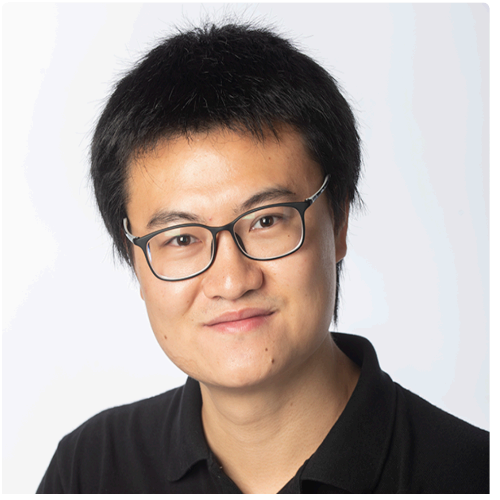
Scaling Scientific Machine Learning at both Training and Inference

Yiping Lu

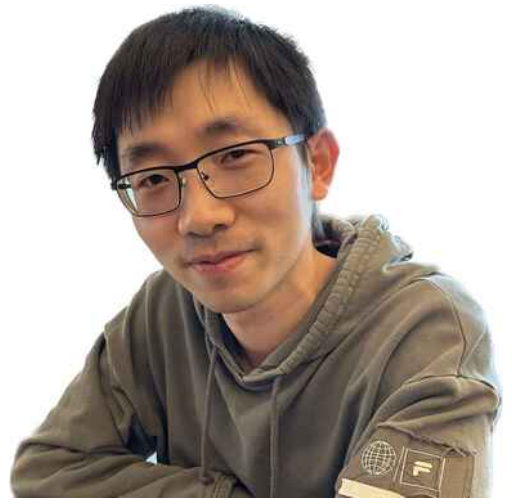
Northwestern | McCORMICK SCHOOL OF ENGINEERING

Lexing Ying (Stanford)

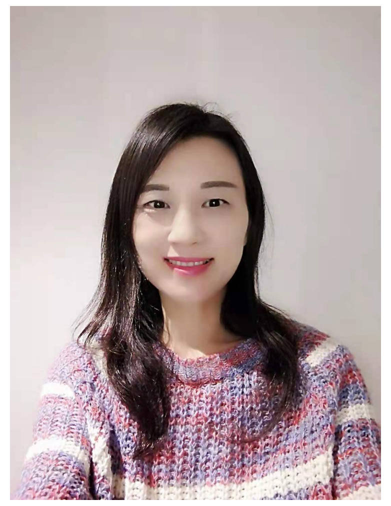
Jose Blanchet (Stanford)



Shihao Yang (Gatech)



Sifan Wang (Yale)



Chunmei Wang (UF)

Jiajin Li (UBC)

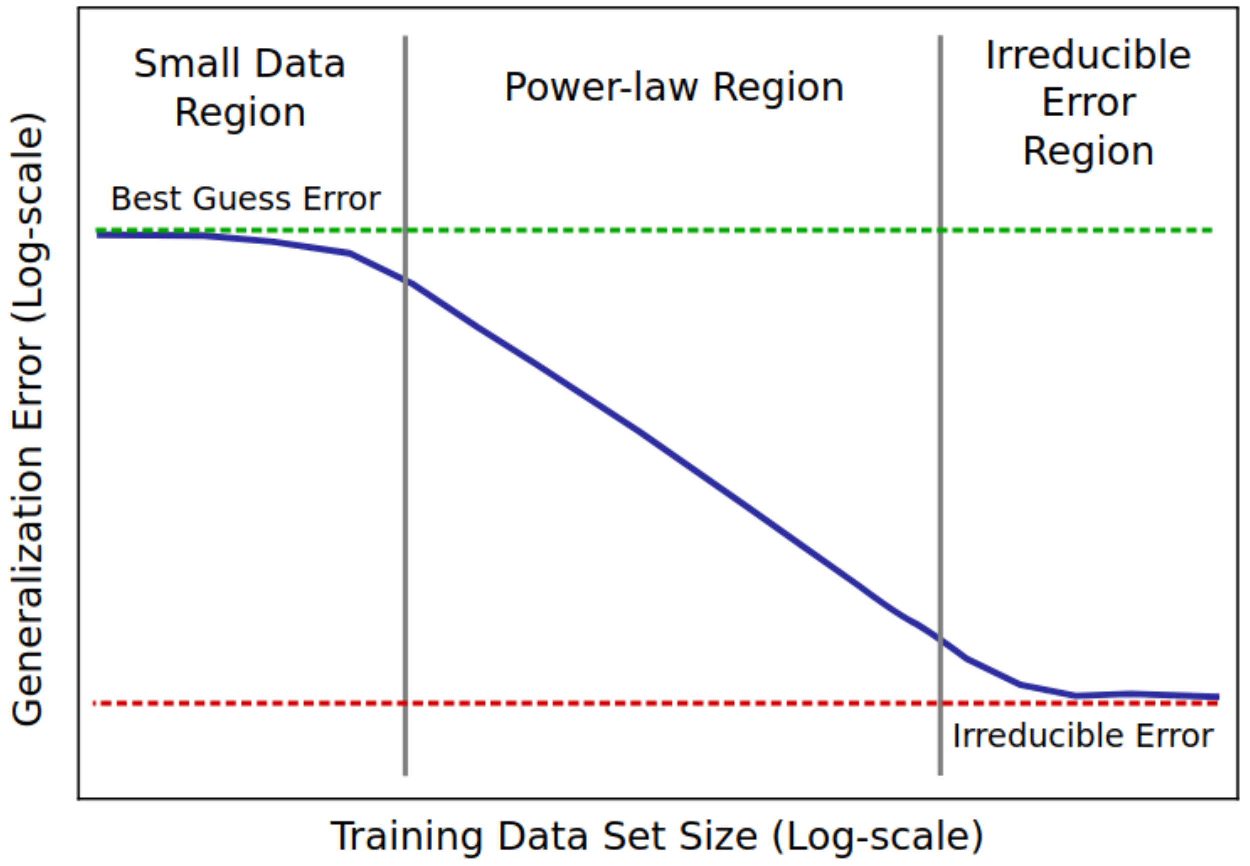
Students: Haoxuan Chen, Yinuo Ren(Stanford), Youheng Zhu, Kailai Chen (Northwestern), Jasen Lai (UF), Zhaoyan Chen, Weizhong Wang (FDU), Kaizhao Liu (PKU->MIT), Zexi Fan (PKU), Ruihan Xu (Uchicago)

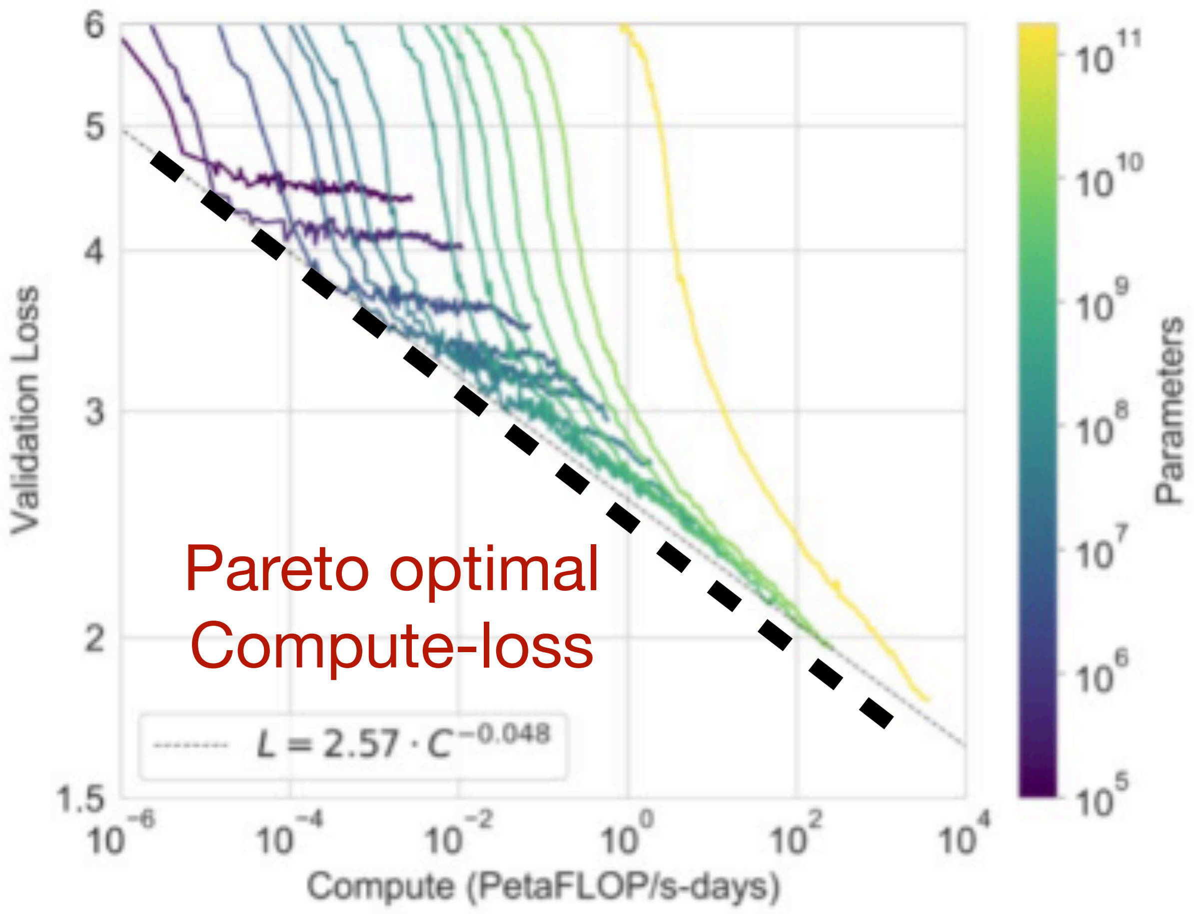
…

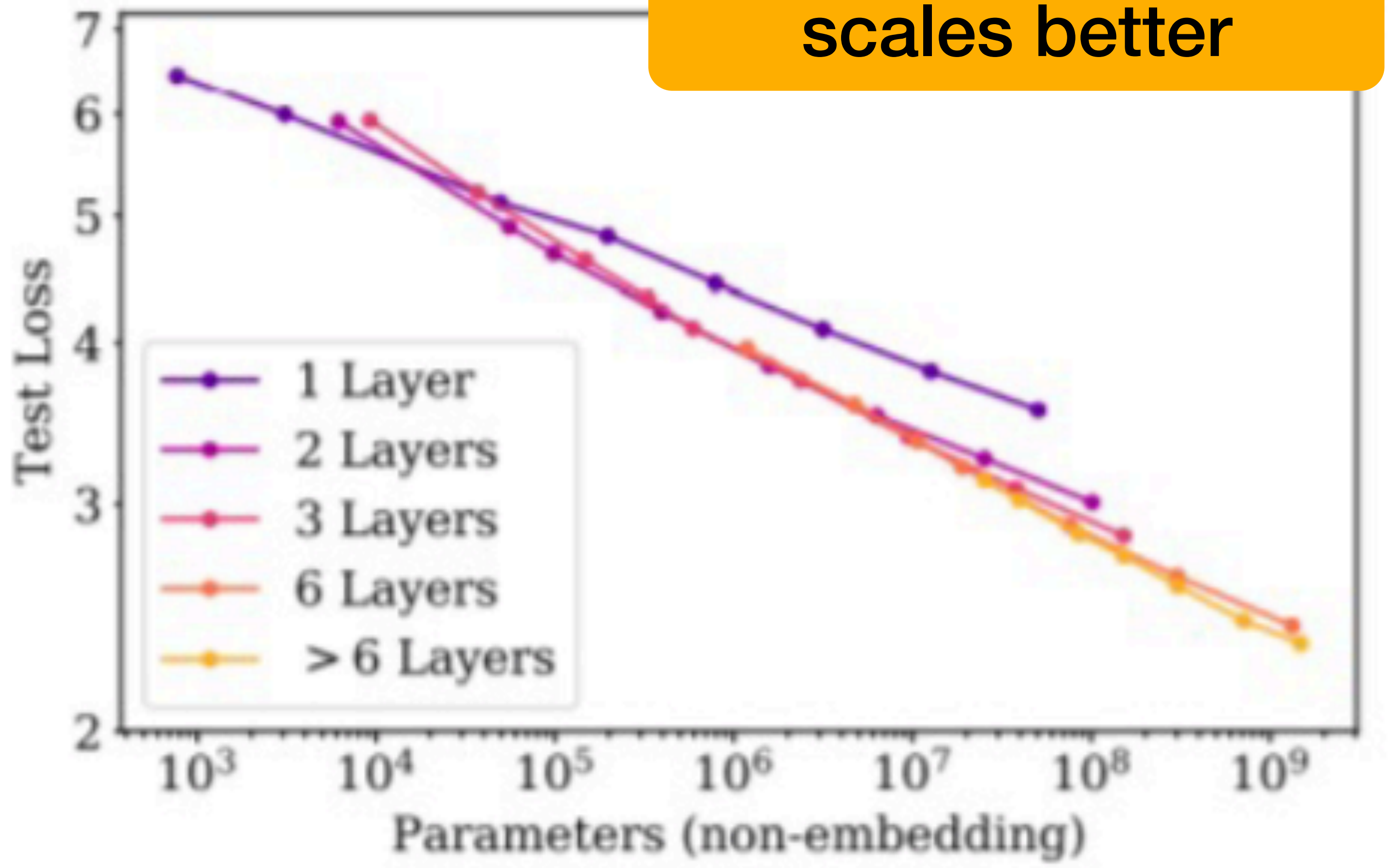
Is Scaling All We Need?

Why is depth all we need?

Because deeper scales better



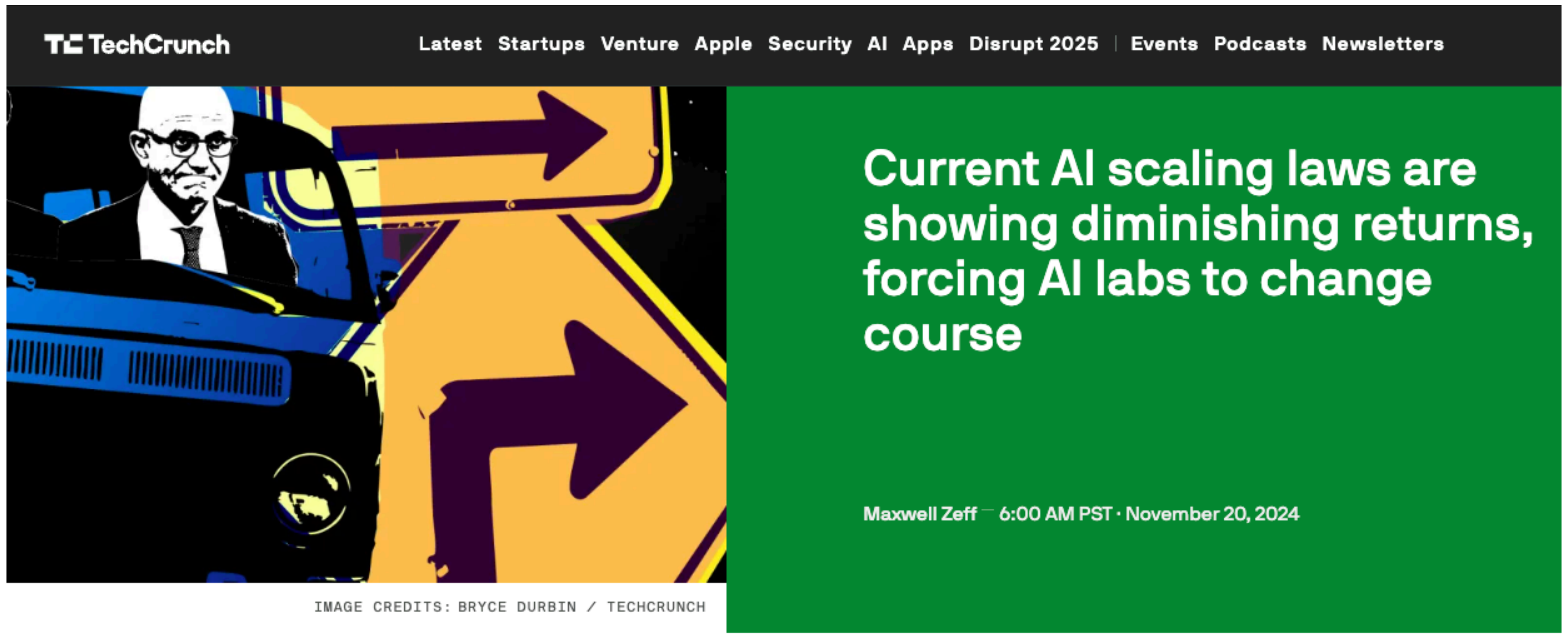




Why is attention all we need?

Because transformer scales?

What does scale means mathmetically?



What is Scaling Law?

Chinchilla scaling law: Training compute-optimal large language models. Neurips, 2022.

$$\hat{L}(N, D) := E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

Irreducible error

N: Number of parameters, D: number of data

This is what we do in the past!

Neurips 1993

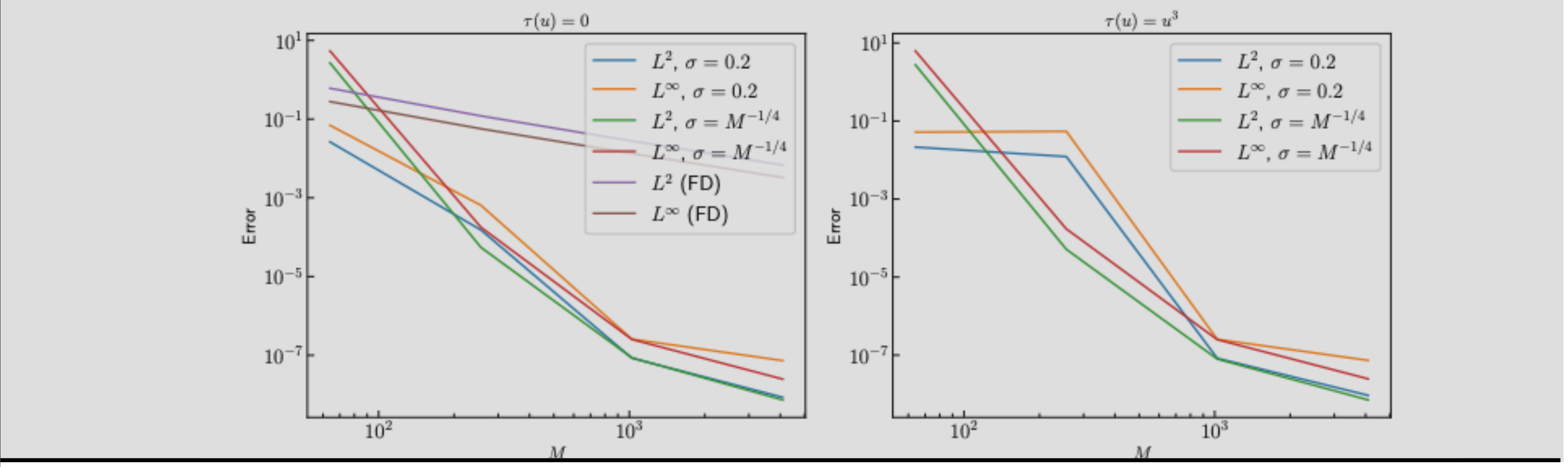
Learning Curves: Asymptotic Values and Rate of Convergence

Corinna Cortes, L. D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker
AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

Training classifiers on large databases is computationally demanding. It is desirable to develop efficient procedures for a reliable prediction of a classifier's suitability for implementing a given task, so that resources can be assigned to the most promising candidates or freed for exploring new classifier candidates. We propose such a practical and principled predictive method. Practical because it avoids the costly procedure of training poor classifiers on the whole training set, and principled because of its theoretical foundation. The effectiveness of the proposed procedure is demonstrated for both single- and multi-layer networks.

GP



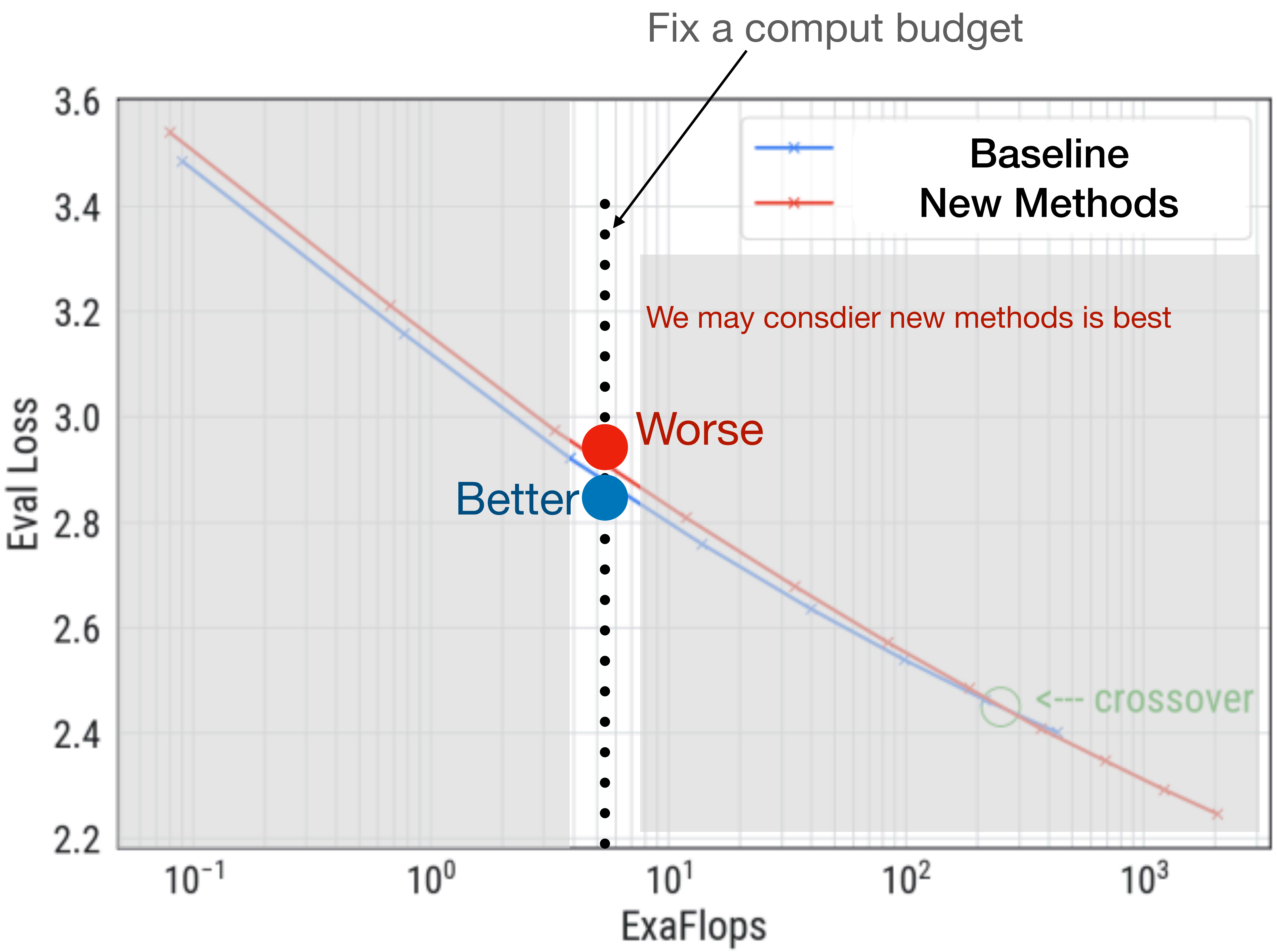
FEM

Table 9.3

Numerical error and convergence order for exact solution $u = x^2(1-x)^2y^2(1-y)^2$ on triangular partitions.

h	$\|u_b - Q_b u\|_\infty$	Order	$\|\mathbf{u}_g - Q_b(\nabla u)\|_\infty$	Order
1	0.41494		8.6485e−018	
5.0000e−01	0.08806	2.24	0.00942	
2.5000e−01	0.037013	1.25	0.00491	0.94
1.2500e−01	0.01069	1.79	0.00354	0.47
6.2500e−02	0.00293	1.87	0.00222	0.67
3.1250e−02	7.935e−004	1.88	0.00102	1.12
1.5625e−02	2.096e−004	1.92	3.577e−004	1.51
7.8125e−03	5.401e−05	1.96	1.053e−04	1.76

How does academia consider an algorithm to be good?

