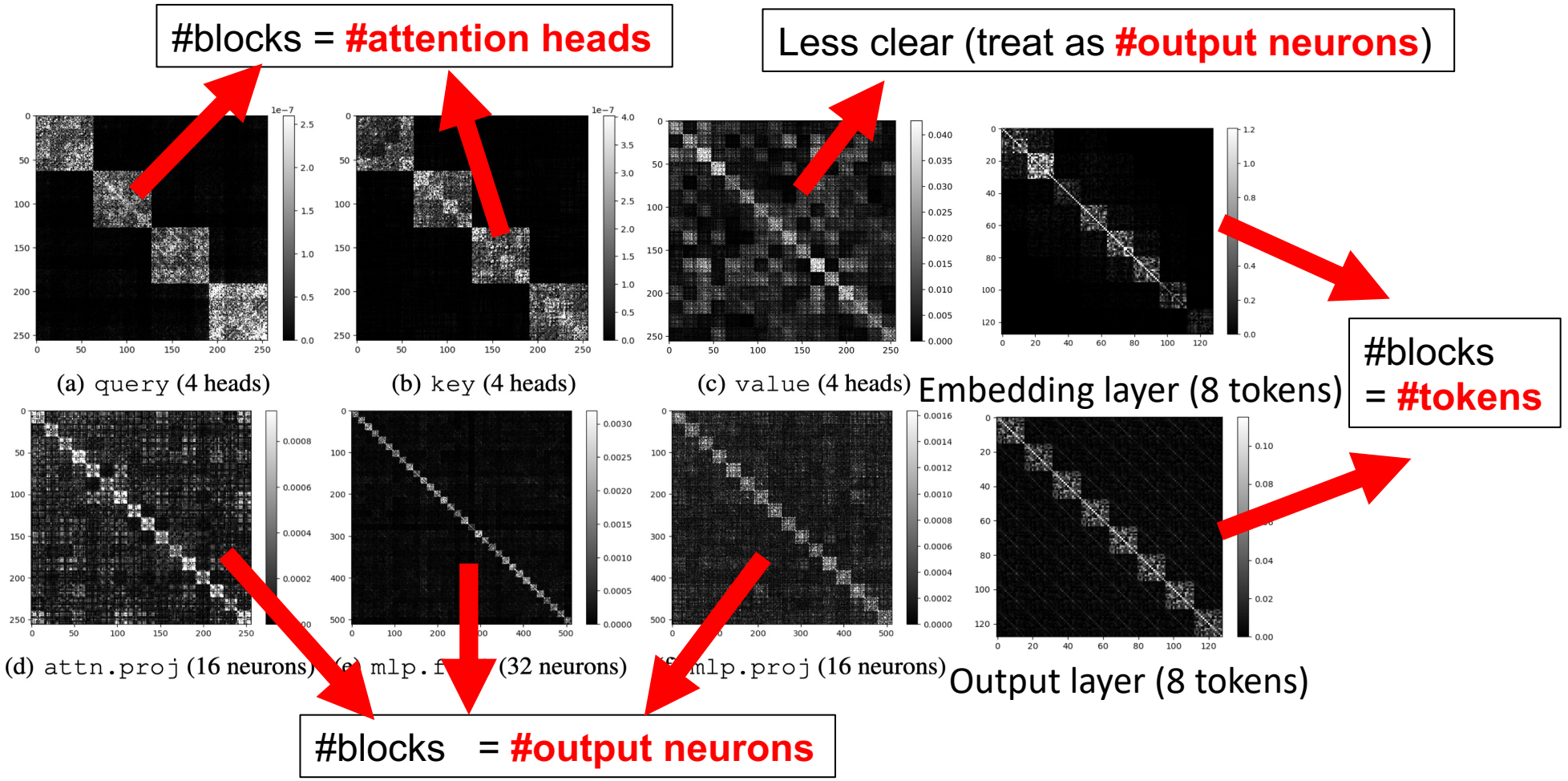
7B model: save ~**50% optimizer state** memory of Adam

Partition Principle

Partition Principle:
Partition parameters into blocks s.t. each block is associated with a dense sub-block in Hessian.



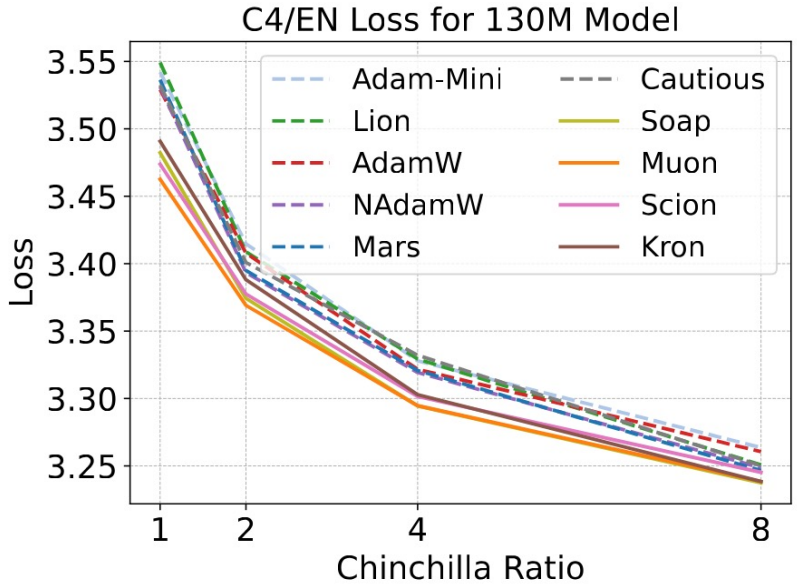
Transformer: Blocks of Hessians

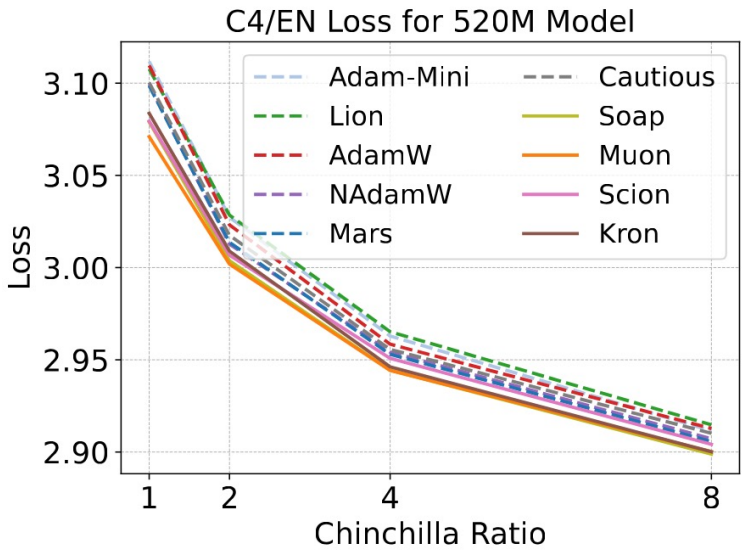


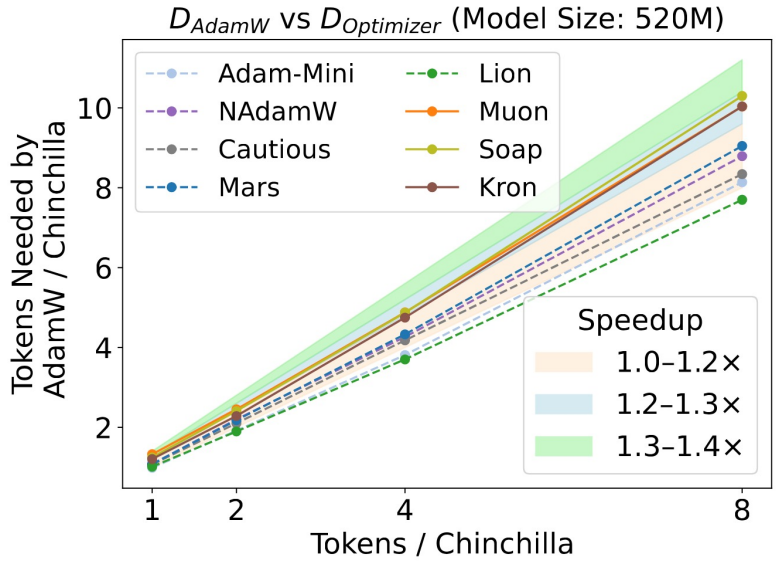
A simplified choice: One block = one output neuron (or one row in a weight matrix)

Adam-Mini Performance in Optimizer Benchmark

- 130/ 300/ 520M model with 1x / 2x / 4x / 8x Chinchilla token amount
- Full hyperparameter sweep for all optimizers
- On 520M model, Adam-mini is slightly faster than AdamW (though designed to be just memory-saving)







"Adam-mini closely tracks the performance of AdamW … and sometimes even performs better than AdamW"

Wen, Hall, Ma, and Liang, (2025). Fantastic pretraining optimizers and where to find them. ICLR 2026.