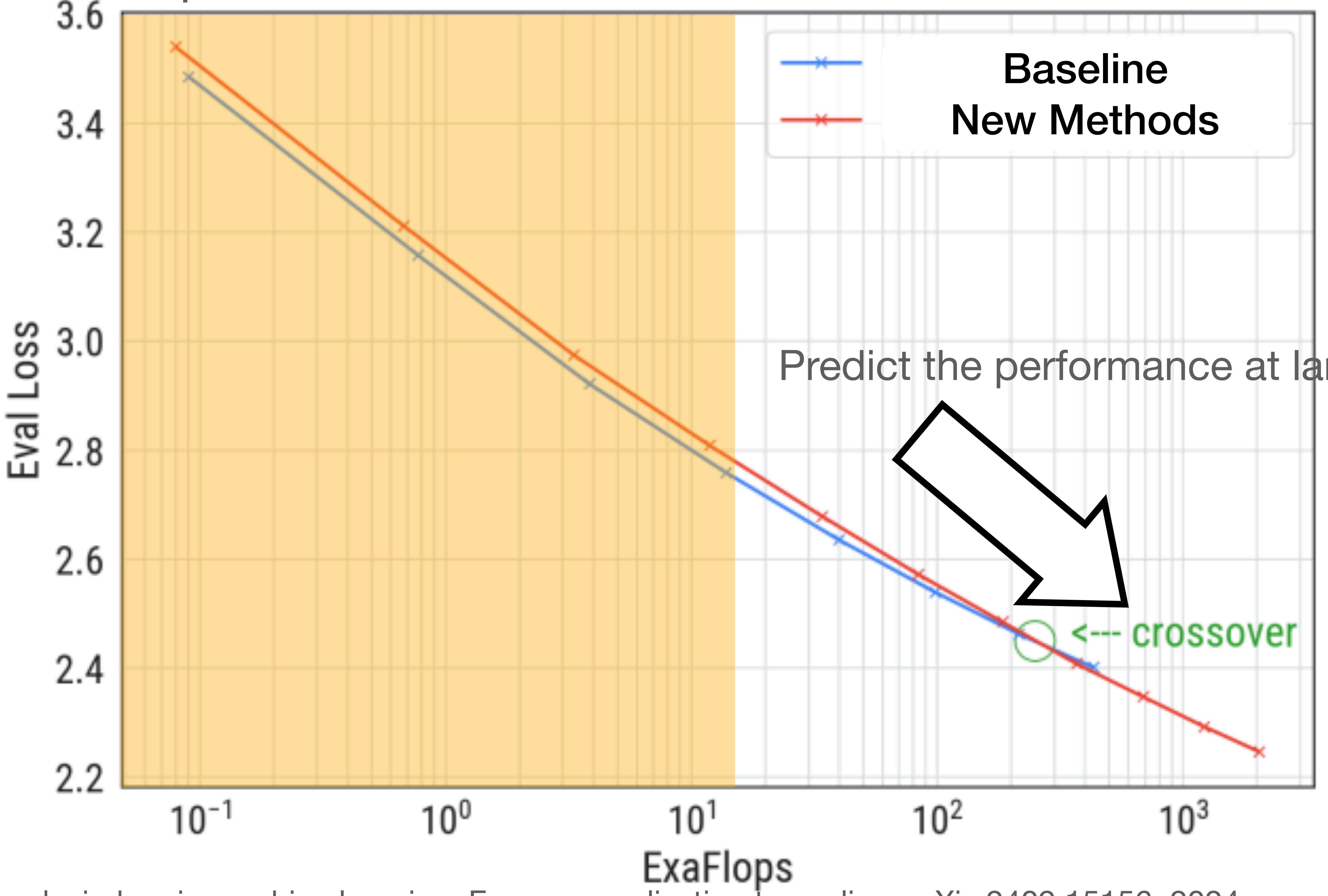


Xiao L. Rethinking conventional wisdom in machine learning: From generalization to scaling. arXiv:2409.15156, 2024.

How does industry consider an algorithm to be good?

Chinchilla Scaling Law

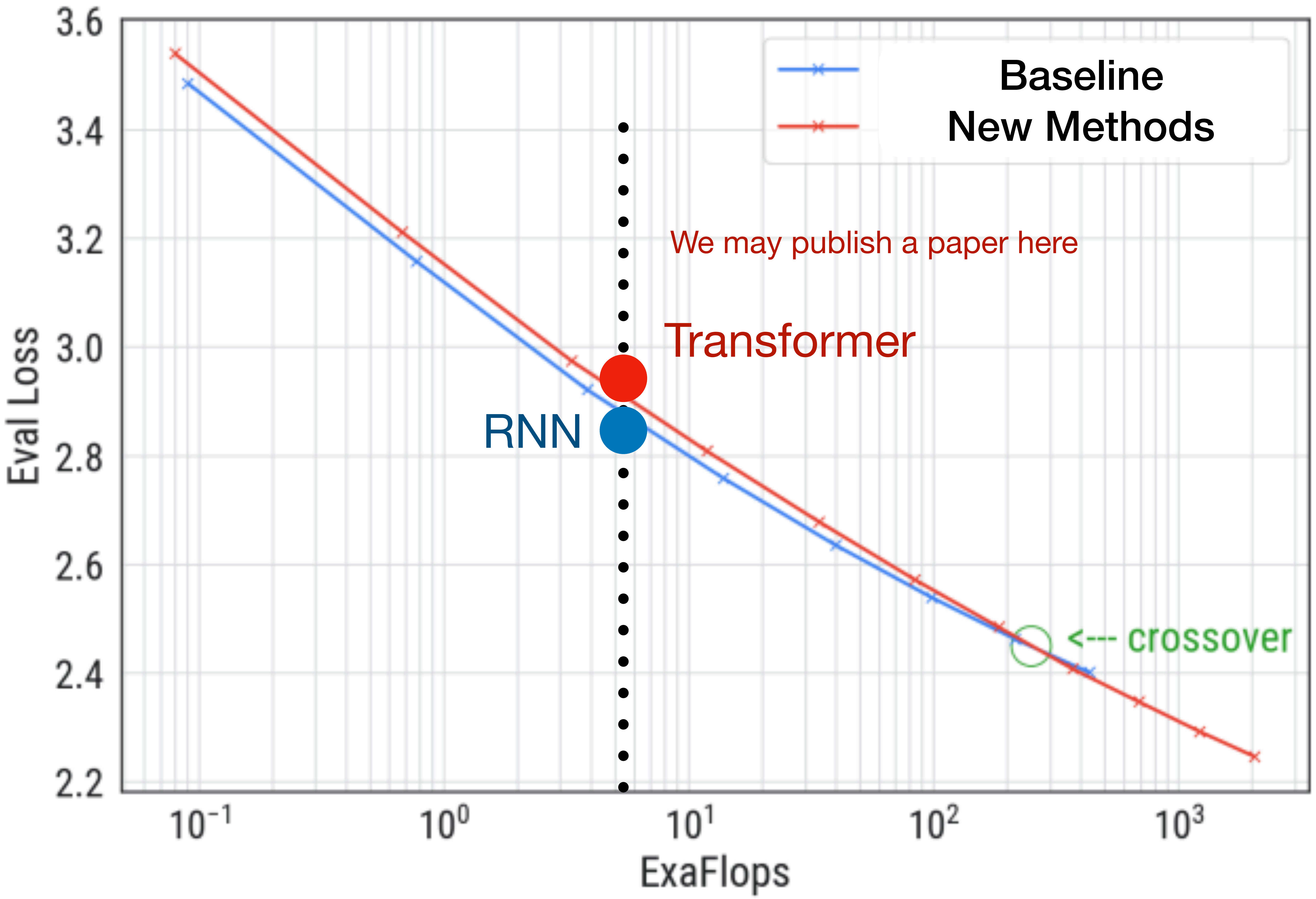
Test performance at small scale



Predict the performance at larger scale

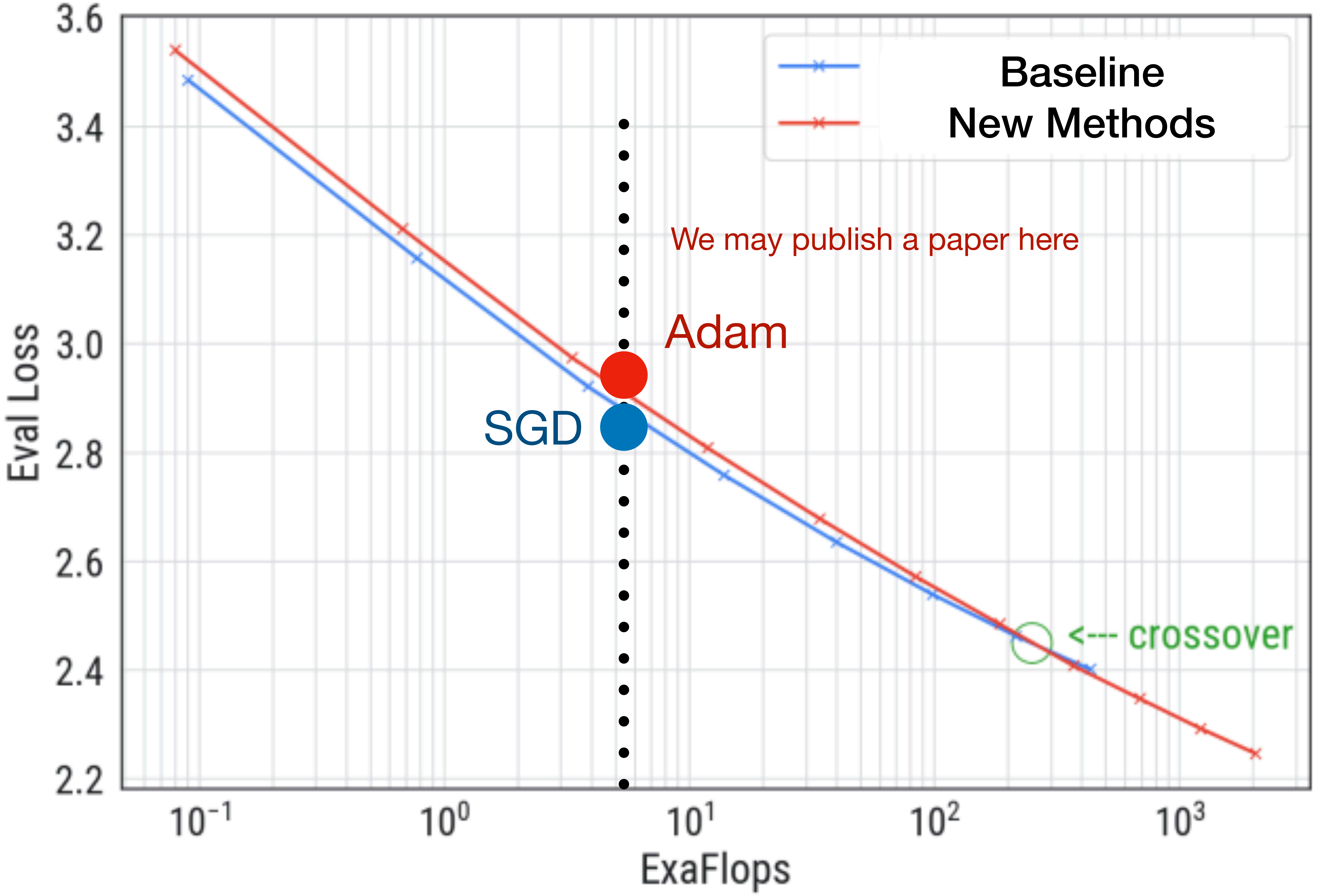
Xiao L. Rethinking conventional wisdom in machine learning: From generalization to scaling. arXiv:2409.15156, 2024.

Imagine what happens at ∞ Compute?



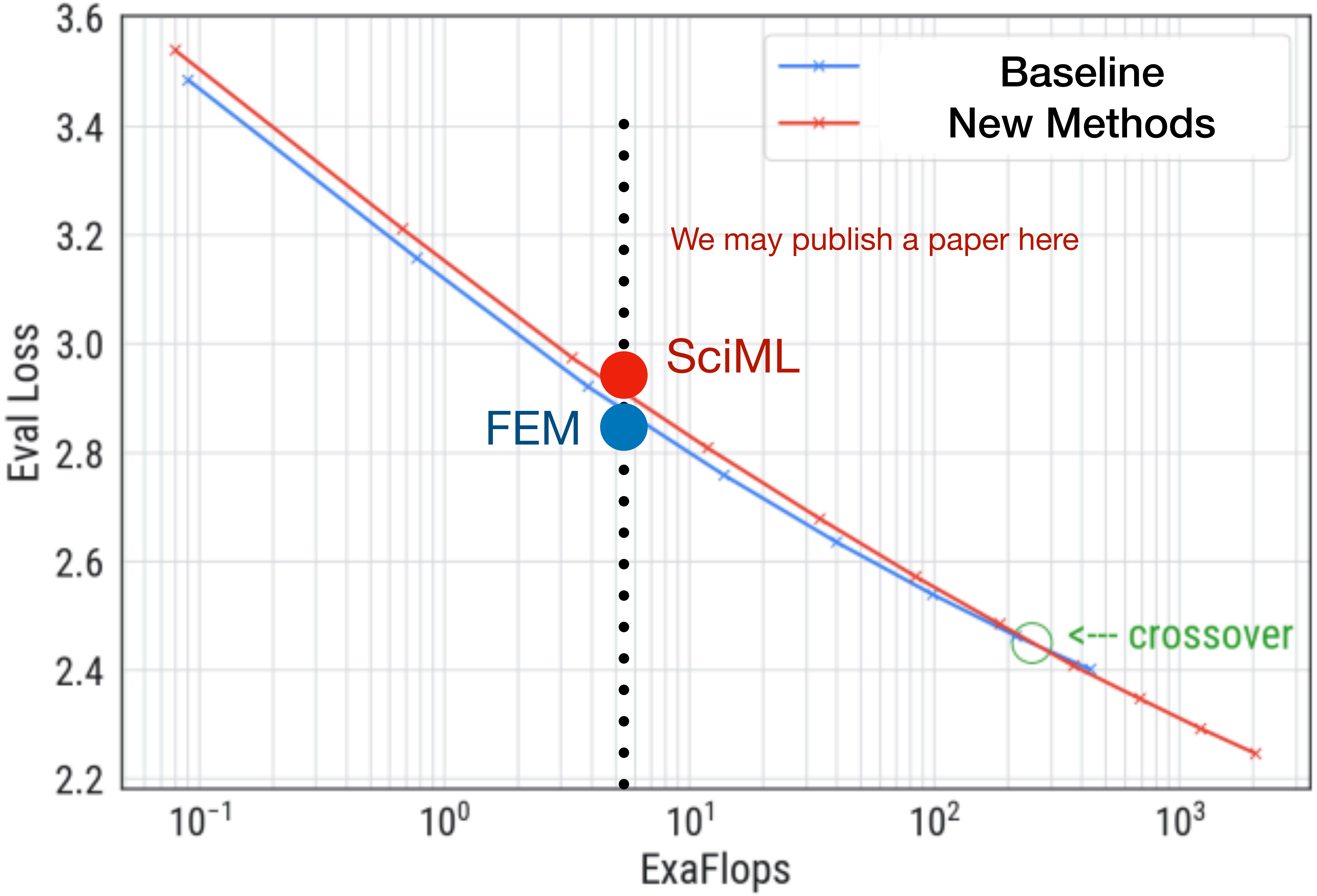
Xiao L. Rethinking conventional wisdom in machine learning: From generalization to scaling. arXiv:2409.15156, 2024.

Imagine what happens at ∞ Compute?



Xiao L. Rethinking conventional wisdom in machine learning: From generalization to scaling. arXiv:2409.15156, 2024.

Imagine what happens at ∞ Compute?



Xiao L. Rethinking conventional wisdom in machine learning: From generalization to scaling. arXiv:2409.15156, 2024.

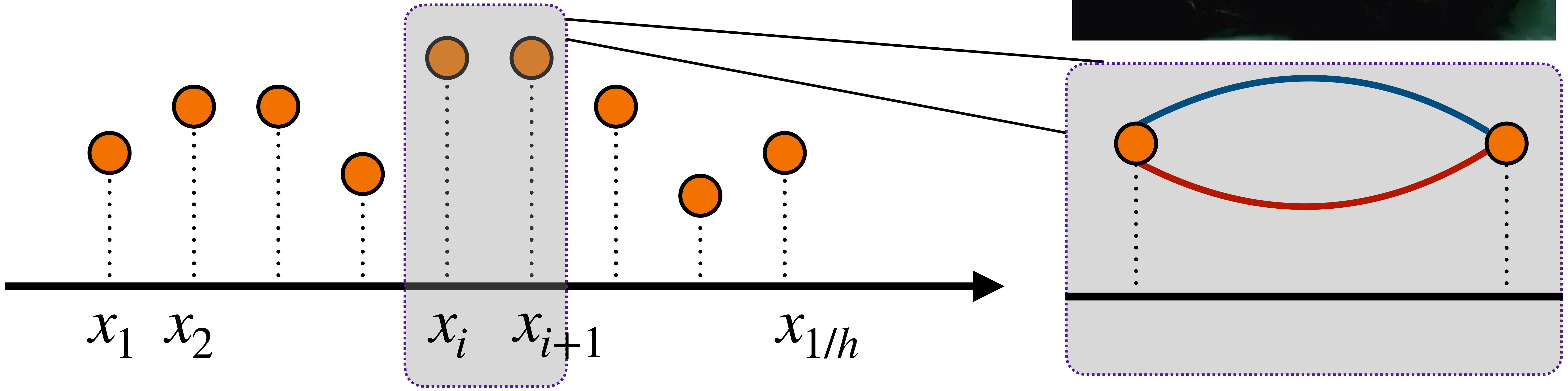
Scaling at Training Time

Is there an optimal scaling law?

Limit 1: Informational limit

Toy Example: Let's assume we work with a function f,
We can evaluate the function at a grid point $f(x_1), f(x_2), \cdots, f(x_{1/h})$

What is the error of best possible guess of f ?



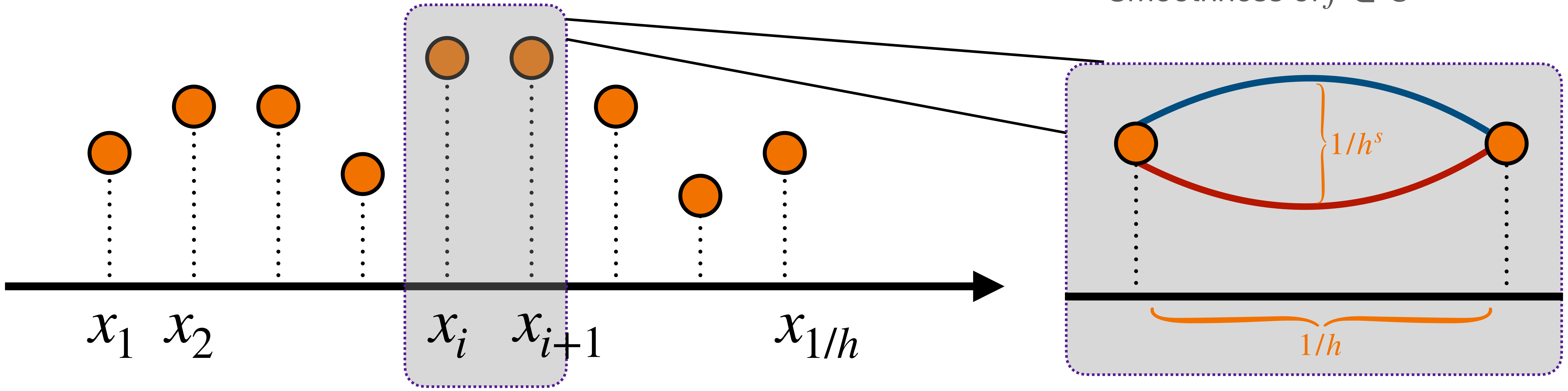
Is there an optimal scaling law?

Limit 1: Informational limit

Toy Example: Let's assume we work with a function f,
We can evaluate the function at a grid point $f(x_1), f(x_2), \cdots, f(x_{1/h})$

What is the error of best possible guess of f ?

- Dimension of collocation points
- Smoothness of $f \in C^s$



Is there an optimal scaling law?

Limit 1: Informational limit

Toy Example: Let's assume we work with a function f,
We can evaluate the function at a grid point $f(x_1), f(x_2), \cdots, f(x_{1/h})$

What is the error of best possible guess of f ?

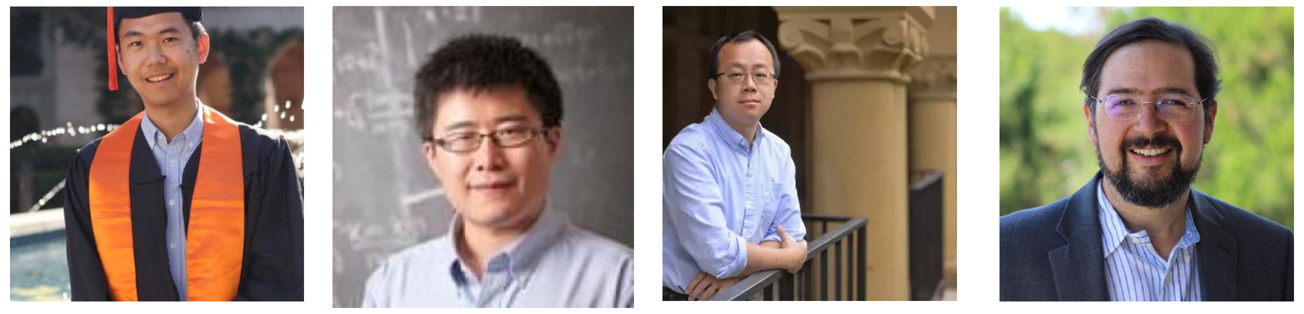
With n observations $(x_i, y_i = f(x_i) + \text{noise})_{i=1}^{n}$
No algorithm can better than $O\left(n^{-\frac{2(s-t_1)}{d+2s-t_2}}\right)$

- we want to evalue $u \in W^s$ in W^{t_1}
- It's a t_2-order PDE (much simplified)

Extend to PDE problems?