Muon As Preconditioned Gradient Method

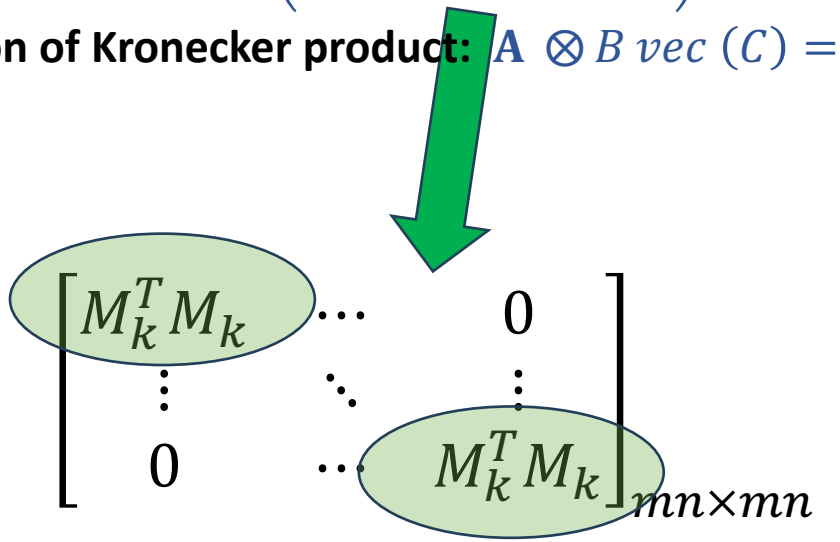
- **Matrix form of Muon:** $W \in R^{m \times n}$

$$W_{k+1} = W_k - \eta_k \left(M_k M_k^T\right)^{-\frac{1}{2}} M_k = W_k - \eta_k \, I_m \, M_k \left(M_k^T M_k\right)^{-\frac{1}{2}}$$

- **Flattened vector form of Muon:**

$$\mathrm{vec}(W_{k+1}) = vec(W_k) - \eta_k \left(I_m \bigotimes \left(M_k^T M_k\right) \right)^{-\frac{1}{2}} vec(M_k)$$

We used the proposition of Kronecker product: $\mathbf{A} \otimes B \; vec\,(C) = ACB$



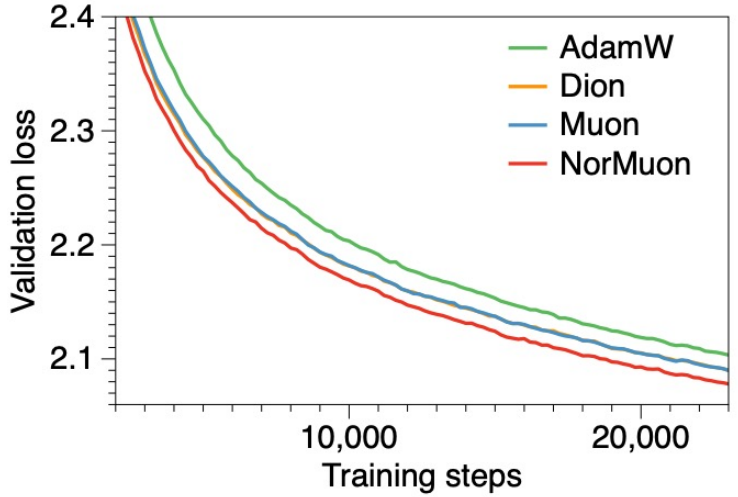
Key observation:
Muon use the **same** preconditioner across neurons in the same layer

But neurons in Transformers are heterogeneous… [1, 2]
How to improve Muon?

[**1] Why Transformers Need Adam: A Hessian Perspective**, NeurIPS 2024
[2] Technical report of Aurora optimizer, May, 2026

NorMuon: Adam-mini + Muon

- **NorMuon:** add neuronwise lr to Muon
- by Microsoft and GaTech, including an author of LoRA



(a) Pretraining results of 1.1B model.

Algorithm 1 NorMuon

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss L, learning rate η, momentum parameters (β_1, β_2), perturbation parameter ε, weight decay λ.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$, $\mathbf{v}_0 \in \mathbb{R}^{m} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta_1 \mathbf{M}_{t-1} + (1-\beta_1)\mathbf{G}_t$
  $\mathbf{O}_t \leftarrow \mathrm{NS5}(\mathbf{M}_t)$
  $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathrm{mean}_{\mathrm{cols}}(\mathbf{O}_t \odot \mathbf{O}_t)$
  $\mathbf{V}_t \leftarrow \mathrm{ExpandRows}(\mathbf{v}_t)$
  $\widehat{\mathbf{O}}_t \leftarrow \mathbf{O}_t \oslash \left(\sqrt{\mathbf{V}_t} + \varepsilon\right)$
  $\hat{\eta} = 0.2\eta\sqrt{mn}/\|\widehat{\mathbf{O}}_t\|_F$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda\mathbf{W}_t - \hat{\eta}\widehat{\mathbf{O}}_t$
end for

NorMuon: Making Muon more efficient and scalable

Zichong Li [*1], Liming Liu [*1 †], Chen Liang [2], Weizhu Chen [2], Tuo Zhao [1]
[1]Georgia Tech [2]Microsoft

This observation motivates our key insight: while Muon's orthogonalization effectively reduces the condition number of updates, the remaining high variance in neuron norms still creates an imbalanced learning dynamic, potentially leading to inefficient parameter usage. Drawing inspiration from Adam-mini's success Zhang et al. (2025) with per-neuron adaptive learning rates, we propose to incorporate second-order momentum to normalize these disparate scales and ensure more balanced parameter updates. Our method, **NorMuon**, augments Muon's orthogonalization with neuron-wise adaptive learning rates computed from accumulated second-order statistics.

Solution: Adam-mini + Muon

NorMuon became the fastest NanoGPT speedrun optimizer, twice. (2025 Nov; 2026 May)

Last tweet by Keller Jordan in May 3rd 2026 (author of Muon, OpenAI):

"Three new modded-NanoGPT optimization benchmark results, **all of them using NorMuon**"