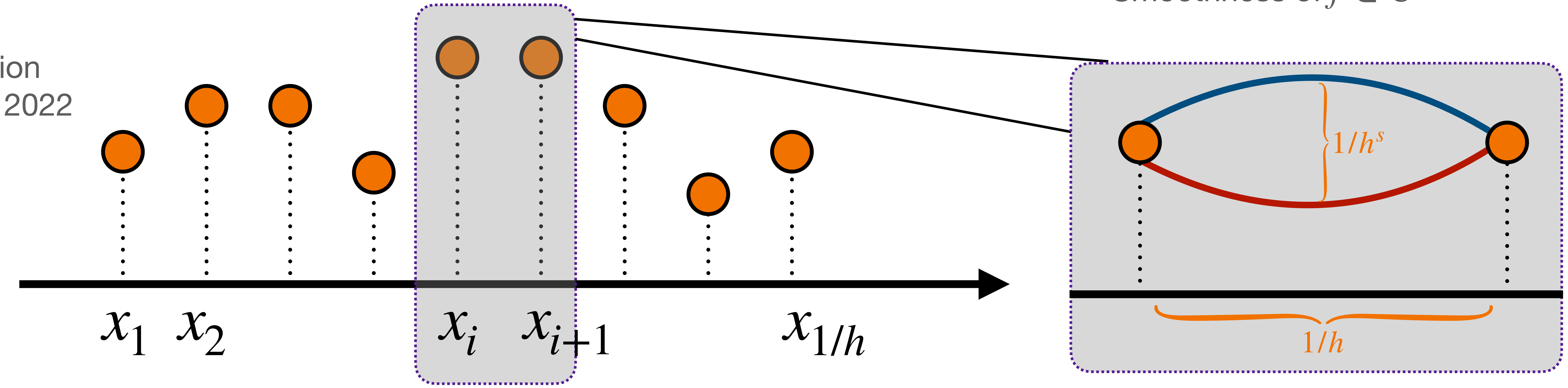
PDE Problem: $\Delta u = f$, with collocation points $f(x_1), \cdots, f(x_{1/h})$
Information theoretically best f leads to the best u

Machine learning for elliptic PDEs: Fast rate generalization bound, neural scaling law and minimax optimality ICLR 2022



Haoxuan Chen, Jianfeng Lu, Lexing Ying, Jose Blanchet

- Dimension of collocation points
- Smoothness of $f \in C^s$

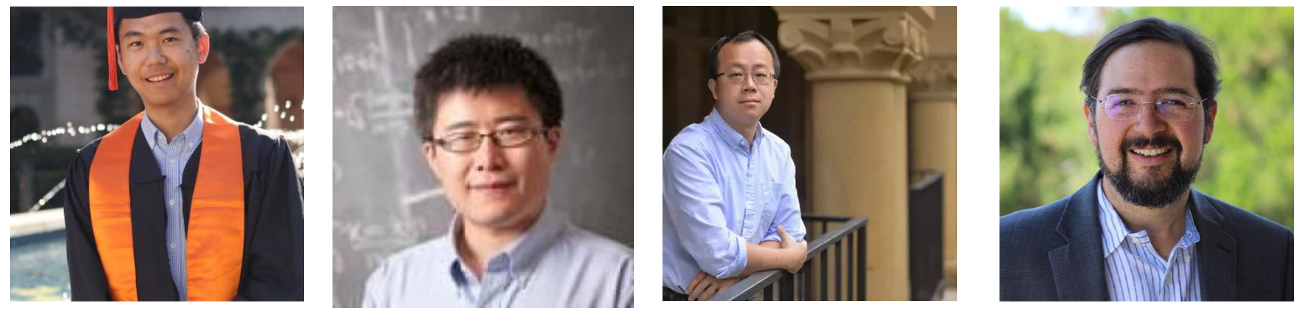


Information Limit for Scientific Computing

PDE Problem: $\Delta u = f$, with *random* collocation points $f(x_1), \cdots, f(x_n)$
Information theoretically best f leads to the best u

Algorithm insight: Integral by parts leads to suboptimal variance

Machine learning for elliptic PDEs: Fast rate generalization bound, neural scaling law and minimax optimality ICLR 2022



Haoxuan Chen, Jianfeng Lu, Lexing Ying, Jose Blanchet

Eigenvalue Problem: $\frac{1}{p}\nabla \cdot (p^2 \nabla u) = \lambda u,$

with collocation points $x_1, \cdots, x_n$ sample from $p \in C^m$
Information theoretically best p leads to the best u?

Algorithm insight: New Kernel Selection for Graph Laplacian $\int K(u)u^s ds = 0$

Optimal Spectral Convergence of High-Order Graph Laplacians under Smooth Densities (arXiv soon)

Weizhong Wang, Ruiyi Yang

Quadrature Rule: $\int_{[0,1]^d} f(u)du, f \in C^m$, with collocation points $f(x_1), \cdots, f(x_{1/h})$

Later today

Algorithm insight: Quadrature rule+MC is better than Quadrature rule/MC

When can a regression-adjusted control variate help? Rare events, Sobolev embedding, and minimax optimality Neurips 2023

Haoxuan Chen, Lexing Ying, Jose Blanchet

Information Limit for Scientific Computing

Linear Operator Learning: recover operator $\mathscr{A}$ using $(f_1, \mathscr{A}f_1), \cdots (f_n, \mathscr{A}f_n)$

Minimax optimal kernel operator learning via multilevel training
ICLR 2023 Spotlight

Jikai Jin, Jose Blanchet, Lexing Ying

Algorithm insight: learning an Infinite-dimensional operator is different from learning finite finite-dimensional matrix. It naturally need multiscale regularization on different spectral.

Similar as MLMC

Solve PDE at a single point : $\Delta u = f$, with *designed* collocation points $f(x_1), \cdots, f(x_n)$

Aim: recover $u(x)$

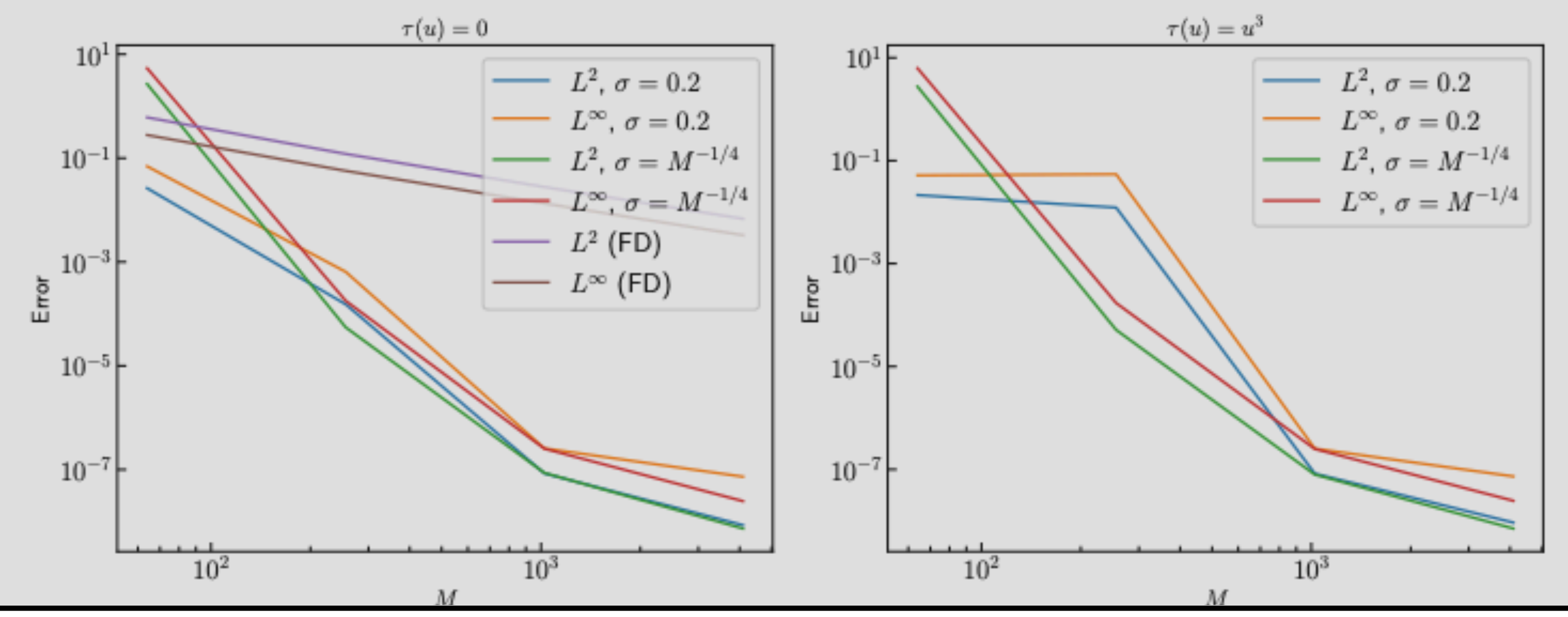
Algorithm insight: solving a PDE at a single point converges faster than approximating the PDE solution over the entire domain

Later today

Is there an optimal scaling law?

Limit 1: Computational (Optimization) limit

GP



FEM

Table 9.3
Numerical error and convergence order for exact solution $u = x^2(1-x)^2y^2(1-y)^2$ on triangular partitions.

h	$\|u_b - Q_b u\|_\infty$	Order	$\|\mathbf{u}_g - Q_b(\nabla u)\|_\infty$	Order
1	0.41494		8.6485e−018	
5.0000e−01	0.08806	2.24	0.00942	
2.5000e−01	0.037013	1.25	0.00491	0.94
1.2500e−01	0.01069	1.79	0.00354	0.47
6.2500e−02	0.00293	1.87	0.00222	0.67
3.1250e−02	7.935e−004	1.88	0.00102	1.12
1.5625e−02	2.096e−004	1.92	3.577e−004	1.51
7.8125e−03	5.401e−05	1.96	1.053e−04	1.76

PINN

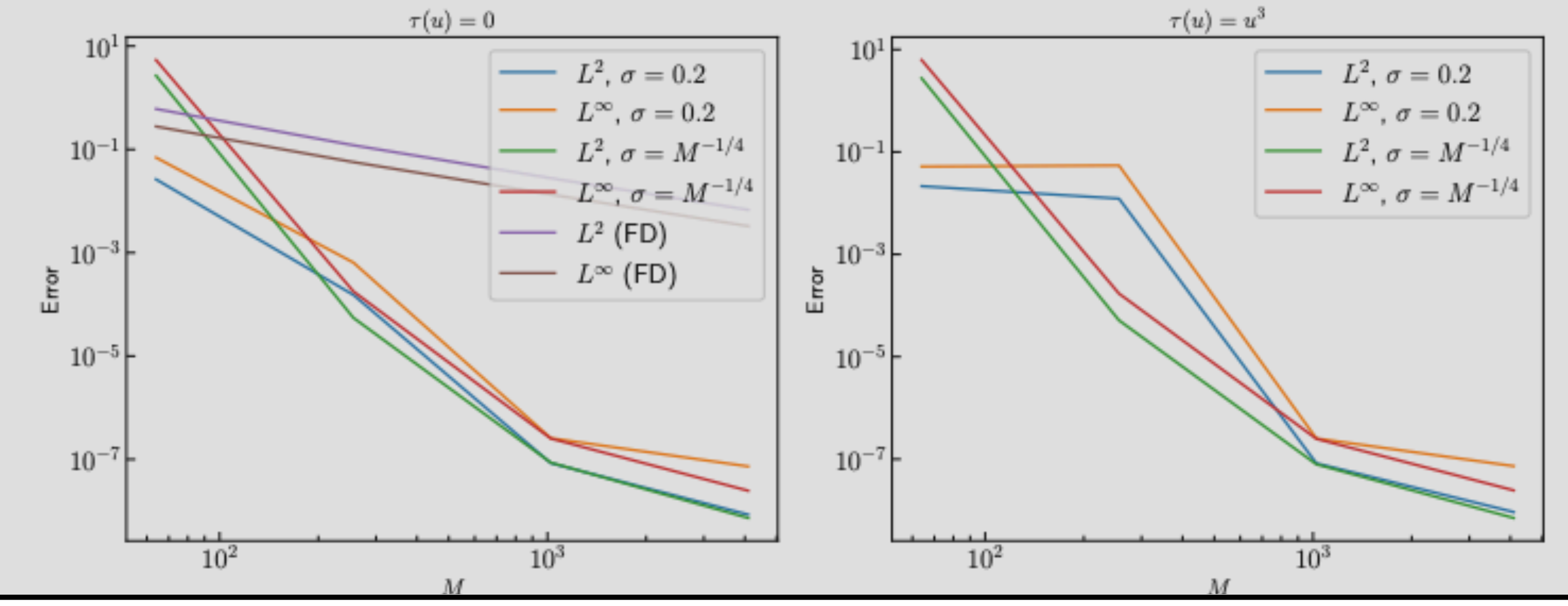
A. There is not a scaling law for NN that can't be optimized to high precision

B. They don't have enough GPUs

Is there an optimal scaling law?

Limit 1: Computational (Optimization) limit

GP



FEM

Table 9.3
Numerical error and convergence order for exact solution $u = x^2(1-x)^2y^2(1-y)^2$ on triangular partitions.

h	$\|u_b - Q_b u\|_\infty$	Order	$\|\mathbf{u}_g - Q_b(\nabla u)\|_\infty$	Order
1	0.41494		8.6485e−018	
5.0000e−01	0.08806	2.24	0.00942	
2.5000e−01	0.037013	1.25	0.00491	0.94
1.2500e−01	0.01069	1.79	0.00354	0.47
6.2500e−02	0.00293	1.87	0.00222	0.67
3.1250e−02	7.935e−004	1.88	0.00102	1.12
1.5625e−02	2.096e−004	1.92	3.577e−004	1.51
7.8125e−03	5.401e−05	1.96	1.053e−04	1.76

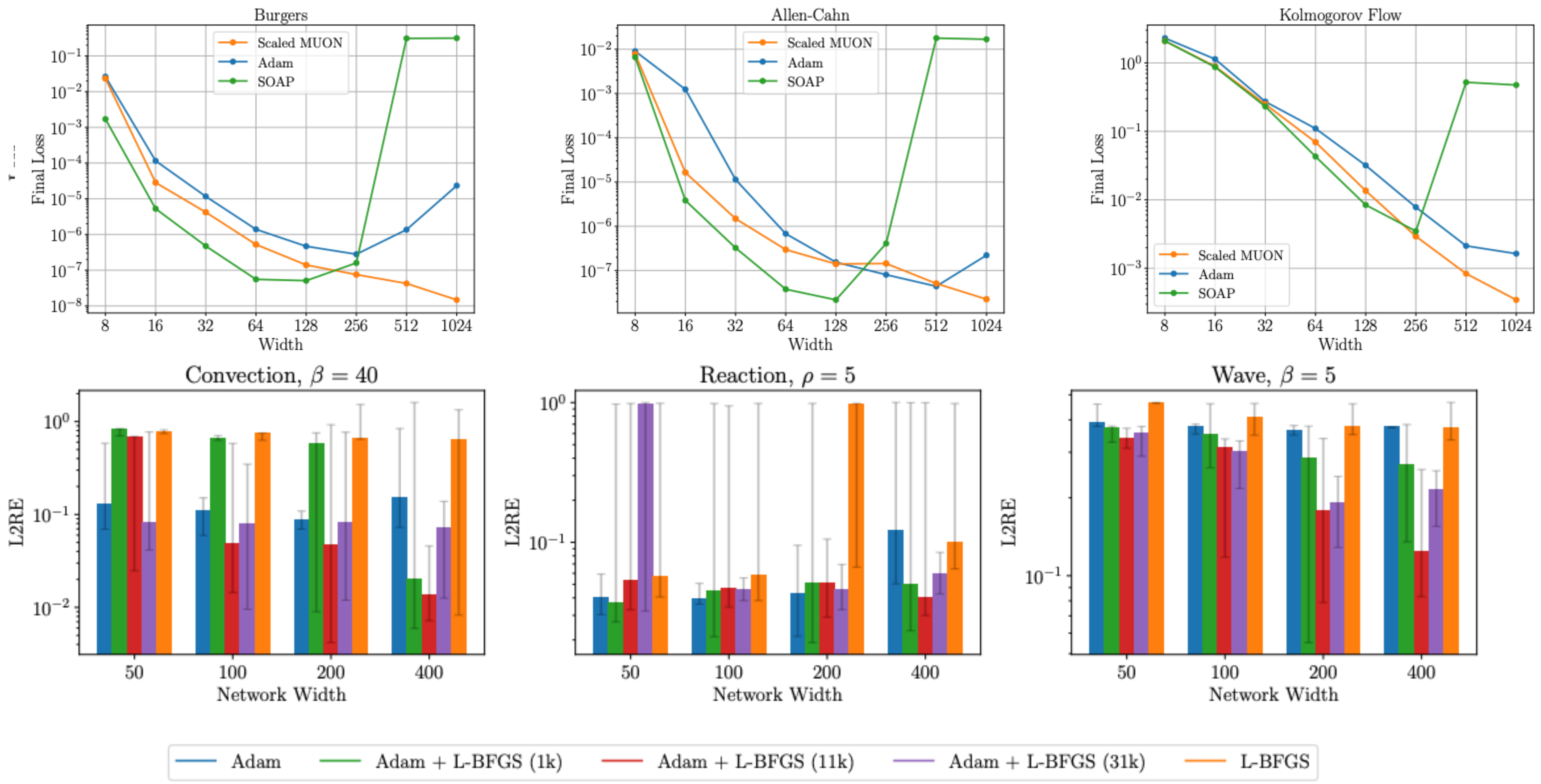
PINN

Loss = Approximation Error + Generalization/MC Error + Optimization Error

Provable optimal with global optimization

Larger networks are harder to optimize !

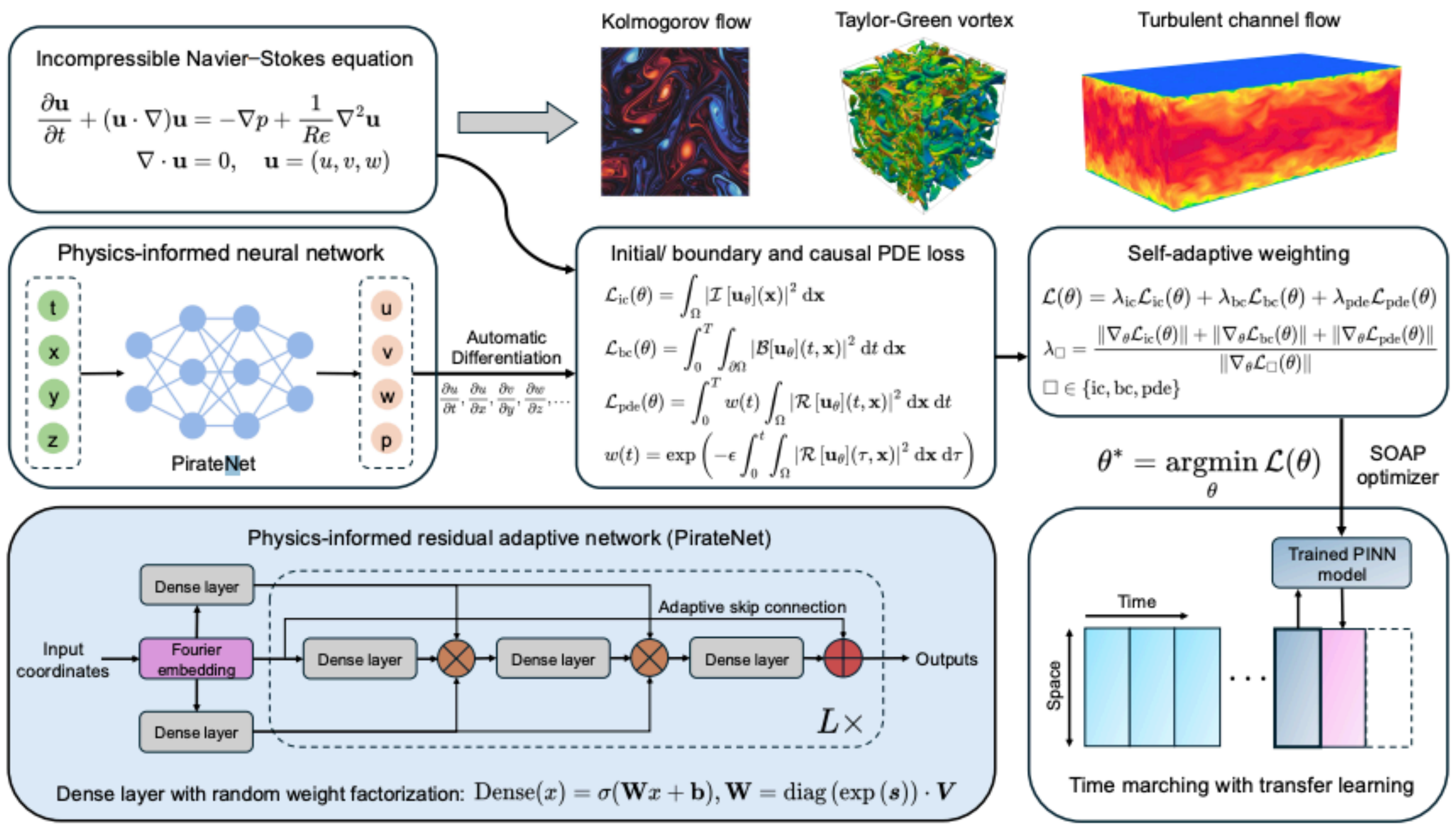
Only thing can't scale!



Rathore P, Lei W, Frangella Z, et al. Challenges in training pinns: A loss landscape pers

Power of Scaling PINN

comparable to 8-th order finite difference on 256x256x256 with $\Delta t = 10^{-3}$
7.45 hour on a single NVIDIA H200 GPU



Key Component: SOAP Optimizer

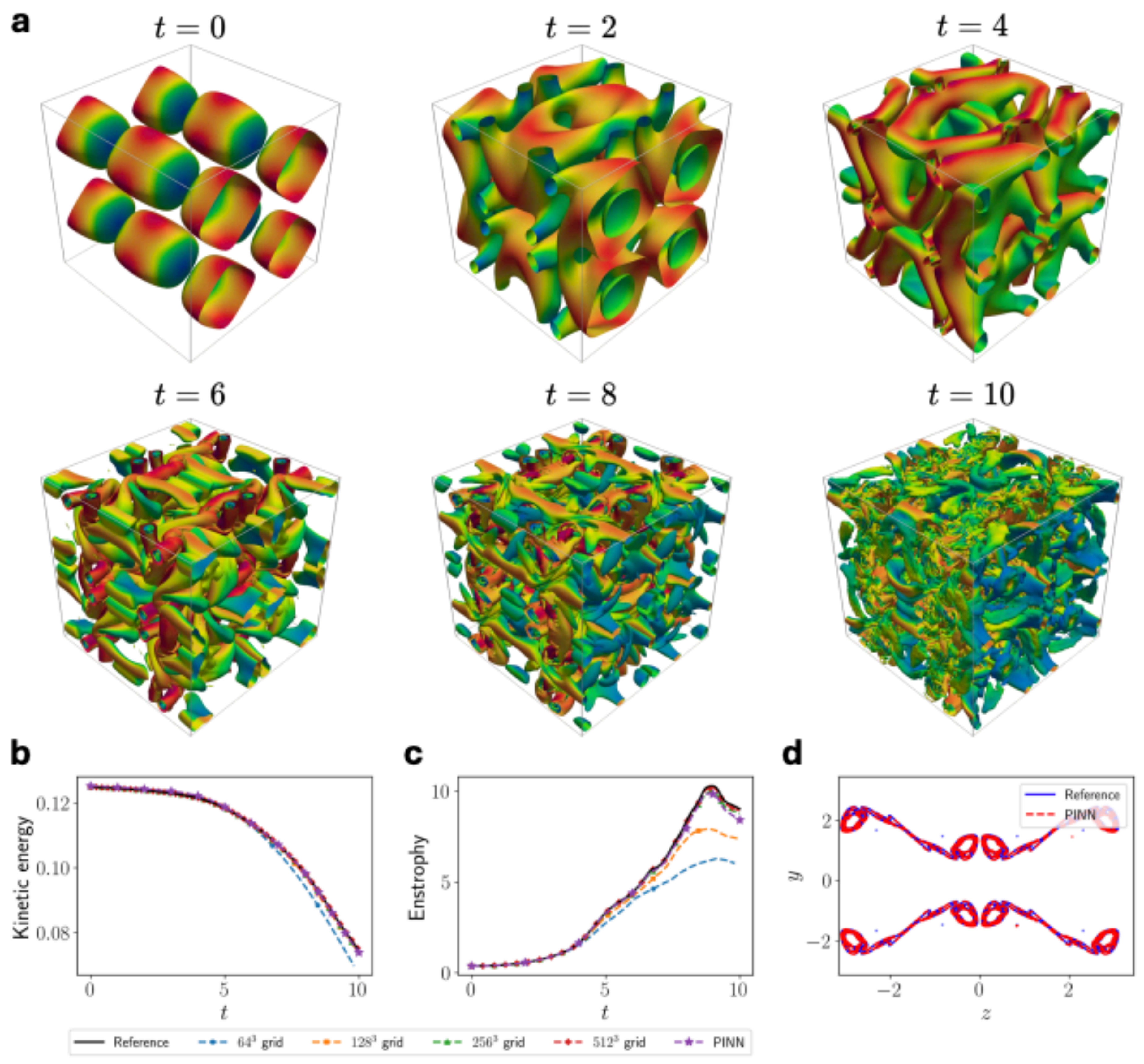


Figure 3. *Taylor-Green Vortex (Re=1600).* (a) Evolution of the iso-surfaces of the Q-criterion ($Q = 0.1$) at different time snapshots, predicted by PINNs and colored by the non-dimensional velocity magnitude. (b–c) Temporal evolution of spatially averaged kinetic energy and enstrophy, comparing PINN predictions against a pseudo-spectral DNS (resolution 512^3) and 8th-order finite difference solvers at various resolutions (64^3–512^3). The PINN achieves accuracy comparable to high-order solvers at moderate resolution and captures key dynamical features of the flow. (d) Comparison of the iso-contours of the dimensionless vorticity norm on the periodic face $x = -\pi$ at $t = 8$.

Wang S, Sankaran S, Stinis P, et al. Simulating three-dimensional turbulence with physics-informed neural networks. arXiv preprint arXiv:2507.08972, 2025.

Optimizers Today

Approximate Gauss-Newton Methods

K-FAC (tensor approximation)

Approximate Newton Methods

Old Days: BFGS, L-BFGS,
Recently: Kron (low rank approximation+online linear regression)

Approximate Adagrad

Adam (diag approximation)
Shampoo (tensor approximation)
SOAP (Adam in spectral space)
One-side shampoo ….

Today

Steepest Descent in New Norm

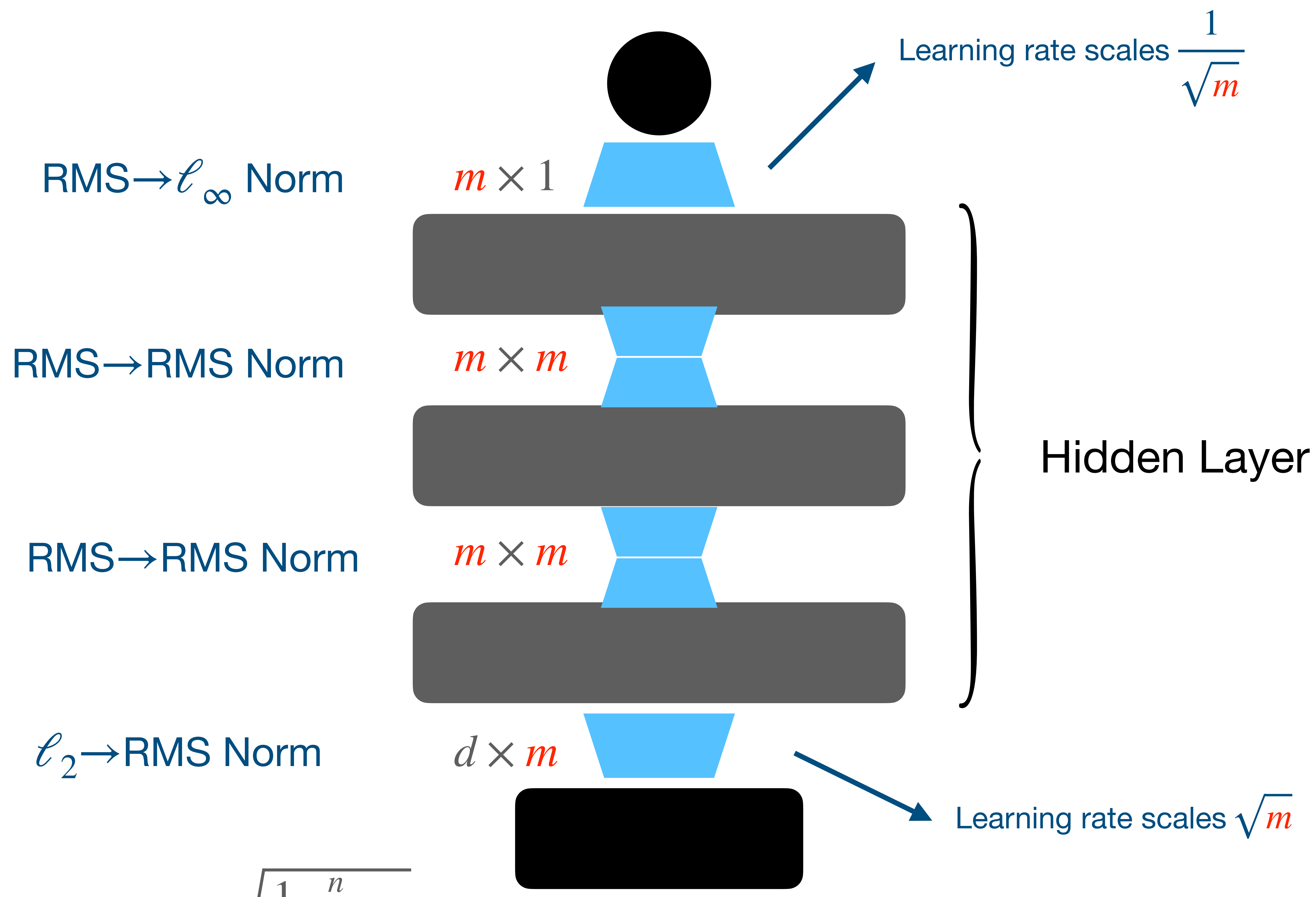
Maddison C J, Paulin D, Teh Y W, et al. Dual space preconditioning for gradient descent. SIAM Journal on Optimization, 2021

Steepest Descent in Different Norms

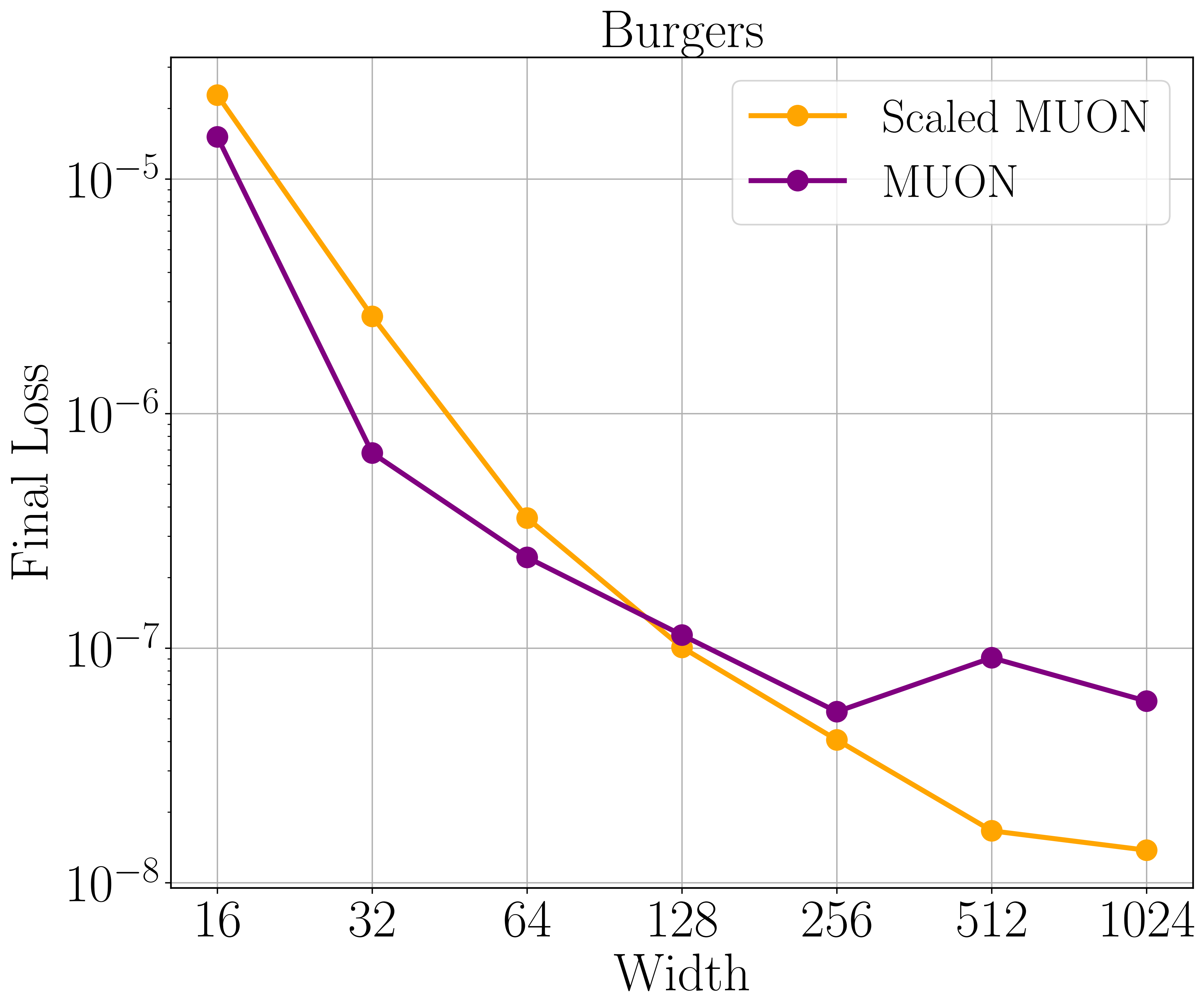
Update Direction: $\arg\max_{X} \langle G, X \rangle + \lambda \|G\|_{?}$

- SignSGD: $x_{t+1} = x_t - \lambda \text{Sign}(\nabla f(x_t)), \|G\|_{?} = \|G\|_{\infty}$
- MUON: $x_{t+1} = x_t - \lambda \text{MatrixSign}(\nabla f(x_t)), \|G\|_{?} = \|G\|_{\text{op}}$
    - Where $\text{MatrixSign}(U\Sigma V^\top) = UV^\top$
    - MatrixSign can be approximated by Newton-Schulz $X_{k+1} = \frac{1}{2} X_k \left(3I - X_k^\top X_k\right)$

The Norm We Select



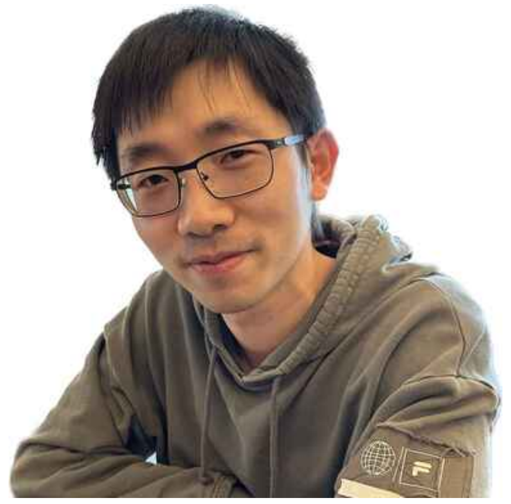
$$\|x\|_{\text{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$



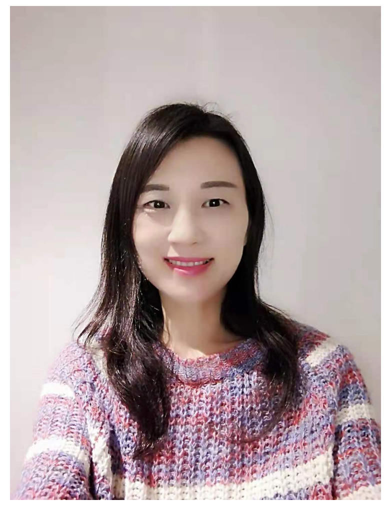
AIM of our paper

A Numiercal Scaling Law for PINN

Jasen Lai (UF)

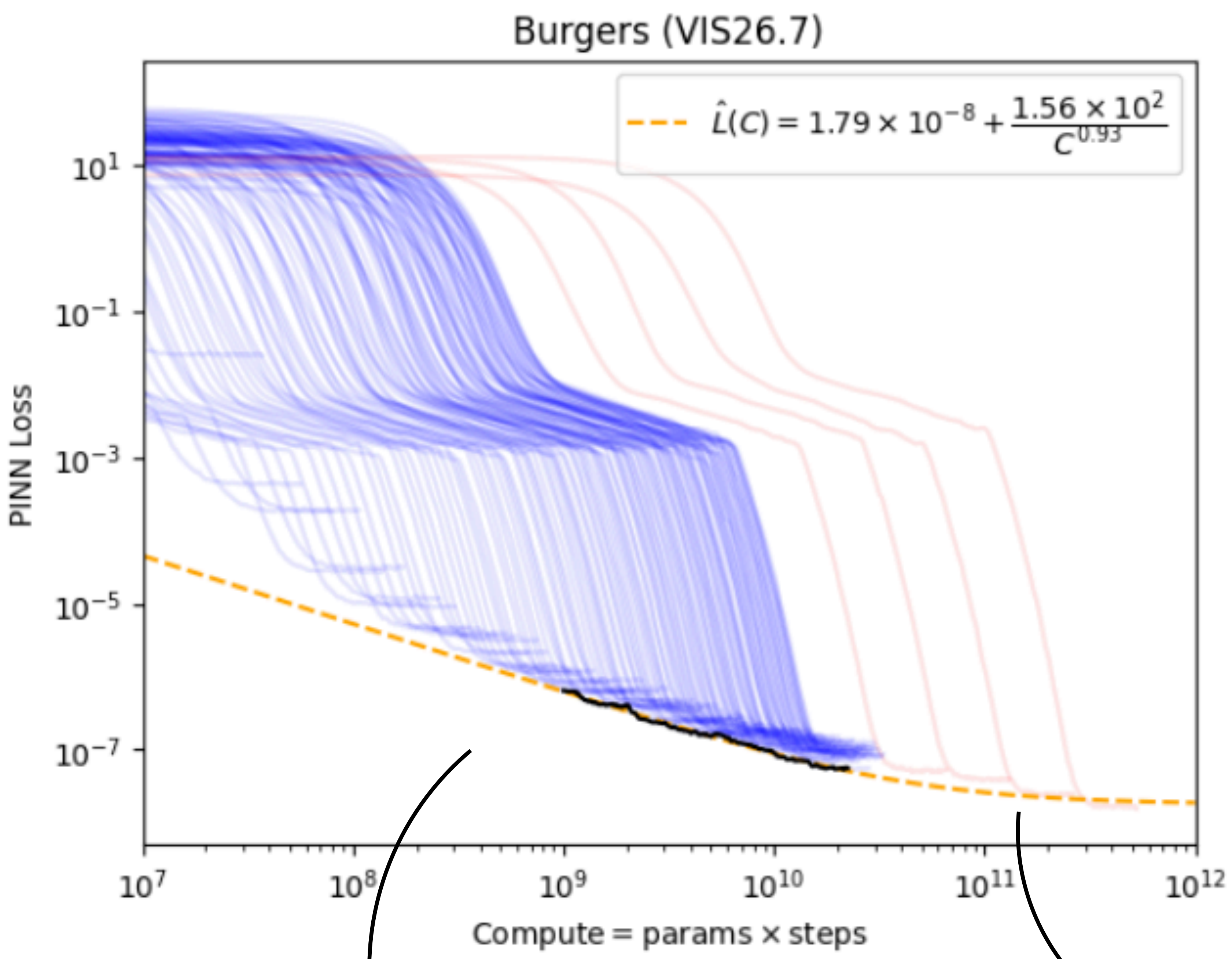


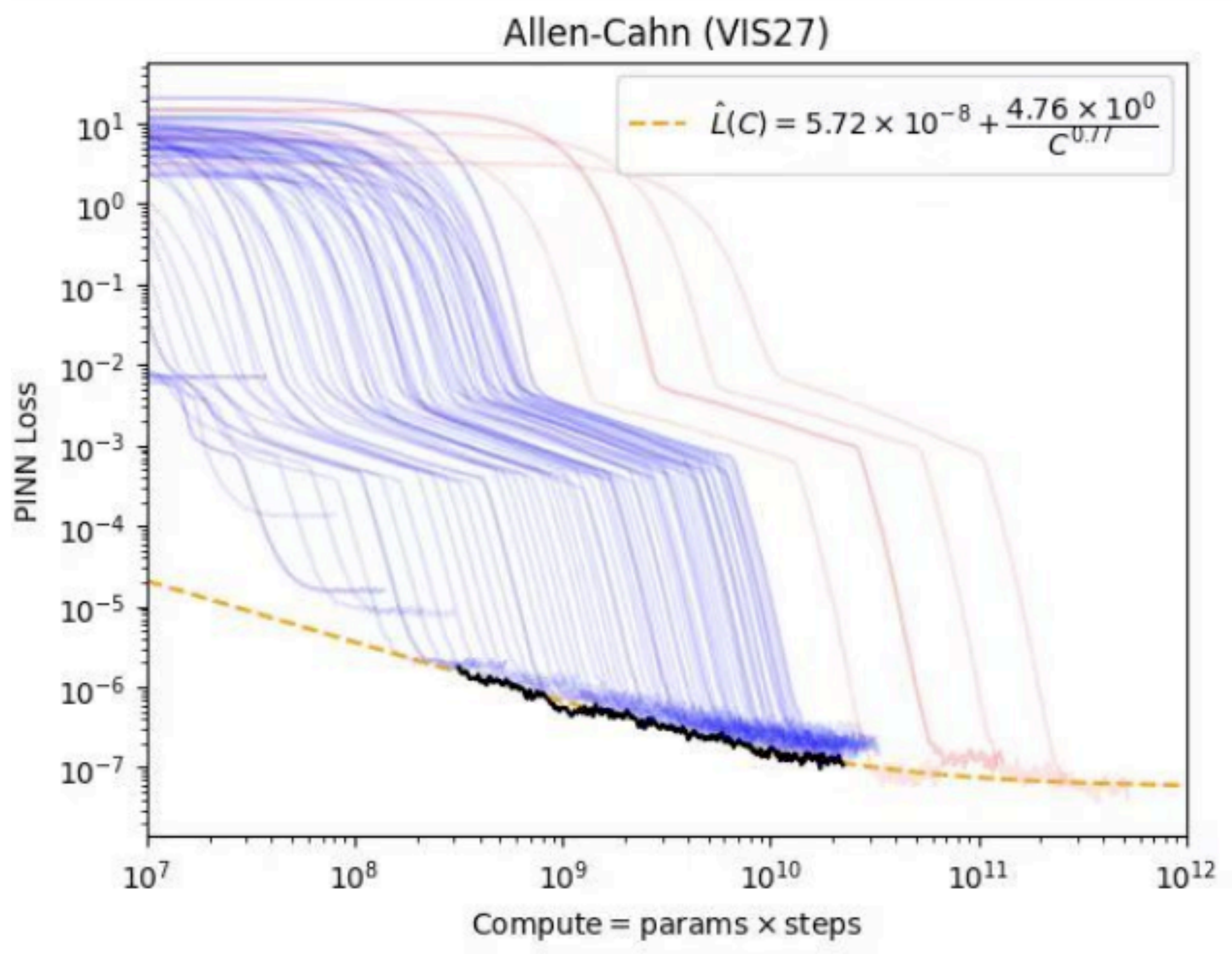
Sifan Wang (Yale)

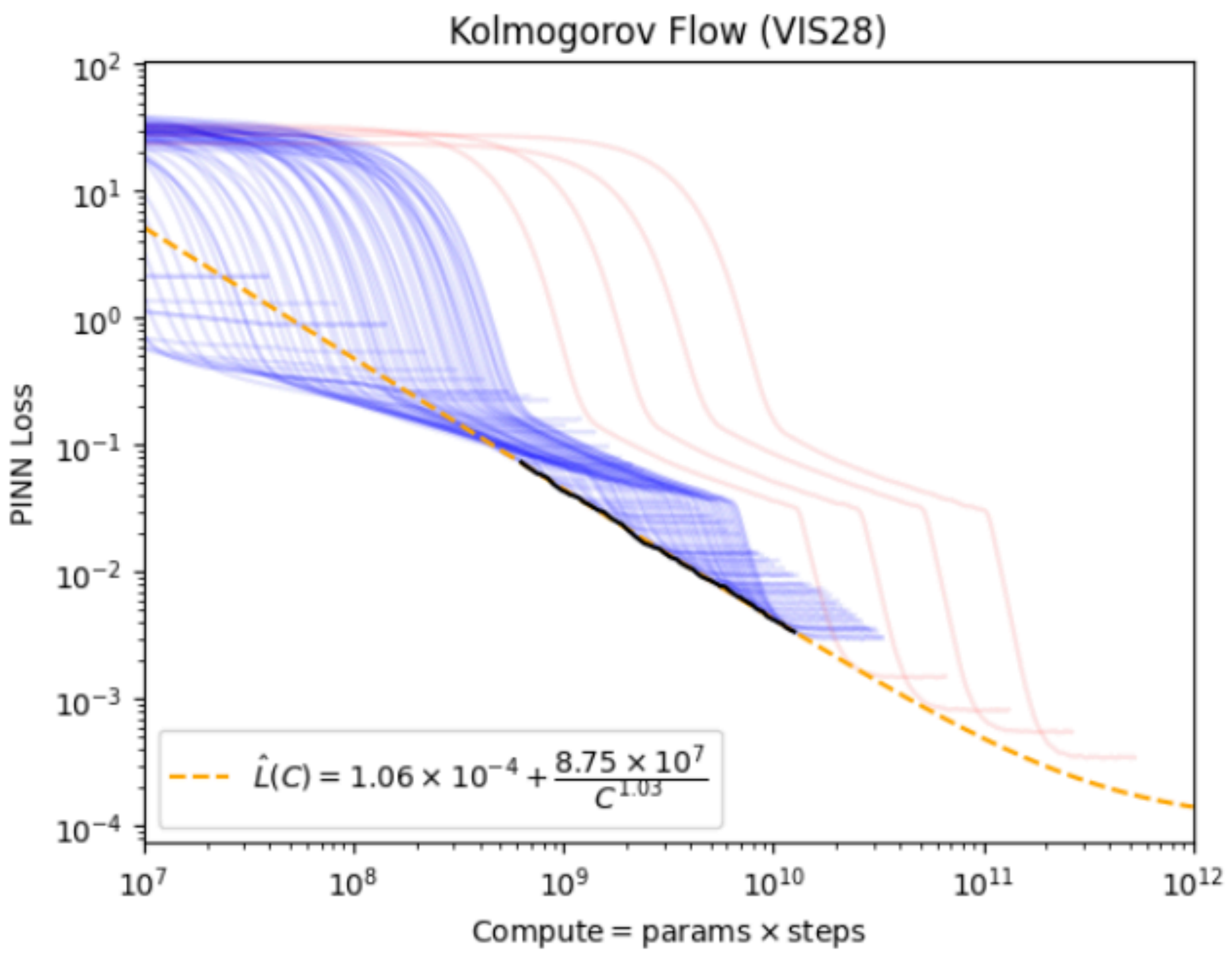


Chunmei Wang (UF)