Keller Jordan
@kellerjordan0

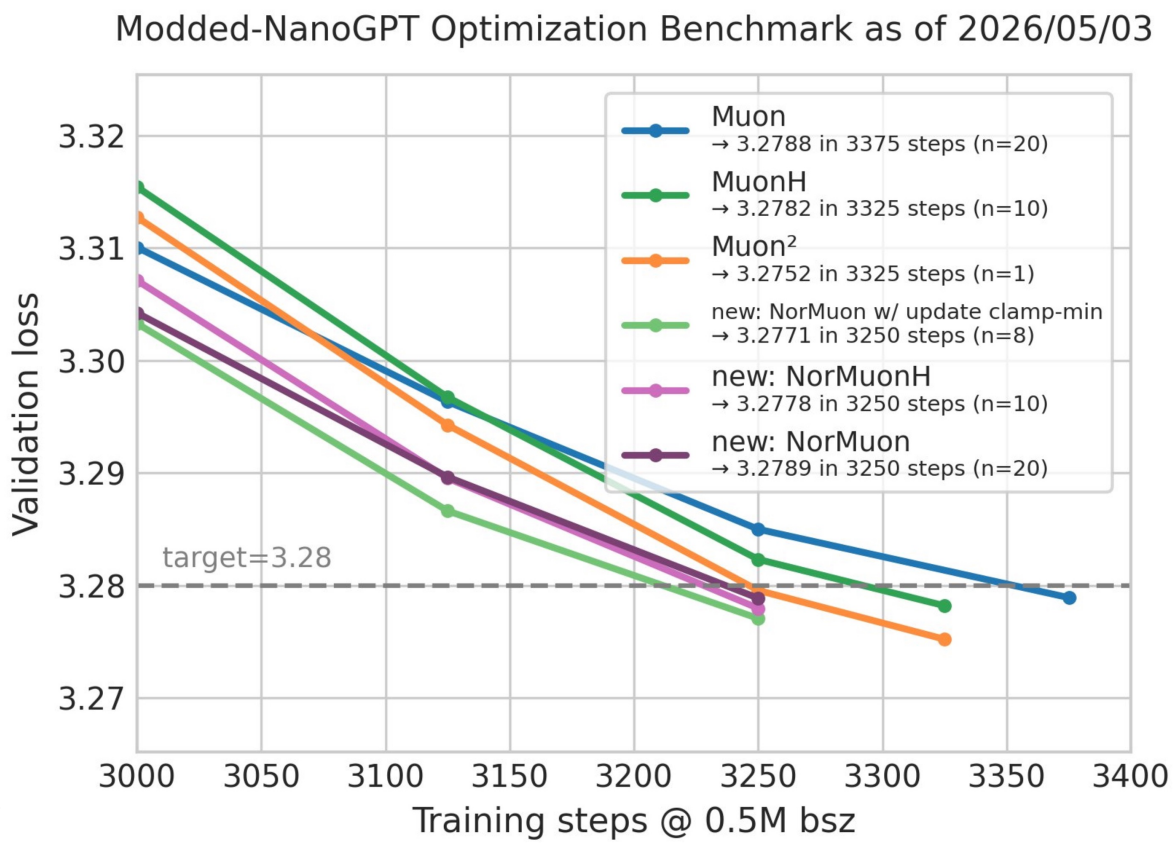
显示翻译

Three new modded-NanoGPT optimization benchmark results, all of them using NorMuon, have near-concurrently improved the benchmark record from 3325 to 3250 steps.

1) Kumar Krishna Agrawal (gh:kumarkrishna) used NorMuon with an update-clamping strategy (no weight decay, like hyperball) to reach the target loss in 3250 steps.
2) @wen_kaiyue used NorMuon with hyperball optimization.
3) Subsequently, Liming Liu (gh:lliu606) (one of the NorMuon authors!) used standard NorMuon with weight decay.

NorMuon is also used in the main NanoGPT speedrun track. Ali Naeimi did the earliest NorMuon runs on the optimization benchmark, albeit below stat sig.

NorMuon authors: @li_zichong, Liming Liu, @chenliang1_, @WeizhuChen, and @tourzhao



Row-wise Normalized Methods

- Recently, a surge of new **row-wise normalized methods**: (RMNP, Mano, Aurora, Nora…)
- The design matches the block structure of Hessian
- Some are motivated by Adam-mini,

  some are independently designed

- **Experiments**: these methods can even match Muon.
- Curious how they perform in larger scale

Concluding Remarks

Summary

Part 1: Global Landscape

--It was believed that wide enough neural nets are easy to train to global minima, due to evidence in vision (e.g. ImageNet classification).

Assume n data points, last hidden layer (or one layer) has m neurons

--Thm 1.1:$m \geq n$, bad local-min exists.

--Thm 1.2:$m \geq n$, no bad basin **(width ==> no basin)**

Thm 1.3:$m < n$, bad basin can exist (**narrow-net has basin**)

--Thm 1.4: Two-layer net with ReQU or ReLU neuron, m > $\Theta(n)$, with regularizers, bad local min disappears.

--Thm 1.5: stationary points become local-min-saddle plateau or all-saddle plateau; specific conditions are given.

Part 2: Local Structure and Algorithms

--Observation: Hessian of Neural nets exhibit nearly block diagonal structure (after early stage of training)

--**Thm 2.1 (nearly block diagonal)** For 2 layer nets, when output dim C is big enough, its off-block-diagonal parts of Hessian corresponding to neurons in the same layer is small.

--Thm 2.2 (convergence): Adam converges with large enough β_2.

--Explanation on benefit of Adam: Transformer has heterogeneous block Hessian, thus Adam works much better.

--Adam-mini: block-partition based on Hessian structure + block-mean of 2nd momentum.

--Nor-muon: apply neuron-wise mean to muon.

Open Question

Open Question 1 (landscape): For more-than-2-layer regularized neural nets, can we show no sub-optimal local-min?

Open Question 2 (convergence to global-min): mildly wide NN with >= 2 neurons, convergence of GD to global-min? [with mild conditions]

Open Question 3 (understanding):
Full characterization on when Adam and muon are faster than SGD.

Open Question 4 (design):
Better algorithms than Adam and muon variants?

Open Question 5 (design):
Algorithms that converge to better local minima in LLMs?

Other AI Related Optimization Questions

- **Continual Learning**

  Important problem now adays.

  Different formulation. Maybe related to landscape.

 (our related work: MoFO

Chen, et al. "Mofo: Momentum-filtered optimizer for mitigating forgetting in llm fine-tuning." *TMLR 2025; J2C track, displayed in ICML 2026)*

- **Self-improvement training**

  Important problem. Any optimization explanation?