All Equation 2 dim in space and 1 dim in time







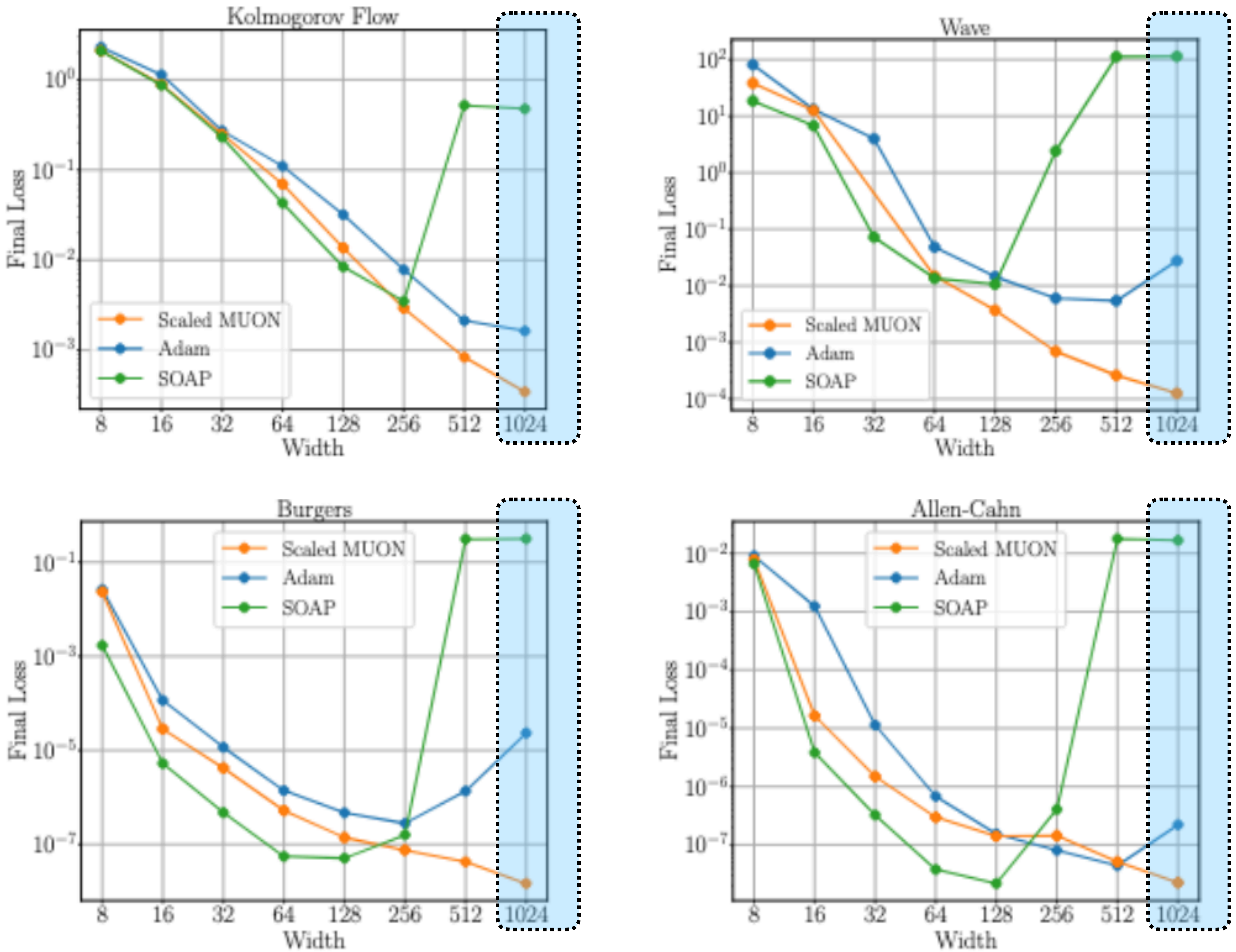
We can predict the behaviour of larger networks

Different line means different widths
Use small scale to estimate the scaling law

$\text{Error} \propto 1/\sqrt{\text{Compute}}$

Key Component: MUON Optimizer

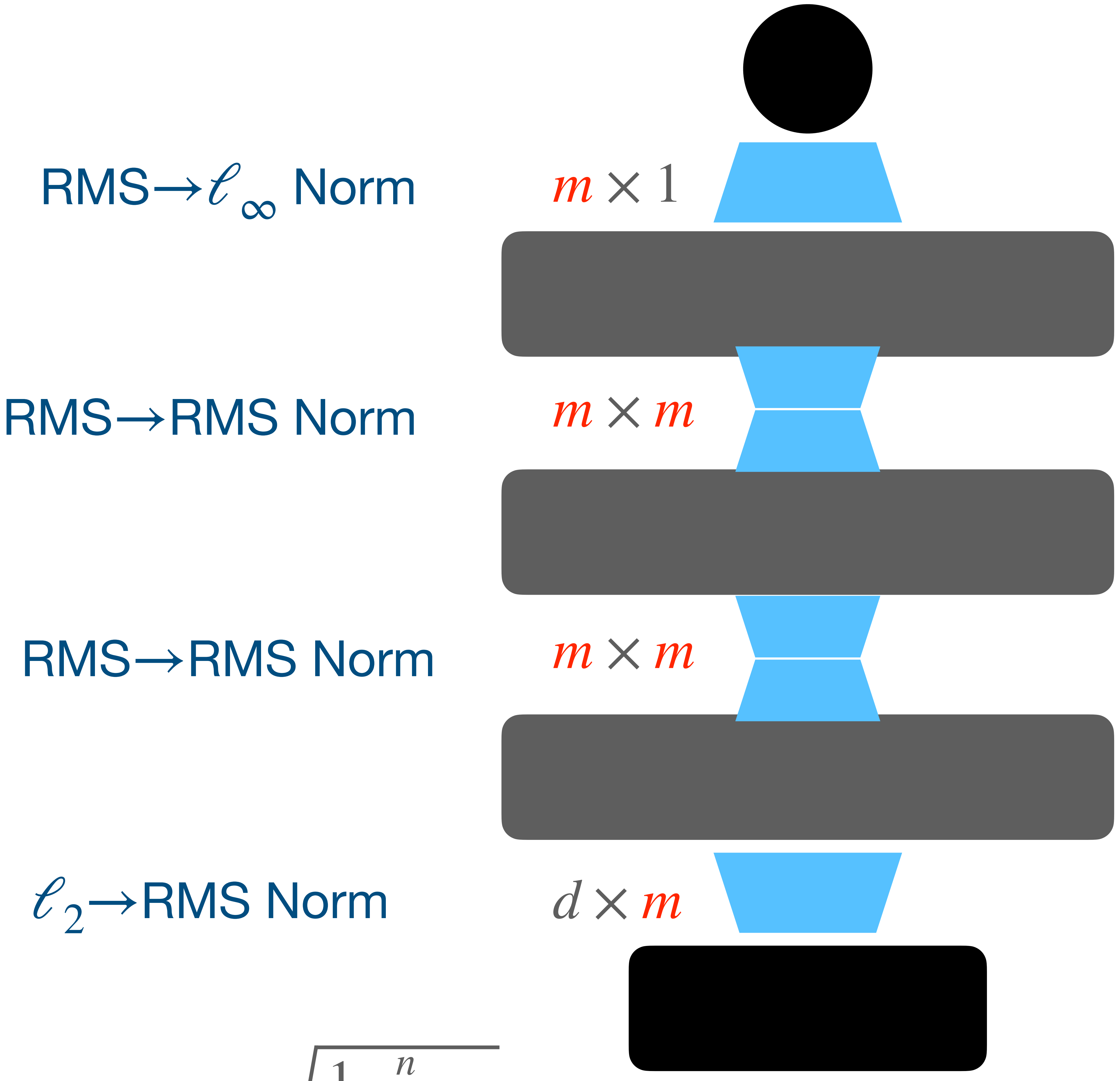
Sclae leads to better results



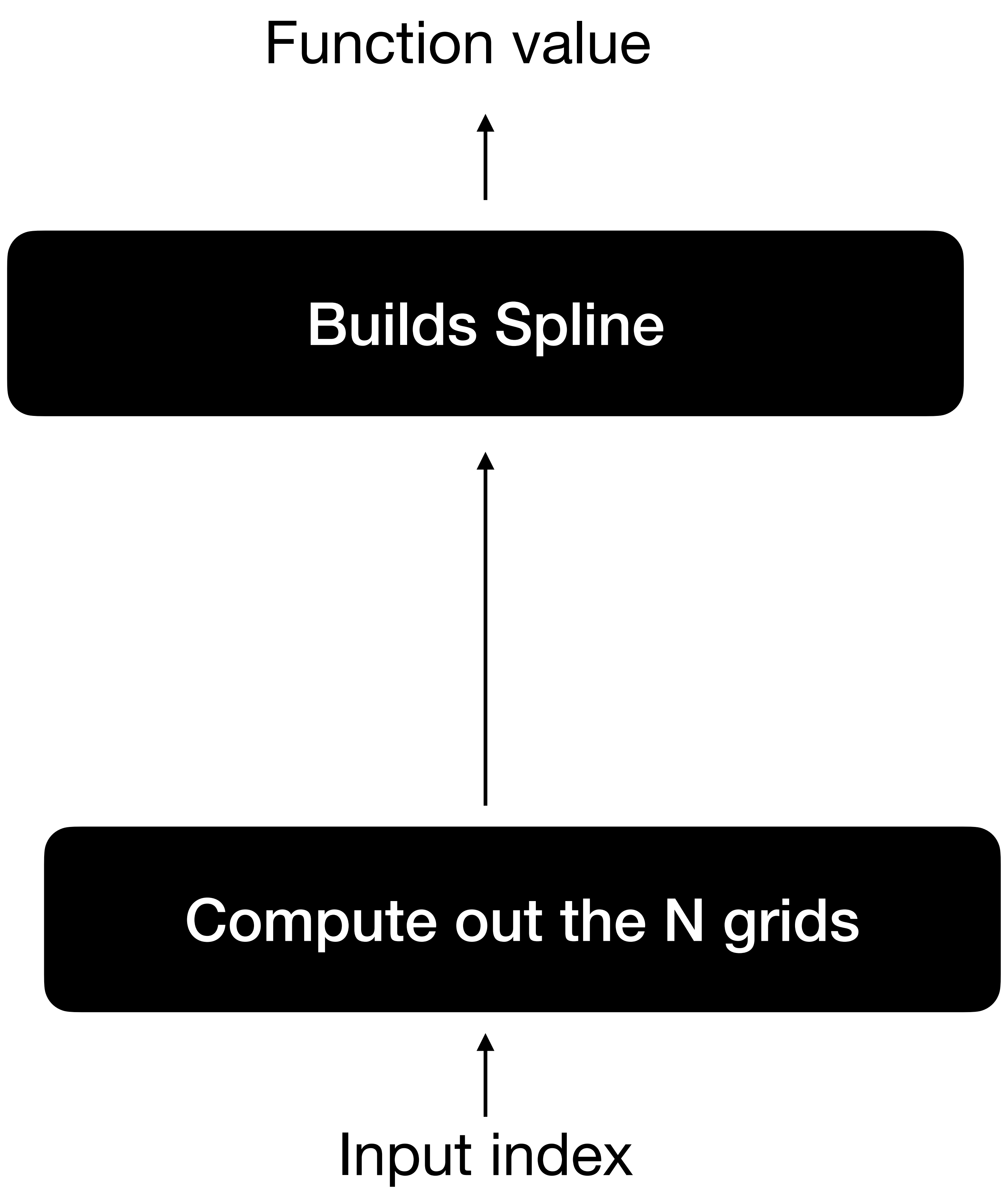
Method	Width	Depth
Vanilla PINN (Raissi et al., 2019)	20–40	5–8
Fourier PINNs (Wang et al., 2021)	128–256	3-5
FBPINNs (Moseley et al., 2023)	16–64	2–5
SPINN (Cho et al., 2023)	32-256	3-4
Causal PINNs (Wang et al., 2024b)	128–256	3-5
SA-PINNs (McClenny & Braga-Neto, 2023)	50–128	4–6
RBA-PINNs (Anagnostopoulos et al., 2023)	128–256	4–6
Curriculum training (Krishnapriyan et al., 2021)	50	4
Natural gradient descent Müller & Zeinhofer (2023); Chen et al. (2024)	20–40	1-3
SSBroyden (Urbán et al., 2025; Kiyani et al., 2025)	20-40	2-6
SOAP (Wang et al., 2025)	256	6-12

Table 1: Representative PINN methods and typical network architectures (width = neurons per hidden layer, depth = number of hidden layers). Exact sizes may vary per problem; ranges indicate commonly reported configurations.

The Norm is Good for Approximation



$$\|x\|_{\text{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$



Scaled Norm: O(1)
F Norm: O(m)
ℓ_∞ Norm: O(m)

Smaller norm is always better?

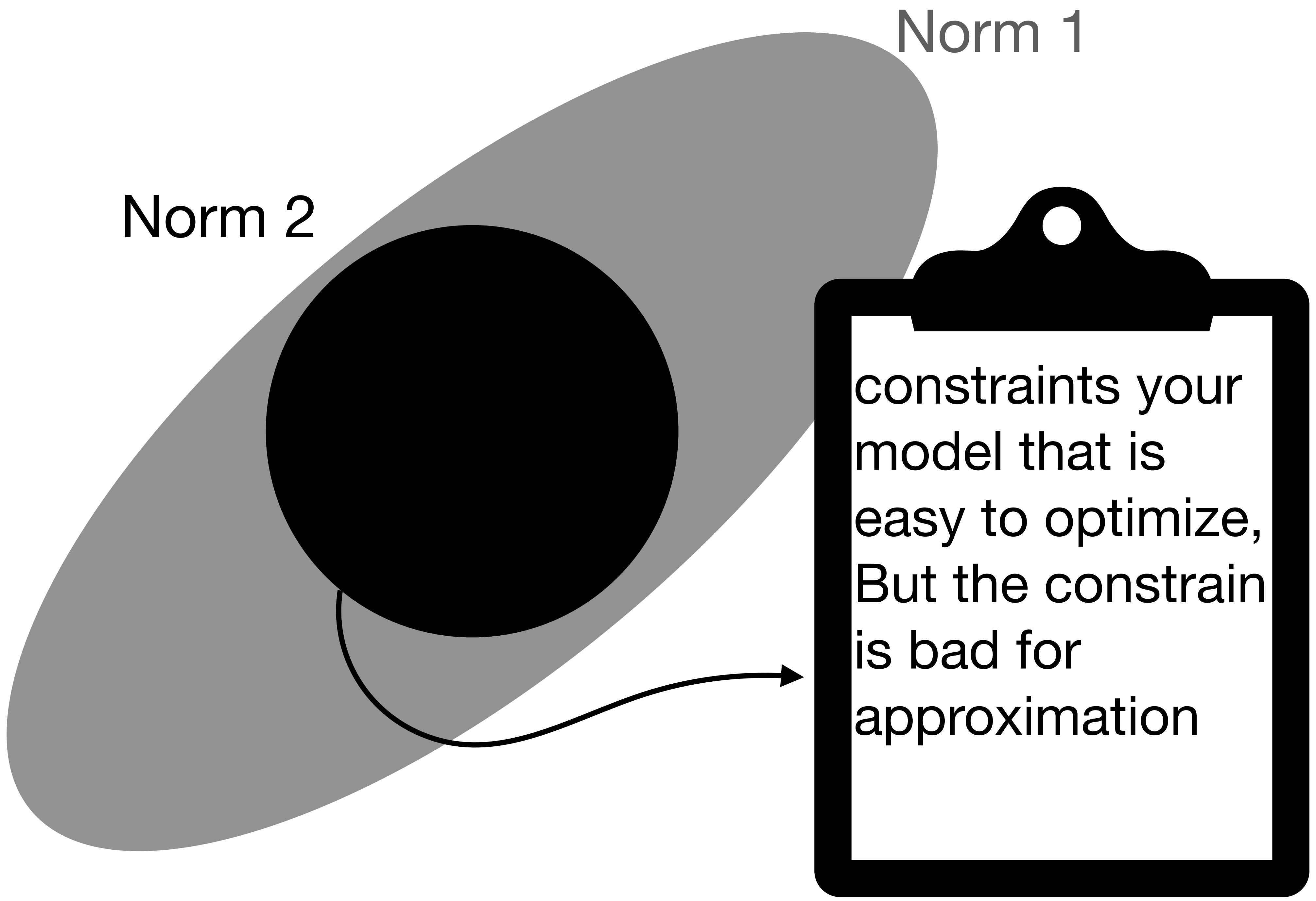
Trade-off: Approximation vs Optimization

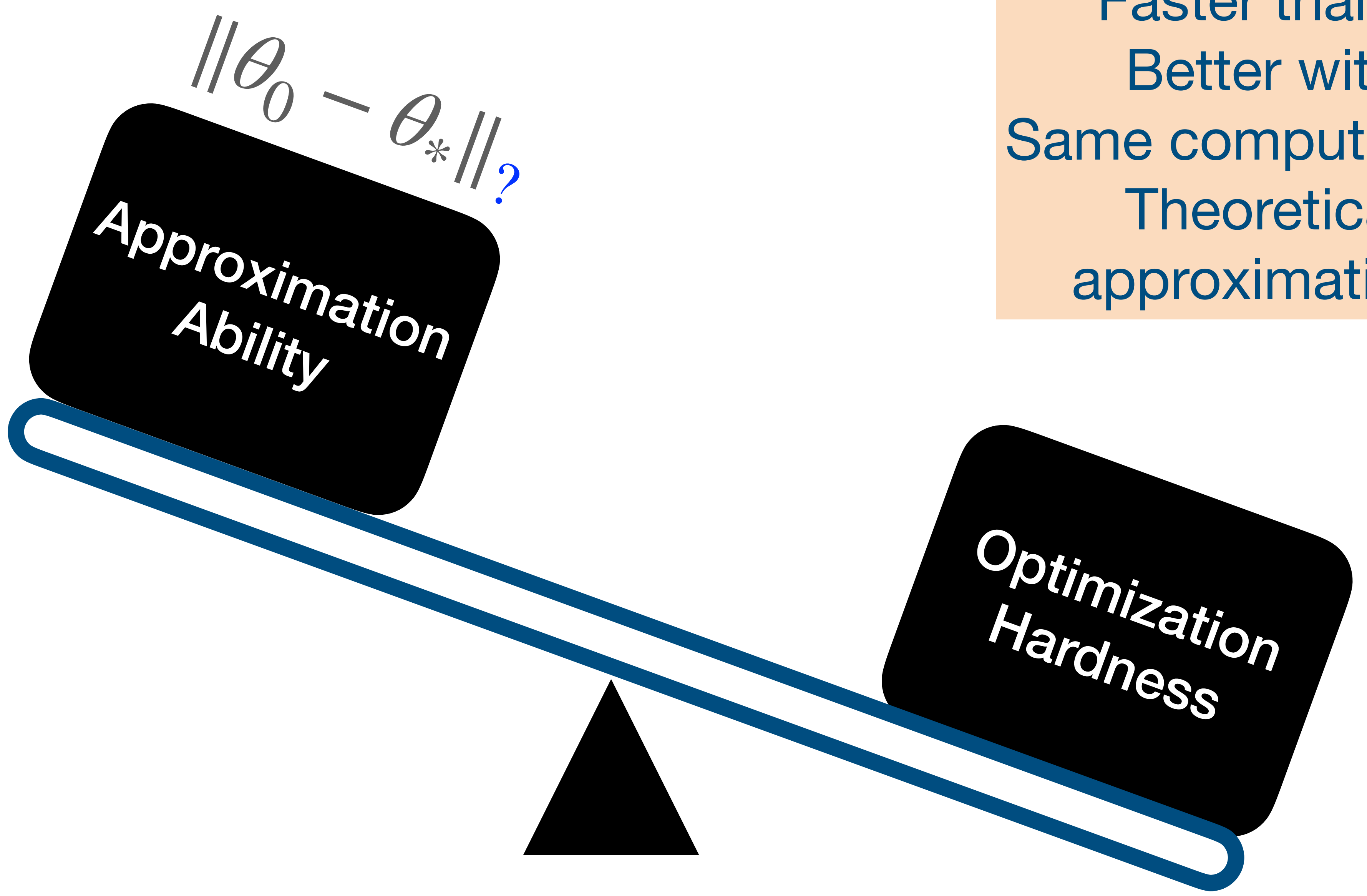
- Optimization Theory:
    - If we need Steepest Descent in $\|\cdot\|_?$, we need relative smoothness $\|f(X) - f(Y) - \nabla f(Y)(X - Y)\| \leq \boxed{L}\|D_h(X) - D_h(Y) - \nabla D_h(Y)(X - Y)\|$

Larger norm is always better? Larger nomr -> better relative smoothness

Optimization and approximation Trade-off

Norm-Constrained Approximation Theory → **Optimization Theory** → A New Computable Norm ! Ask me 1 month later!



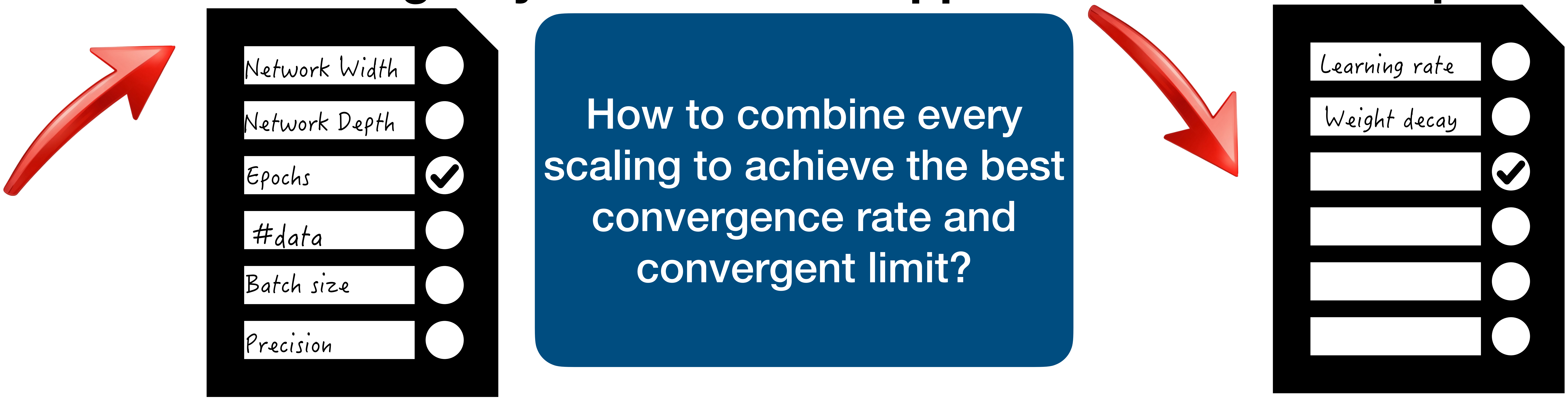


50% smaller memory use
Faster than Adamw on CIFAR
Better with MUON on PINN
Same computational cost as Adamw
Theoretically "width" free in approximation and optimization

Jiajin Li (UBC)

Tons of chances and the unreasonable effectiveness of mathematics

Think in a limiting way: what would happen if we had ∞ compute



Scaling learning rate schleuder: Schaipp F, Hägele A, Taylor A, et al. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. arXiv:2501.18965
Scaling Weight Decay achieves 17.5x more data efficiency: Kim K, Kotha S, Liang P, et al. Pre-training under infinite compute. arXiv:2509.14786, 2025.
Training precision: Kumar T, Ankner Z, Spector B F, et al. Scaling laws for precision. arXiv preprint arXiv:2411.04330
Scaling Depth: Dey N, Zhang B C, Noci L, et al. Don't be lazy: CompleteP enables compute-efficient deep transformers arXiv:2505.01618
Scaling Batch size: Bergsma S, Dey N, Gosal G, et al. Power lines: Scaling laws for weight decay and batch size in llm pre-training. arXiv preprint arXiv:2505.13738, 2025.

Scaling at Inference Time

Consider How you use ChatGPT…

What is the most efficient route from Times Square to Chelsea Market that minimizes walking?

Option 1: Subway

1. **Walk to a Nearby Subway Entrance:** Head to the Times Sq–42nd Street station.
2. **Board the E Train:** Catch an E train heading downtown (toward World Trade Center).
3. **Ride to 8th Avenue–14th Street Station:** Get off at this station (roughly 4 stops).
4. **Walk to Chelsea Market:** Exit the station and walk east on 14th Street for a few blocks until you reach Chelsea Market at 75 9th Avenue.

double-check with a map

1. **At Times Square:** Enter the Times Square–42nd Street station.
2. **Board the 1 Train:** Hop on a downtown 1 train (the red line).
3. **Ride to 14th Street:** Stay on until you reach the 14th Street station.
4. **Exit Appropriately:** Use the exit that leads toward 9th Avenue—this drop-off point is just a short walk from Chelsea Market (located at 75 9th Ave).

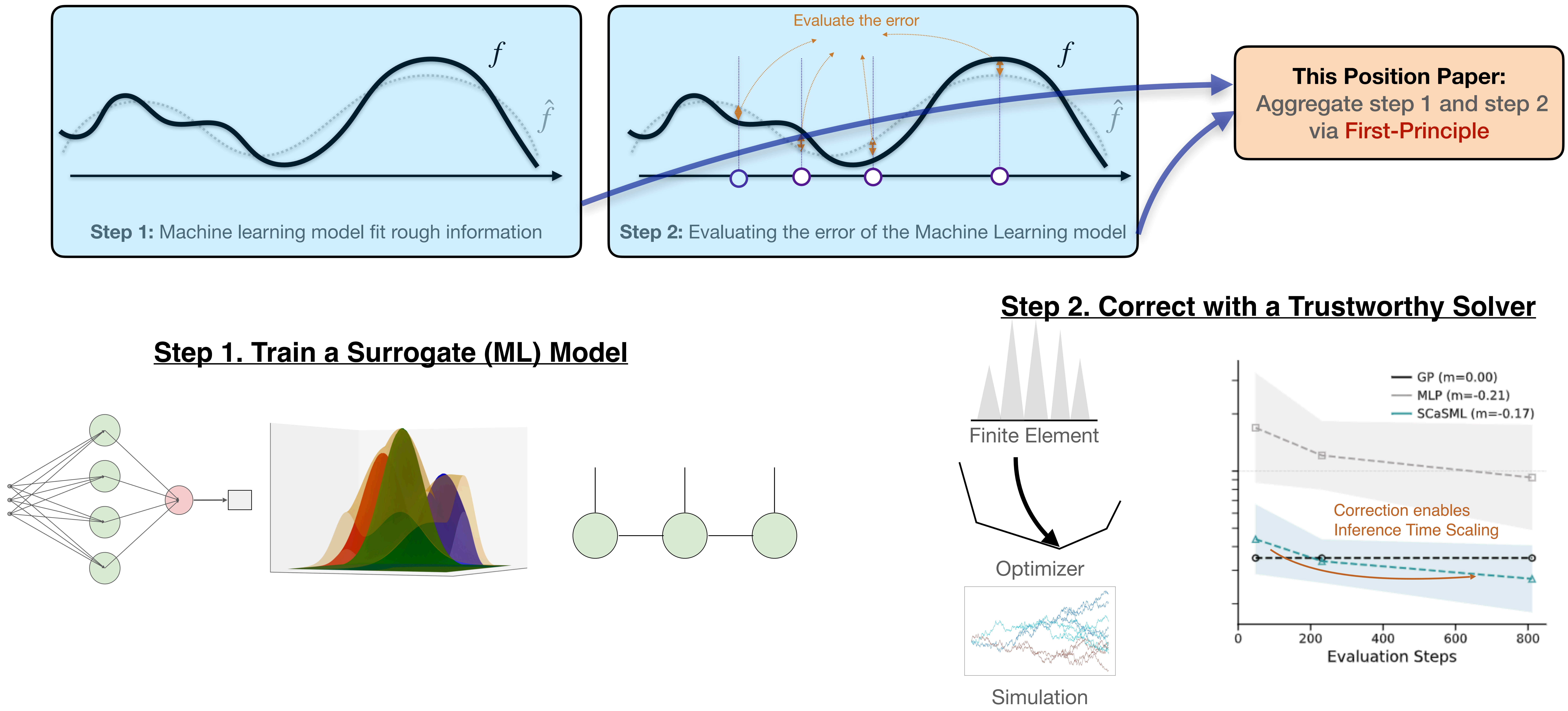
Inference Time Computing in LLM

https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute

How can we perform Inference-Time Scaling for Scientific Machine Learning?

With trustworthy garuntee

Physics-Informed Inference Time Scaling



The 101 Example

Haoxuan Chen, Lexing Ying, Jose Blanchet

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

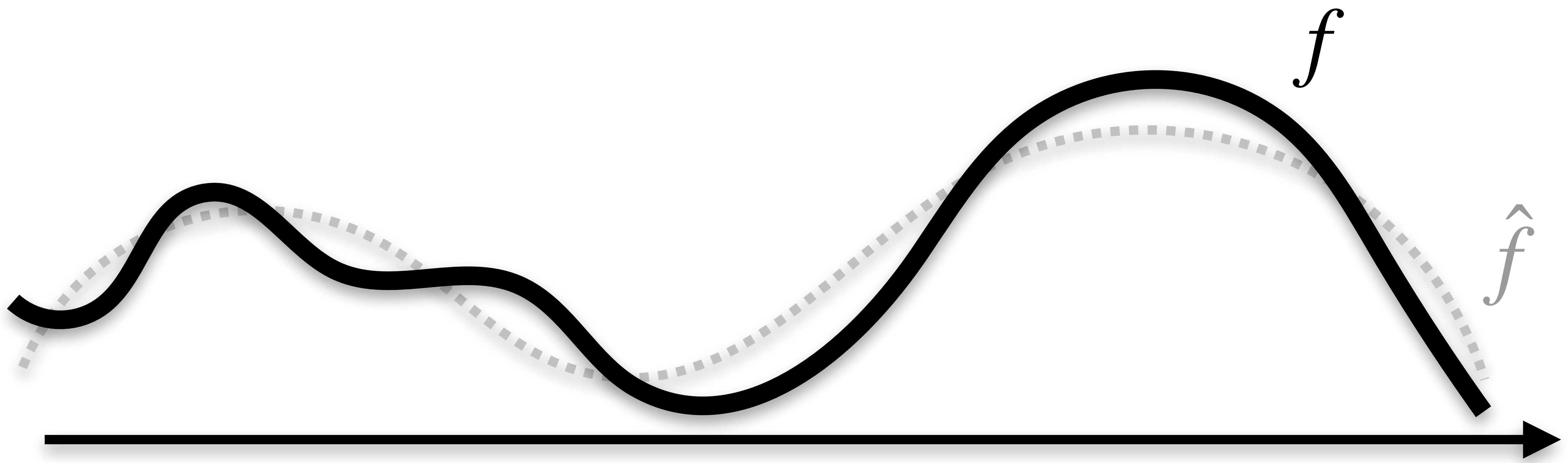
$$\theta = f, \quad \underbrace{X_i =}\ (x_i, f(x_i))$$

$$\Phi(\theta) = \int (f(x))dx$$

Machine Learning: $\hat{\theta} = \hat{f}$ → $\Phi(\hat{\theta}) = \int \hat{f}(x)dx$

Simpson's Rule

When $\hat{f}$ is piecewise poly



The 101 Example

Faster and Optimal convergence than both qudrature rule and Monte Carlo

$$\underbrace{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}}_{\text{Scientific Machine Learning}} \to \underbrace{\Phi}_{\text{Downstream application}}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\theta = f, \quad \underbrace{X_i = }(x_i, f(x_i))$$

$$\Phi(\theta) = \int (f(x))dx$$

$$\|$$

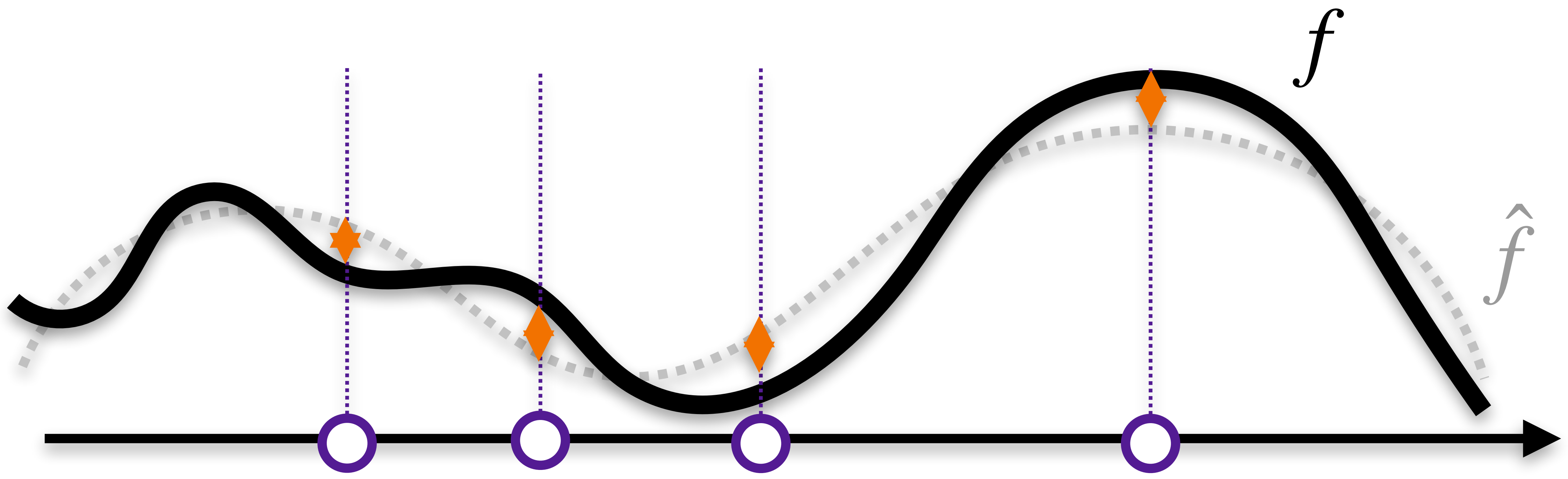
Machine Learning: $\hat{\theta} = \hat{f}$

$$\Phi(\hat{\theta}) = \int \hat{f}(x)dx$$

$$+$$

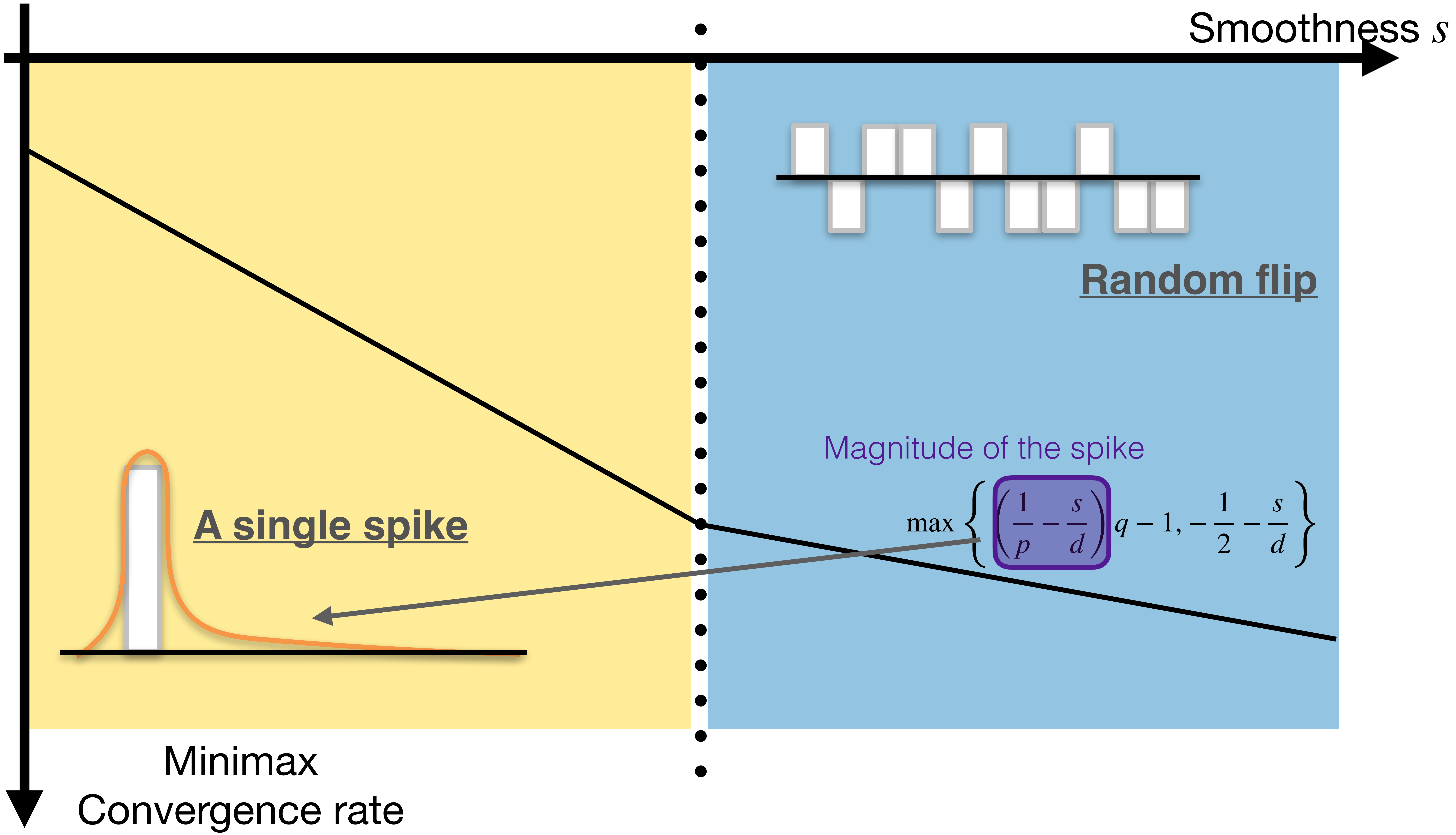
$$\Phi(\theta) - \Phi(\hat{\theta}) = \underbrace{\int (f(x) - \hat{f}(x))dx}$$

Using Monte Carlo Methods to approximate



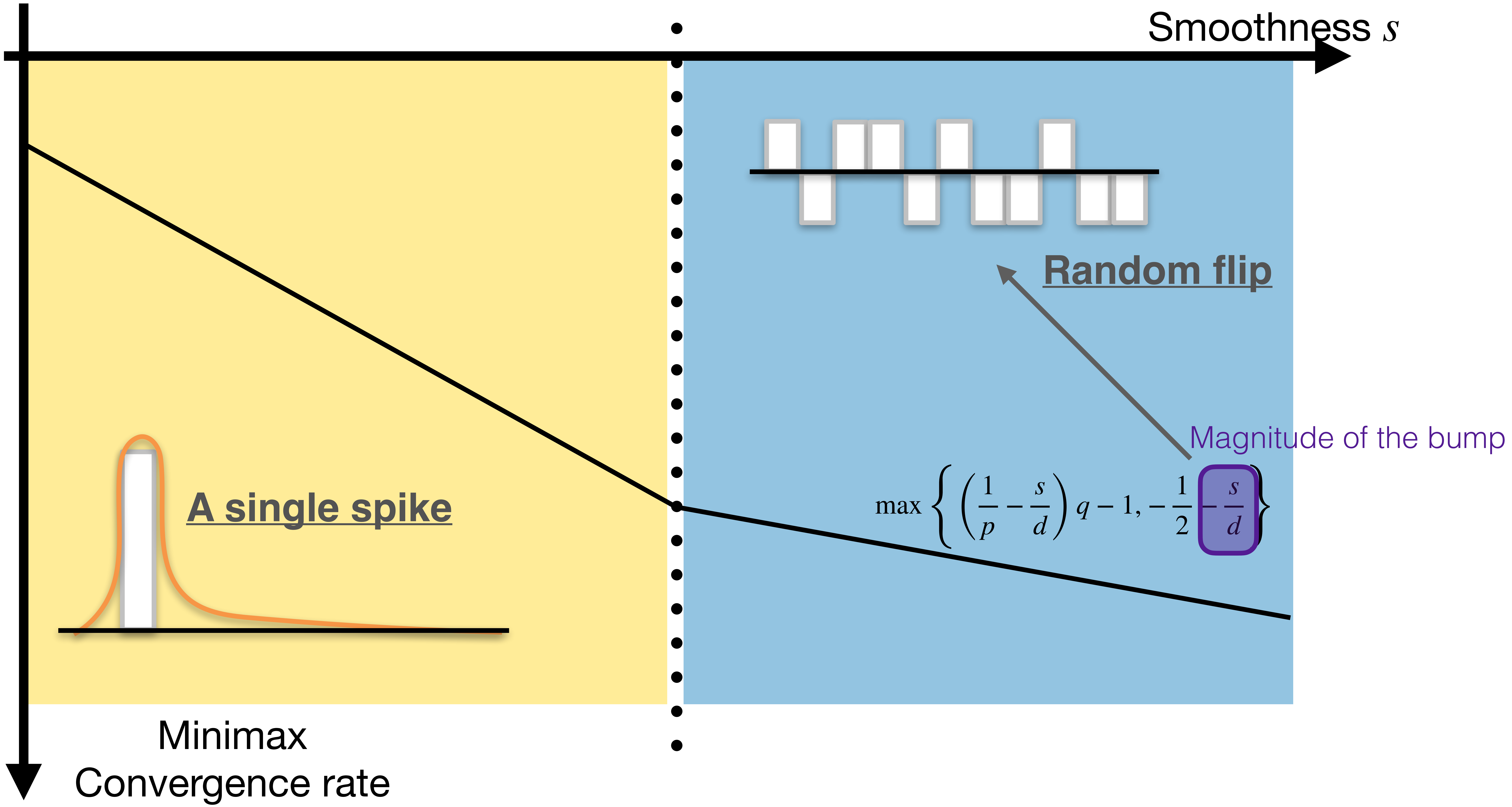
Estimate $\int |f(x)|^q dx, \quad f \in W^{s,p}$

Lower Bound



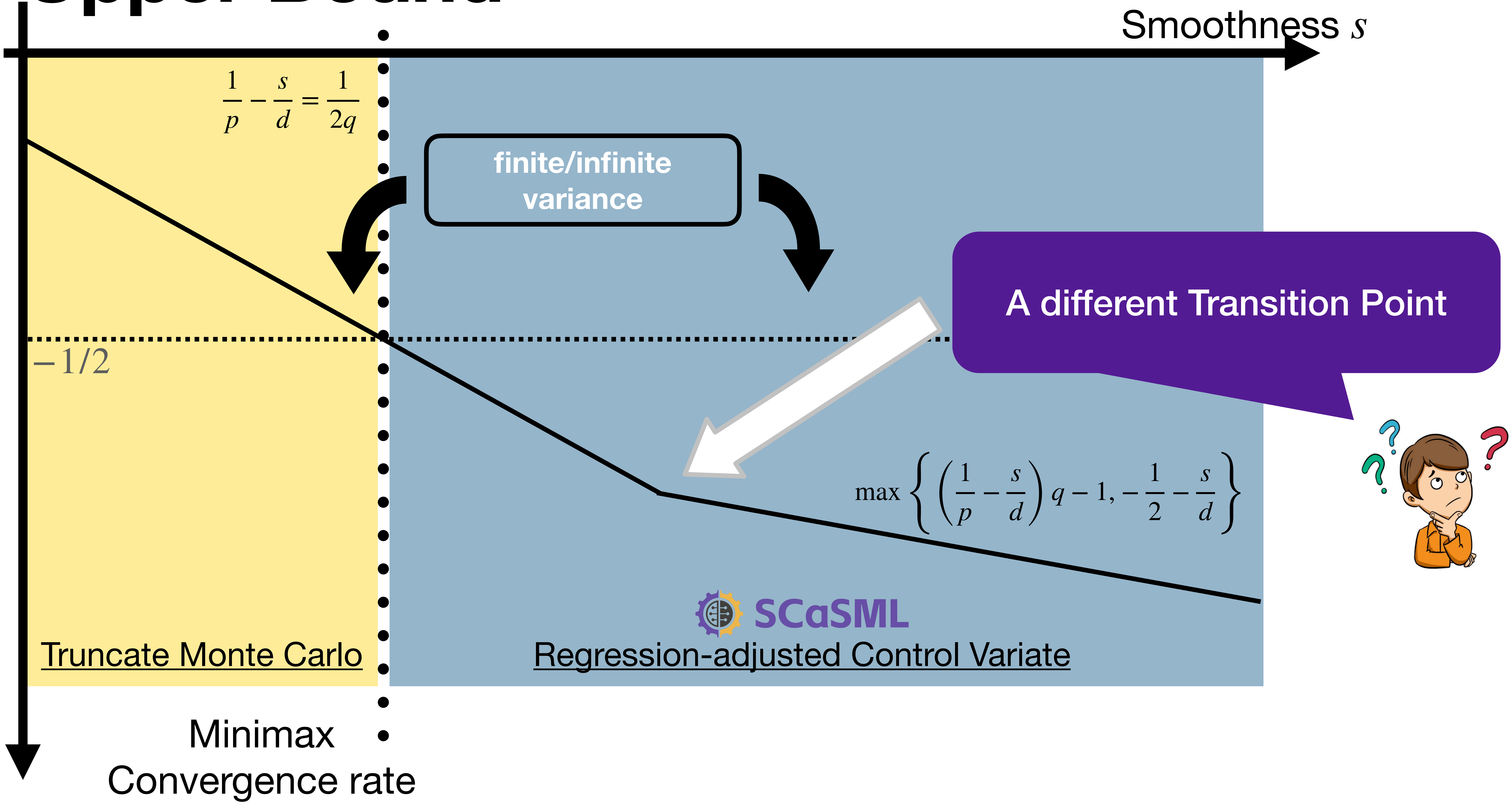
Estimate $\int |f(x)|^q dx, \quad f \in W^{s,p}$

Lower Bound



Estimate $\int |f(x)|^q dx, \quad f \in W^{s,p}$

Upper Bound



Analysis of Error propagation

SCaSML

SCaSML estimate of $\mathbb{E}_P f^q, f \in W^{s,p}$

Step 1 Using half of the data to estimate $\hat{f}$

Step 2 $\mathbb{E}_P f^q = \mathbb{E}_P(\hat{f}^q) + \mathbb{E}_P(f^q - \hat{f}^q)$

How does step2 variance depend on estimation error?

Hardness = The variance of the debasing step

Analysis of Error propagation

SCaSML

SCaSML estimate of $\mathbb{E}_P f^q, f \in W^{s,p}$

Step 1 Using half of the data to estimate $\hat{f}$

Step 2 $\mathbb{E}_P f^q = \mathbb{E}_P(\hat{f}^q) + \mathbb{E}_P(f^q - \hat{f}^q)$

How does step2 variance depend on estimation error?