 (our related work SePT, with Xiao Li:
Li, Mengqi, Lei Zhao, Anthony So, Ruoyu Sun, and Xiao Li. "A Model Can Help Itself: Reward-Free Self-Training for LLM Reasoning." *arXiv,* 2025).)

Thanks to My Collaborators

Yushun Zhang

Tian Ding

Shiyu Liang

Congliang Chen

Dawei Li

Naichen Shi

R. Srikant

Zhi-Quan Luo

References on landscape

Book in preparation:

R. Sun, “Lectures on deep learning optimization”, in preparation. To be published by **Springer Nature**.

Surveys:

- [S1] **R. Sun**, D. Li, S. Liang, T. Ding, R. Srikant, “The Global landscape of neural networks: An overview.” **IEEE Signal Processing Magazine**, 2020
- **[S2] R. Sun**, “Optimization for deep learning: an overview.” (survey paper). Journal of Operations Research Society of China, Special Issue on “Continuous Optimization: Past, Present and Future” (invited). 2020.

Major results on landscape are from the following papers:

- [R1] T. Ding, D. Li, **R. Sun**, “Spurious local minima exist for almost all over-parameterized neural networks”, accepted to **Mathematics of Operations Research, 2021**.
- [R2] D. Li, T. Ding, **R. Sun,** “On the Benefit of Width for Neural Networks: Disappearance of Basins”, accepted to **SIAM Journal on Optimization, 2022**.
- [R3] Shiyu Liang, **Ruoyu Sun** and R. Srikant. Revisiting Landscape Analysis in Deep Neural Networks: Eliminating Decreasing Paths to Infinity. **SIAM Journal on Optimization**, 2022.
- [R4] S. Liang, **R. Sun**, R Srikant. Achieving Small Test Error in Mildly Overparameterized Neural Networks. arxiv.
- [R5] **R. Sun**, T. Fang, A. Schwing, “Towards a better global loss landscape of GANs.” **NeurIPS** 2020 (**Oral**, 1.1% of 9500+ submissions)
- [R6] A Geometric Characterization of the Stationary Plateau for Two-Layer Neural Networks. Tian Ding, Dawei Li, Ruoyu Sun. Arxiv. https://arxiv.org/abs/2606.04327

Work on matrix completion:

R. Sun, Z.-Q. Luo, "Guaranteed Matrix Completion via Nonconvex Factorization”, **FOCS** 2015; IEEE **TIT** (IEEE Transaction on Information Theory), 2016.

References on Local Structure and Algorithms

Work on Hessian structure:

Dong*, Zhang*, Yao, **Sun**; Towards Quantifying the Hessian Structure of Neural Networks, arxiv, 2025.

Works on Adam:

N. Shi, D. Li, M. Hong, **R. Sun**, "RMSprop converges with proper hyper-parameter." **ICLR** 2021 (**Spotlight**, 3.3% of 3000+ submissions)

X. Chen, S. Liu, **R. Sun**, M. Hong. On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization*, Proc.* **ICLR** 2019.

Yushun Zhang, Congliang Chen, Naichen Shi, **Ruoyu Sun***, Zhi-Quan Luo, Adam Can Converge Without Any Modification on Update Rules, NeurIPS 2022 (spotlight, 5.5%).

New version with updated proofs:: https://arxiv.org/pdf/2603.02092

Zhang, Chen, Ding, Li, **Sun***, & Luo, Why Transformers Need Adam: A Hessian Perspective, NeurIPS 24

Zhang*, Chen*, Li, Ding, Chen, Kingma, Ye, Luo & **Sun***, Adam-mini: Use Fewer Learning Rate To Gain More, ICLR 2025

Other works on optimization of neural nets:

- S. Liang, R. **Sun**, Y. Li, R. Srikant, "Understanding the loss surface of neural networks for binary classification". **ICML** 2018.
- S. Liang, R. **Sun**, J.D. Lee, R. Srikant, "Adding one neuron can eliminate all bad local minima". ***NIPS*** 2018.
- Lin, R. **Sun** and Z. Zhang, Faster Directional Convergence of Linear Neural Networks under Spherically Symmetric Data, **NeurIPS** 2021

Thank You for Listening!

Email 邮箱：sunruoyu@cuhk.edu.cn
Website 个人网站：ruoyus.github.io