Low order term: $f^q - \hat{f}^q$

$f^{q-1}(f - \hat{f}) + (f - \hat{f})^q$

"influnce function" (gradient) **Error propagation**

Yiping Lu yiping.lu@northwestern.edu

Analysis of Error propagation

SCaSML

SCaSML estimate of $\mathbb{E}_P f^q, f \in W^{s,p}$

Step 1 Using half of the data to estimate $\hat{f}$

Step 2 $\mathbb{E}_P f^q = \mathbb{E}_P(\hat{f}^q) + \mathbb{E}_P(f^q - \hat{f}^q)$

Low order term

$f^{q-1}(f - \hat{f}) + (f - \hat{f})^q$

"influence function" (gradient)

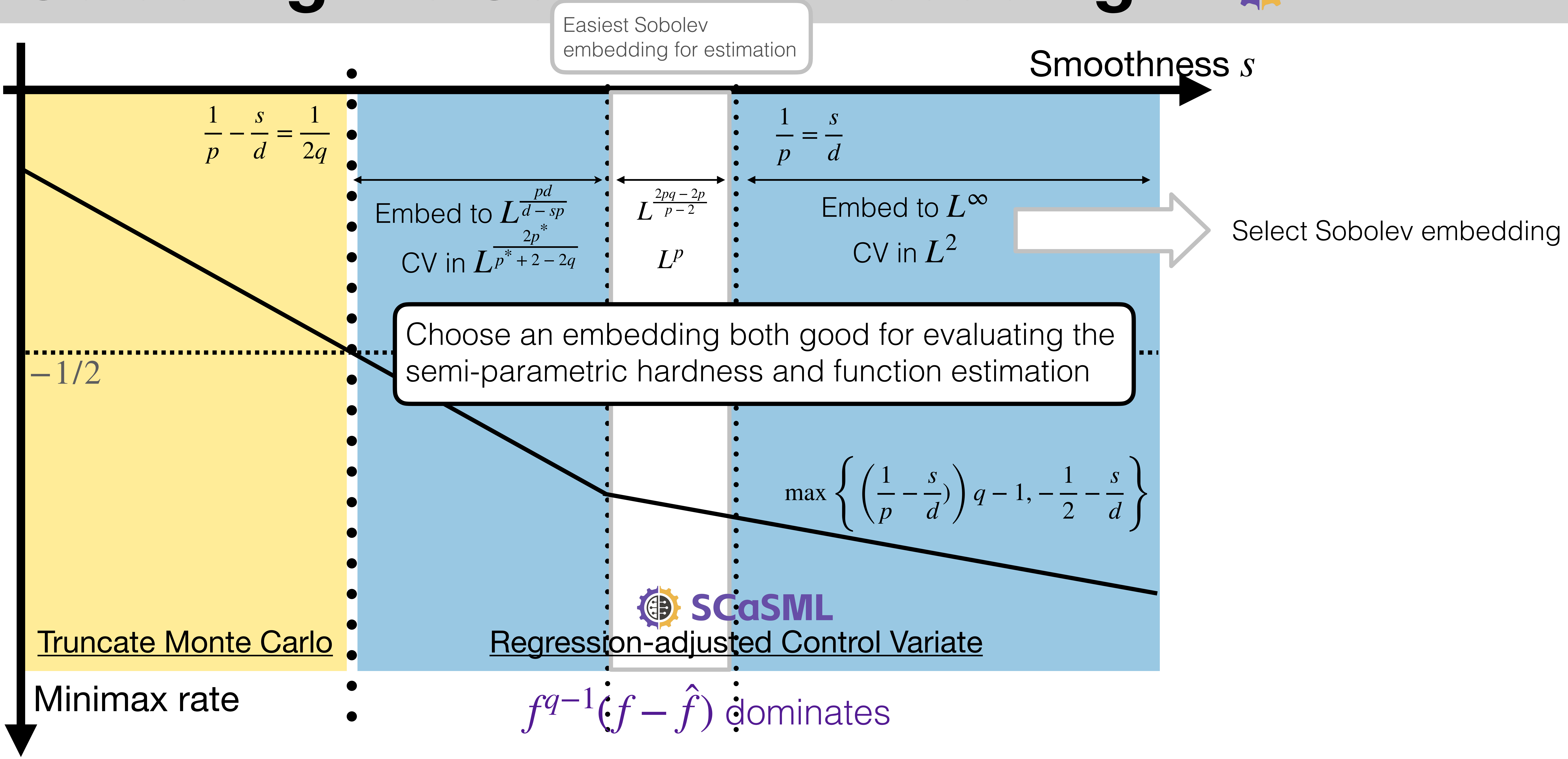
Error p

Embed f^{q-1} and $f - \hat{f}$ into "dual" space

How to select the Sobolev emebedding?

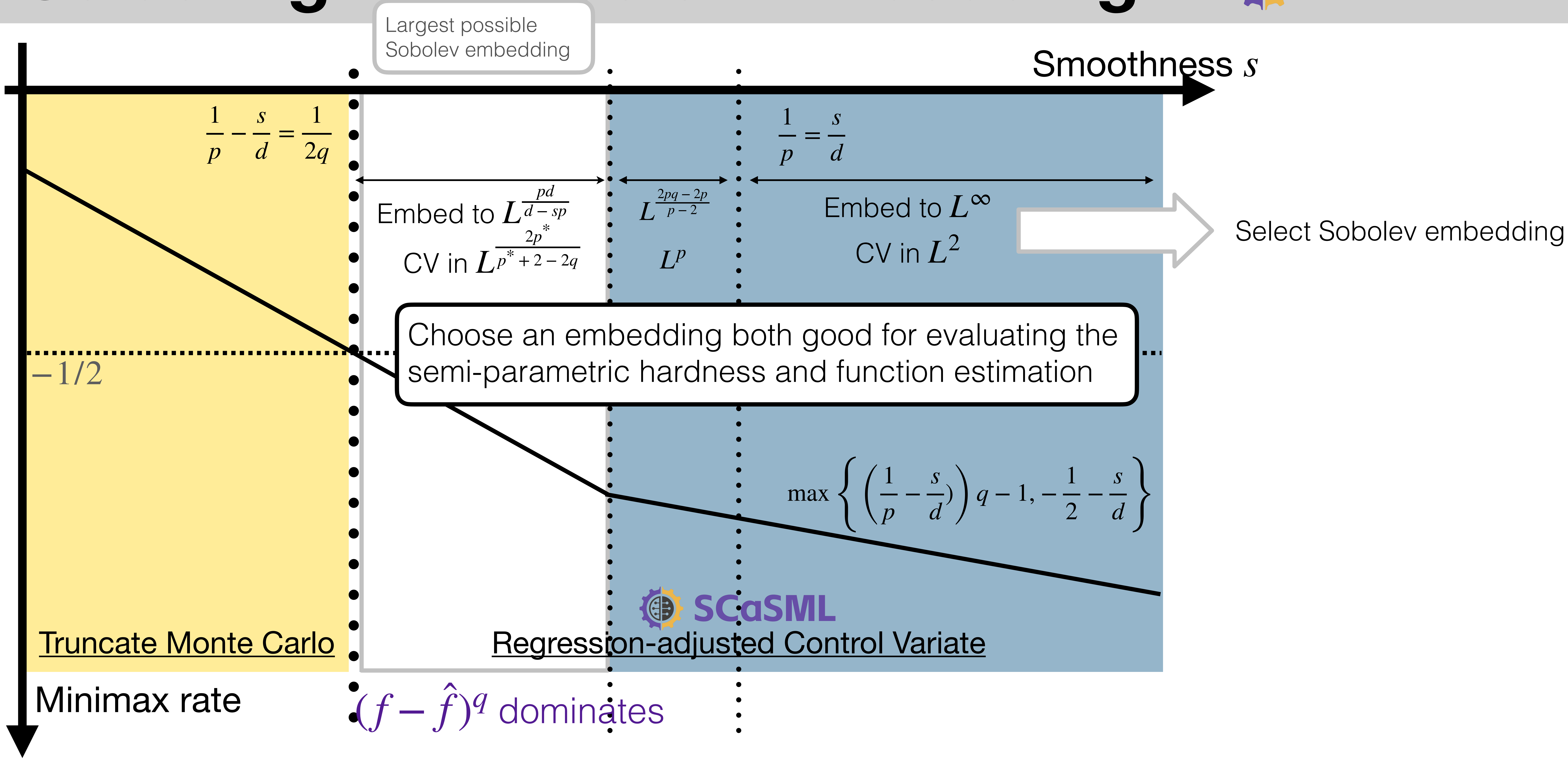
Selecting the Sobolev Embedding

SCaSML



Yiping Lu yiping.lu@northwestern.edu

Selecting the Sobolev Embedding



Yiping Lu yiping.lu@northwestern.edu

Neurips 2023

When can Regression-Adjusted Control Variates Help?
Rare Events, Sobolev Embedding and Minimax Optimality

Jose Blanchet
Department of MS&E and ICME
Stanford University
Stanford, CA 94305
jose.blanchet@stanford.edu

Haoxuan Chen
ICME
Stanford University
Stanford, CA 94305
haoxuanc@stanford.edu

Yiping Lu
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012
yiping.lu@nyu.edu

Lexing Ying
Department of Mathematics and ICME
Stanford University
Stanford, CA 94305
lexing@stanford.edu

PDE Solver

The PDE Example

Let's consider $\Delta u = f$

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

$$\theta = u, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = u(x)$$

What is $\Phi(\theta) - \Phi(\hat{\theta}) = u(x) - \hat{u}(x)$?

FEM/PINN/DGM/Tensor/Sparse Grid/…:

$$\hat{\theta} = \hat{u} \longrightarrow \Phi(\hat{\theta}) = \hat{u}(x)$$

The **PDE** Example

Let's consider $\Delta u = f$

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning — Downstream application

$$\theta = u, \quad \underbrace{X_i = (x_i, f(x_i))}$$

$$\Phi(\theta) = u(x)$$

What is $\Phi(\theta) - \Phi(\hat{\theta}) = u(x) - \hat{u}(x)$?

$\Delta u = f$

−

$\Delta \hat{u} = \hat{f}$

=

$$\Delta(u - \hat{u}) = f - \hat{f}$$

FEM/PINN/DGM/Tensor/Sparse Grid/…:

$\hat{\theta} = \hat{u} \longrightarrow \Phi(\hat{\theta}) = \hat{u}(x)$

The PDE Example

Let's consider $\Delta u = f$

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning — Downstream application

$$\Delta u = f \qquad \theta = u, \quad \underbrace{X_i = (x_i, f(x_i))} \qquad \Phi(\theta) = u(x)$$

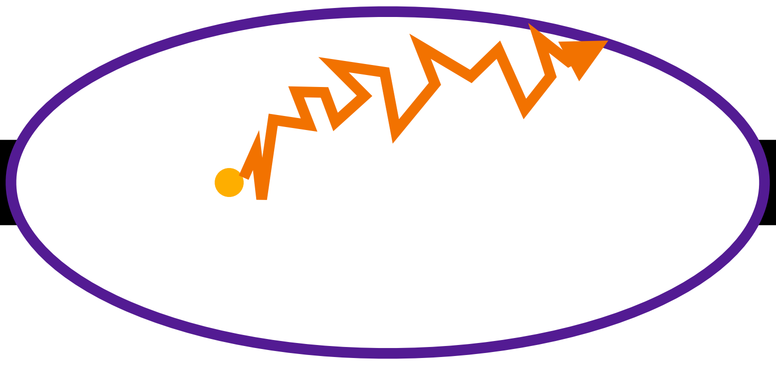
What is $\Phi(\theta) - \Phi(\hat{\theta}) = u(x) - \hat{u}(x)$?

$$\Delta \hat{u} = \hat{f}$$

FEM/PINN/DGM/Tensor/Sparse Grid/…:

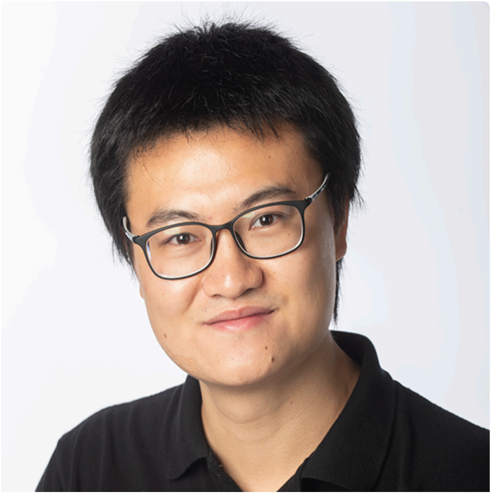
$$\hat{\theta} = \hat{u} \longrightarrow \Phi(\hat{\theta}) = \hat{u}(x)$$

$$\Delta(u - \hat{u}) = f - \hat{f}$$

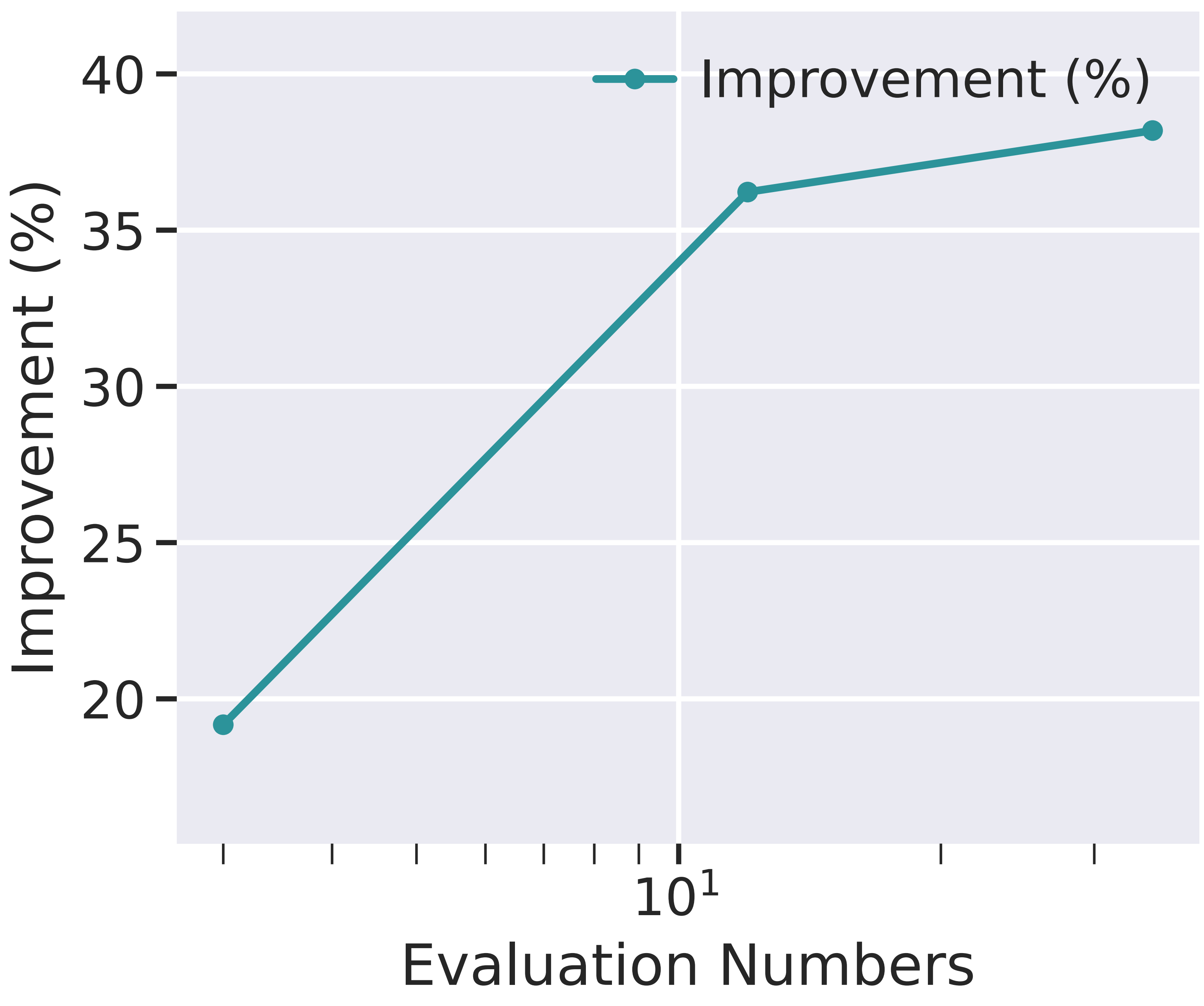


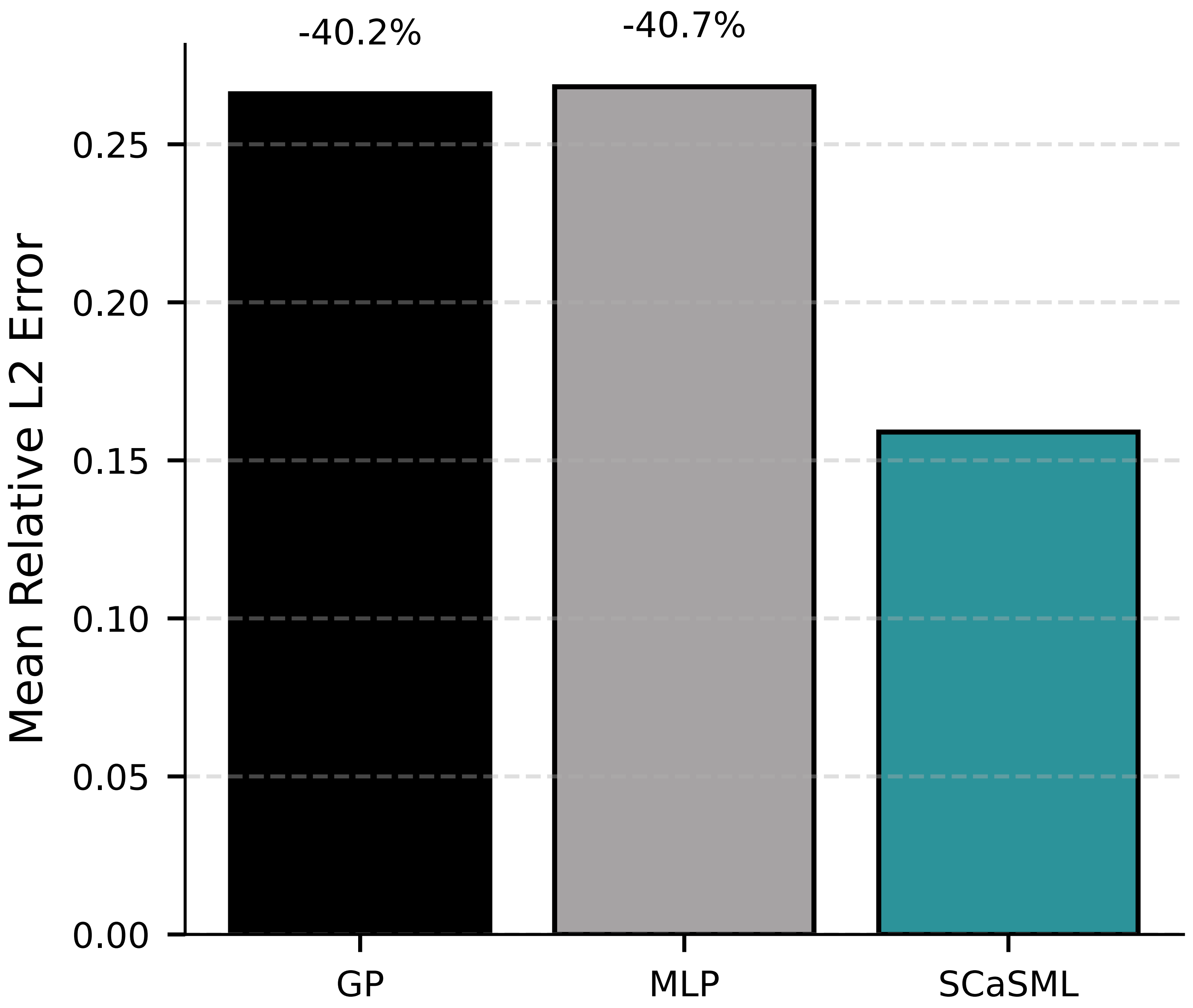
$$(u - \hat{u})(x) = \mathbb{E}\int (f - \hat{f})(X_t)\,dt$$

Inference-Time Scaling



$$\frac{\partial}{\partial t}u + \left[\sigma^2 u - \frac{1}{d} - \frac{\bar{\sigma}^2}{2}\right](\nabla \cdot u) + \frac{\bar{\sigma}^2}{2}\Delta u = 0$$ have closed-form solution $g(x) = \frac{\exp(T + \sum_i x_i)}{1 + \exp(T + \sum_i x_i)}$

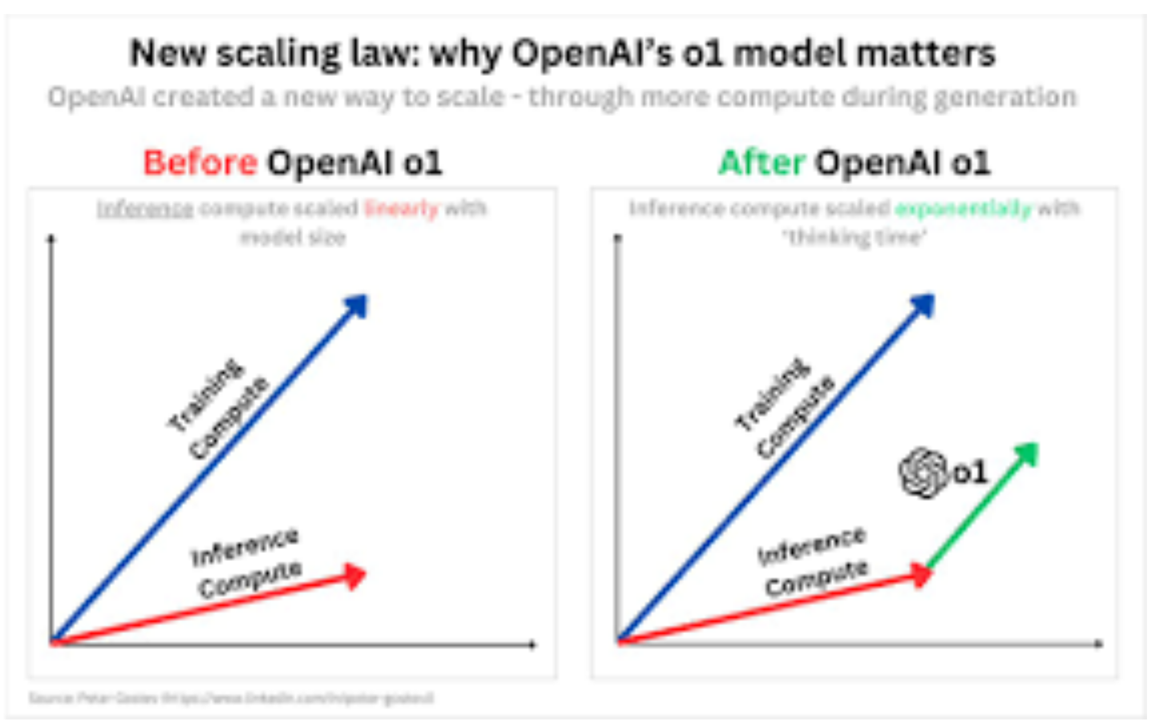




Method	Convergence Rate
PINN	$O(n^{-s/d})$
MLP	$O(n^{-1/2})$
ScaSML	$O(n^{-1/2-s/d})$

Solving a PDE at a single point converges faster than approximating the PDE solution over the entire domain

Inference time scaling



don't fine-tune/retrain/add a new surrogate model

The first Inference-Time Scaling for Scientific Machine Learnin

"Physics-informed"

With trustworthy garuntee

Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x,t) + \Delta U(x,t) + f(U(x,t)) = 0$$

Keeps the structure to enable brownian motion simulation

Can you do simulation for nonlinear equation?

Δ is linear!

Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x,t) + \Delta U(x,t) + f(U(x,t)) = 0$$

Keeps the structure to enable brownian motion simulation

NN

$$\frac{\partial \hat{U}}{\partial t}(x,t) + \Delta \hat{U}(x,t) + f(\hat{U}(x,t)) = g(x,t)$$

$g(x,t)$ is the error made by NN

Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x,t) + \Delta U(x,t) + f(U(x,t)) = 0$$

Keeps the structure to enable brownian motion simulation

NN

$$\underbrace{\frac{\partial \hat{U}}{\partial t}(x,t) + \Delta \hat{U}(x,t) + f(\hat{U}(x,t)) = g(x,t)}$$

$g(x,t)$ is the error made by NN

Subtract two equations

Keeps the linear structure

$$\frac{\partial (U-\hat{U})}{\partial t}(x,t) + \Delta(U-\hat{U})(x,t)) + \underbrace{f(t,\hat{U}(x,t)+U(x,t)-\hat{U}(x,t)) - f(t,\hat{U}(x,t))}_{G\left(t,(U-\hat{U})(x,t)\right)} = g(x,t)\,.$$

Numerical Results

		Time (s)			Relative L^2 Error			L^∞ Error			L^1 Error		
		SR	MLP	SCaSML	SR	MLP	SCaSML	SR	MLP	SCaSML	SR	MLP	SCaSML
LCD	10d	2.64	11.24	23.75	5.24E-02	2.27E-01	**2.73E-02**	2.50E-01	9.06E-01	**1.61E-01**	3.43E-02	1.67E-01	**1.78E-02**
	20d	1.14	7.35	17.59	9.09E-02	2.35E-01	**4.73E-02**	4.52E-01	1.35E+00	**3.28E-01**	9.47E-02	2.37E-01	**4.52E-02**
	30d	1.39	7.52	25.33	2.30E-01	2.38E-01	**1.84E-01**	4.73E+00	1.59E+00	**1.49E+00**	**1.75E-01**	2.84E-01	1.91E-01
	60d	1.13	7.76	35.58	3.07E-01	2.39E-01	**1.32E-01**	3.23E+00	2.05E+00	**1.55E+00**	5.24E-01	4.07E-01	**2.06E-01**
VB-PINN	20d	1.15	7.05	13.82	1.17E-02	8.36E-02	**3.97E-03**	3.16E-02	2.96E-01	**2.16E-02**	5.37E-03	3.39E-02	**1.29E-03**
	40d	1.18	7.49	16.48	3.99E-02	1.04E-01	**2.85E-02**	8.16E-02	3.57E-01	**7.16E-02**	1.97E-02	4.36E-02	**1.21E-02**
	60d	1.19	7.57	19.83	3.97E-02	1.17E-01	**2.90E-02**	8.10E-02	3.93E-01	**7.10E-02**	1.95E-02	4.82E-02	**1.24E-02**
	80d	1.32	7.48	21.99	6.78E-02	1.19E-01	**5.68E-02**	1.89E-01	3.35E-01	**1.79E-01**	3.24E-02	4.73E-02	**2.49E-02**
VB-GP	20d	1.97	10.66	65.46	1.47E-01	8.32E-02	**5.52E-02**	3.54E-01	**2.22E-01**	2.54E-01	7.01E-02	3.50E-02	**1.91E-02**
	40d	1.68	10.14	49.38	1.81E-01	1.05E-01	**7.95E-02**	4.01E-01	3.47E-01	**3.01E-01**	9.19E-02	4.25E-02	**3.43E-02**
	60d	1.01	7.25	35.14	2.40E-01	2.57E-01	**1.28E-01**	3.84E-01	9.50E-01	**7.10E-02**	1.27E-01	9.99E-02	**6.11E-02**
	80d	1.00	7.00	38.26	2.66E-01	3.02E-01	**1.52E-01**	3.62E-01	1.91E+00	**2.62E-01**	1.45E-01	1.09E-01	**7.59E-02**
LQG	100d	1.54	8.67	26.95	7.96E-02	5.63E+00	**5.51E-02**	7.78E-01	1.26E+01	**6.78E-01**	1.40E-01	1.21E+01	**8.68E-02**
	120d	1.25	8.17	27.46	9.37E-02	5.50E+00	**6.64E-02**	9.02E-01	1.27E+01	**8.02E-01**	1.73E-01	1.22E+01	**1.05E-01**
	140d	1.80	8.27	29.72	9.79E-02	5.37E+00	**6.78E-02**	1.00E+00	1.27E+01	**9.00E-01**	1.91E-01	1.23E+01	**1.11E-01**
	160d	1.74	9.07	32.08	1.11E-01	5.27E+00	**9.92E-02**	1.38E+00	1.28E+01	**1.28E+00**	2.15E-01	1.23E+01	**1.79E-01**
DR	100d	1.62	7.75	60.86	9.52E-03	8.99E-02	**8.87E-03**	7.51E-02	6.37E-01	**6.51E-02**	1.13E-02	9.74E-02	**1.11E-02**
	120d	1.26	7.28	65.66	1.11E-02	9.13E-02	**9.90E-03**	7.10E-02	5.74E-01	**6.10E-02**	1.40E-02	9.97E-02	**1.23E-02**
	140d	2.38	7.82	76.90	3.17E-02	8.97E-02	**2.94E-02**	1.79E-01	8.56E-01	**1.69E-01**	3.96E-02	9.77E-02	**3.67E-02**
	160d	1.75	7.42	82.40	3.46E-02	9.00E-02	**3.23E-02**	2.08E-01	8.02E-01	**1.98E-01**	4.32E-02	9.75E-02	**4.02E-02**

SCaSML
Simulation-Calibrated
Scientific Machine Learning

Arxiv

Physics-Informed Inference Time Scaling via Simulation-Calibrated Scientific Machine Learning

Zexi Fan[1], Yan Sun [2], Shihao Yang[3], Yiping Lu*[4]

[1] **Peking University** [2] **Visa Inc.** [3] **Georgia Institute of Technology** [4] **Northwestern University**

fanzexi_francis@stu.pku.edu.cn,yansun414@gmail.com,
shihao.yang@isye.gatech.edu,yiping.lu@northwestern.edu

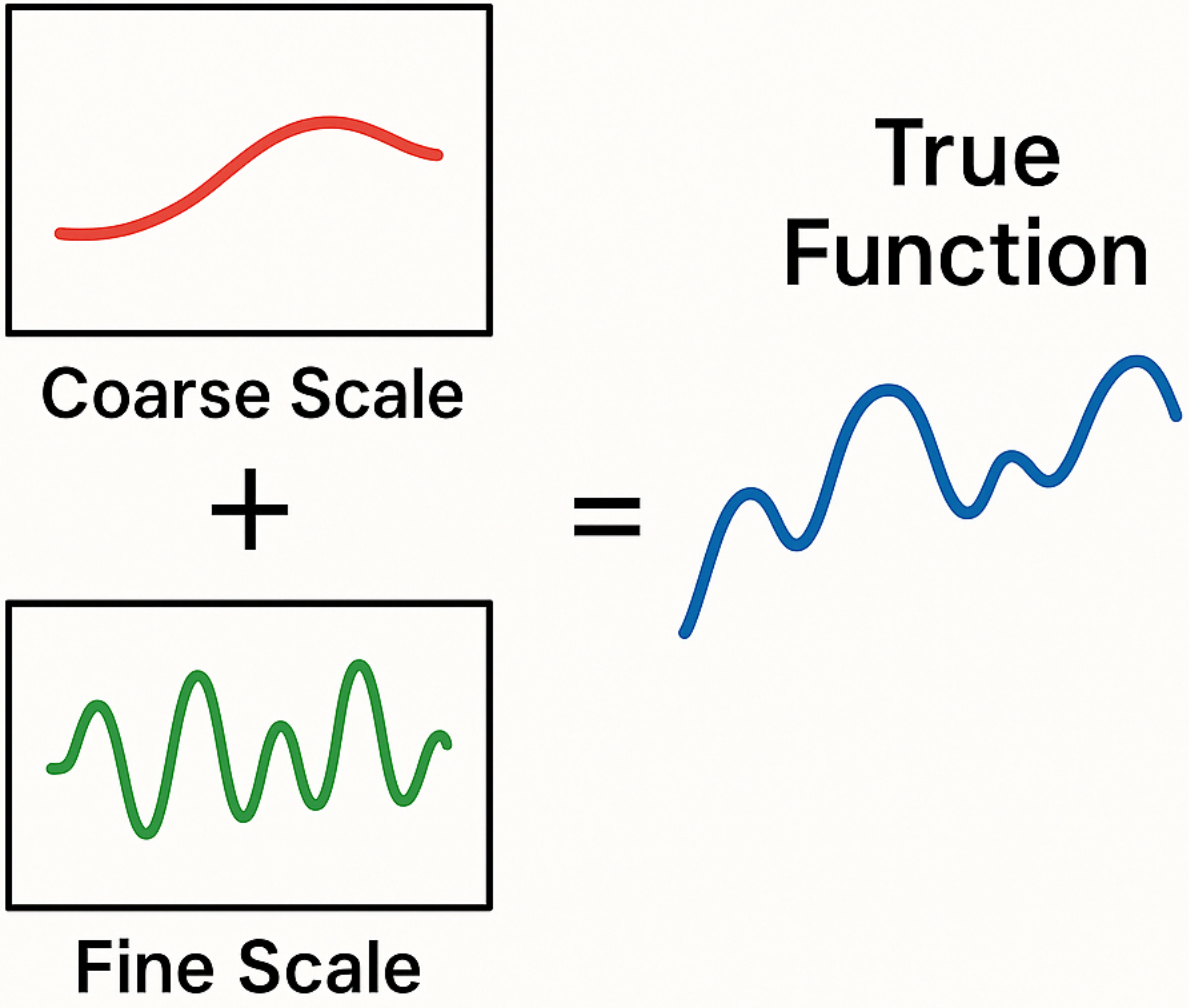
https://2prime.github.io/files/scasml_techreport.pdf

A multiscale view

Capture via surrogate model

Capture via Monte-Carlo

Don't need/use the smoothness structure



More Examples…

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example 1 $\theta = f, \quad X_i = (x_i, f(x_i))$ $\quad \Phi(\theta) = \int f^q(x)dx$

Example 2 $\theta = \Delta^{-1}f, \quad X_i = (x_i, f(x_i))$ $\quad \Phi(\theta) = \theta(x)$

Example 3 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\quad \Phi(\theta) = \mathrm{tr}(A)$

Estimation $\hat{A}$ via Randomized SVD

Estimate tr$(A - \hat{A})$ via Hutchinson's estimator

Lin 17 Numerische Mathematik and Mewyer-Musco-Musco-Woodruff 20

Application in graph theory, quantum …

Eigenvalue Problem

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example 4 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\Phi(\theta) = \text{eigen}(A)$

Eigenvalue Problem

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \rightarrow \hat{\theta} \rightarrow \Phi(\hat{\theta})$$

Scientific Machine Learning — Downstream application

Example 4 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\quad \Phi(\theta) = \text{eigen}(A)$

Randomized SVD

Sketching a Matrix Approximation

$\hat{\theta} = \hat{A}$ → $\Phi(\hat{\theta}) = \text{eign}(\hat{A})$

Eigenvalue Problem

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta} \to \Phi(\hat{\theta})$$

Scientific Machine Learning — Downstream application

Example 4 $\theta = A, \quad X_i = (x_i, Ax_i) \qquad \Phi(\theta) = \text{eigen}(A)$

Randomized SVD

Sketching a Matrix Approximation

$\hat{\theta} = \hat{A} \longrightarrow \Phi(\hat{\theta}) = \text{eign}(\hat{A})$

What is $\Phi(\theta) - \Phi(\hat{\theta})$?

Taylor Expansion

A new Preconditioned Power method + Enable Online Updates

Relationship with Inverse Power Methods

(Approximate) Inverse Power Method	Our Method
$X_{n+1} = (\lambda I - A)^{\dagger} X_n$	$X_{n+1} = (\lambda I - \hat{A})^{\dagger} (A - \hat{A}) X_n$
Replace with an approximate solver $\hat{A}$ changes the fixed point	Ture eigenvector is the fix point for every approximate solver $\hat{A}$

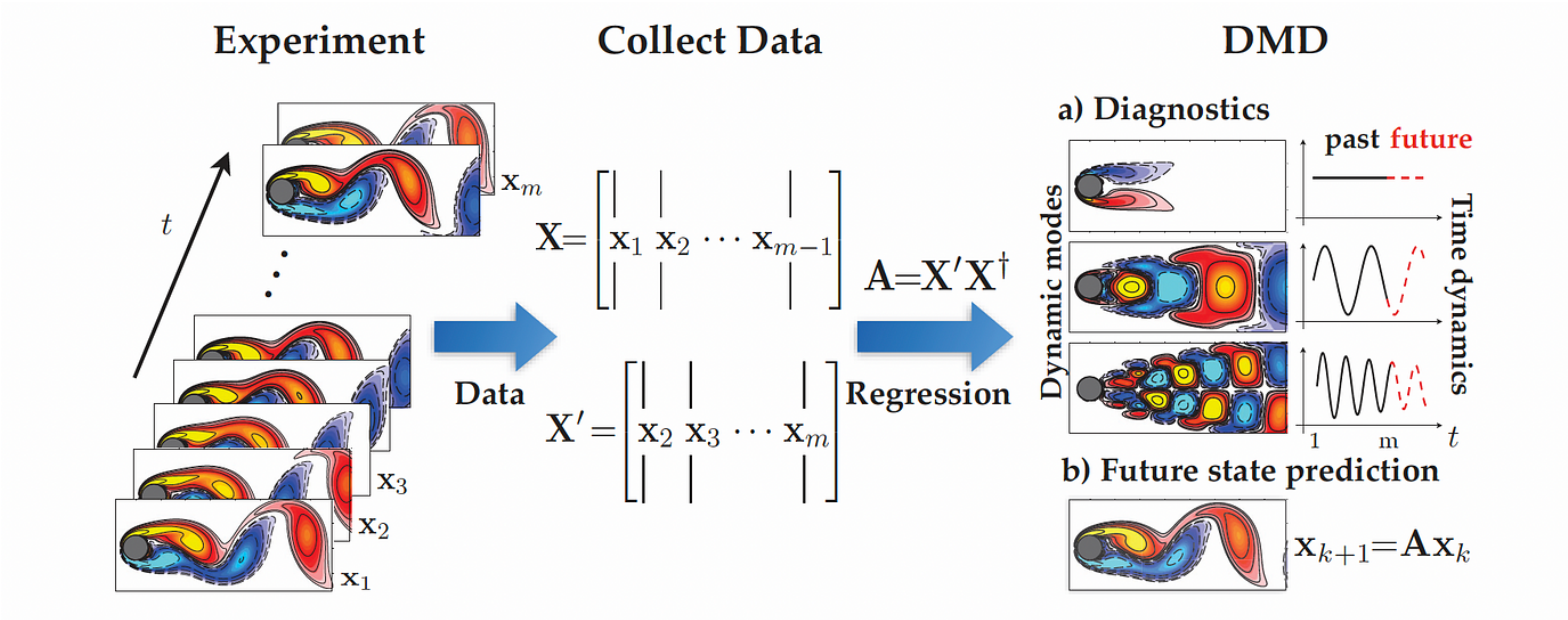
Easy to compute when $\hat{A}$ is low rank

Another Supersing Fact…

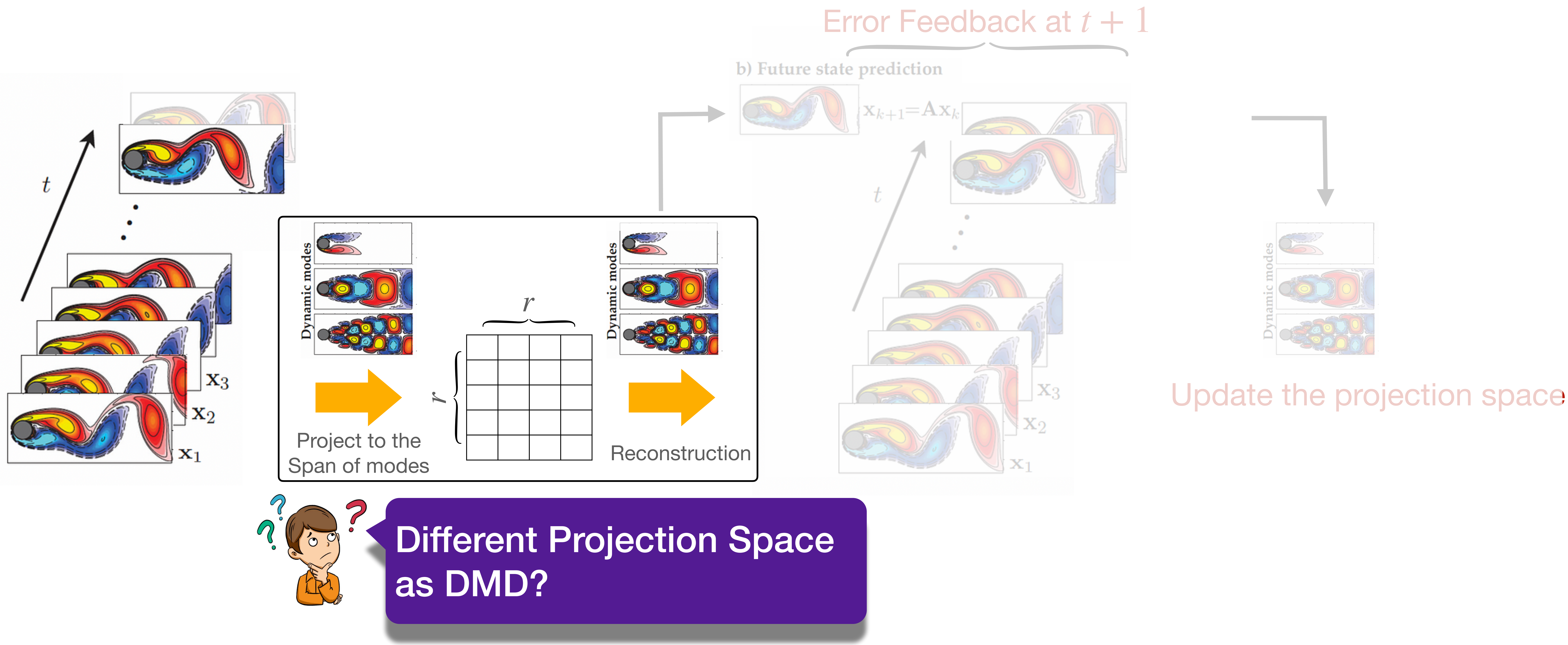
Iteration lies in the Krylov Subspace

- enable dynamic mode decomposition
- Online fast update
- Much better than DMD

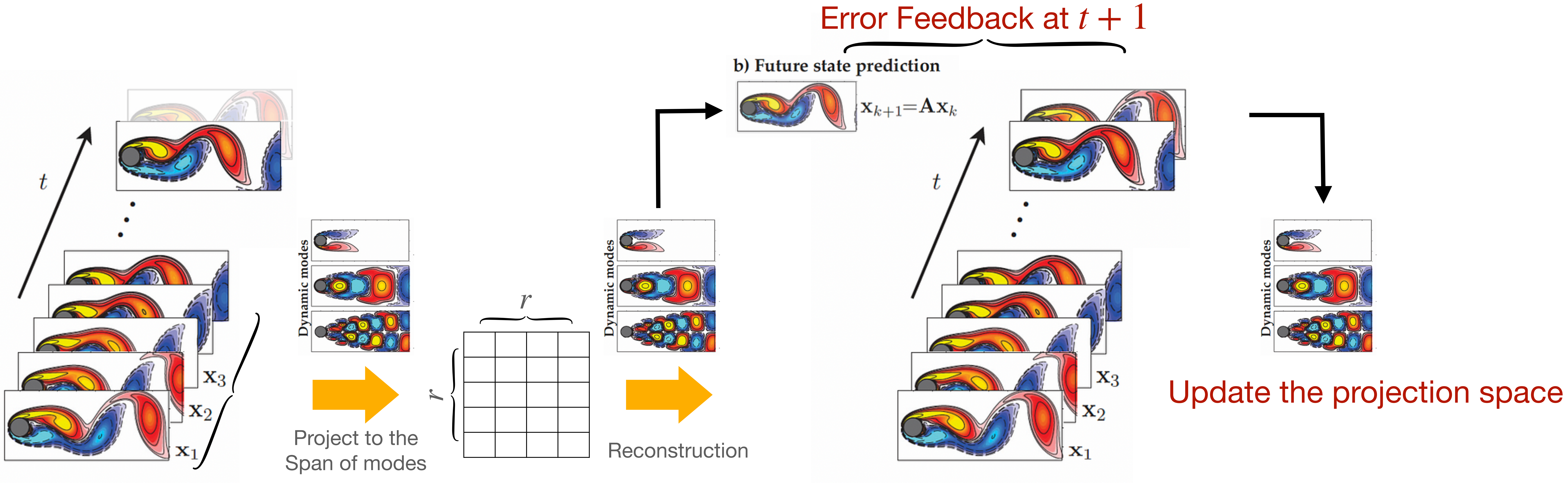
Enable online update!



DMD with First-Order Feedback



DMD with First-Order Feedback



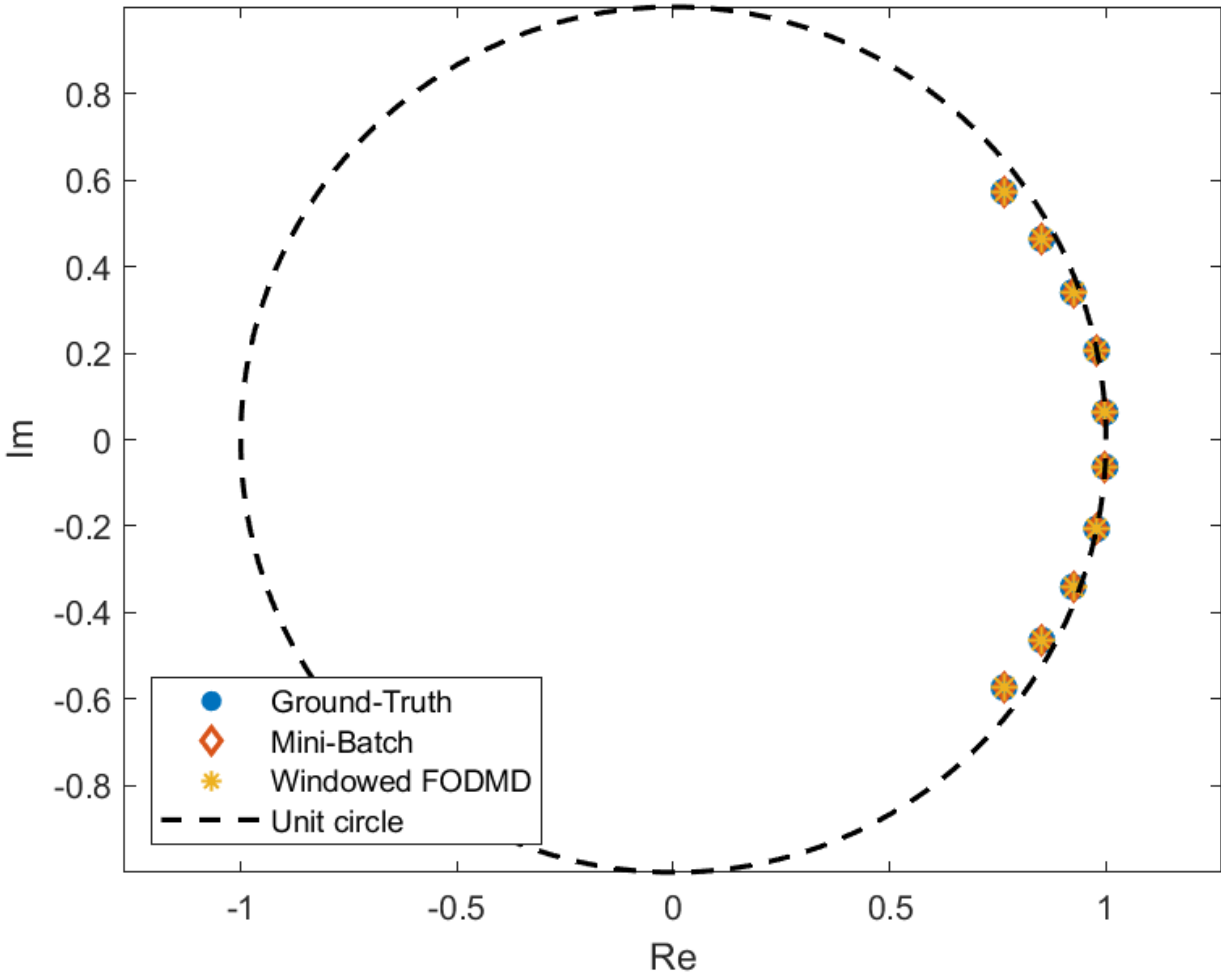
DMD with First-Order Feedback

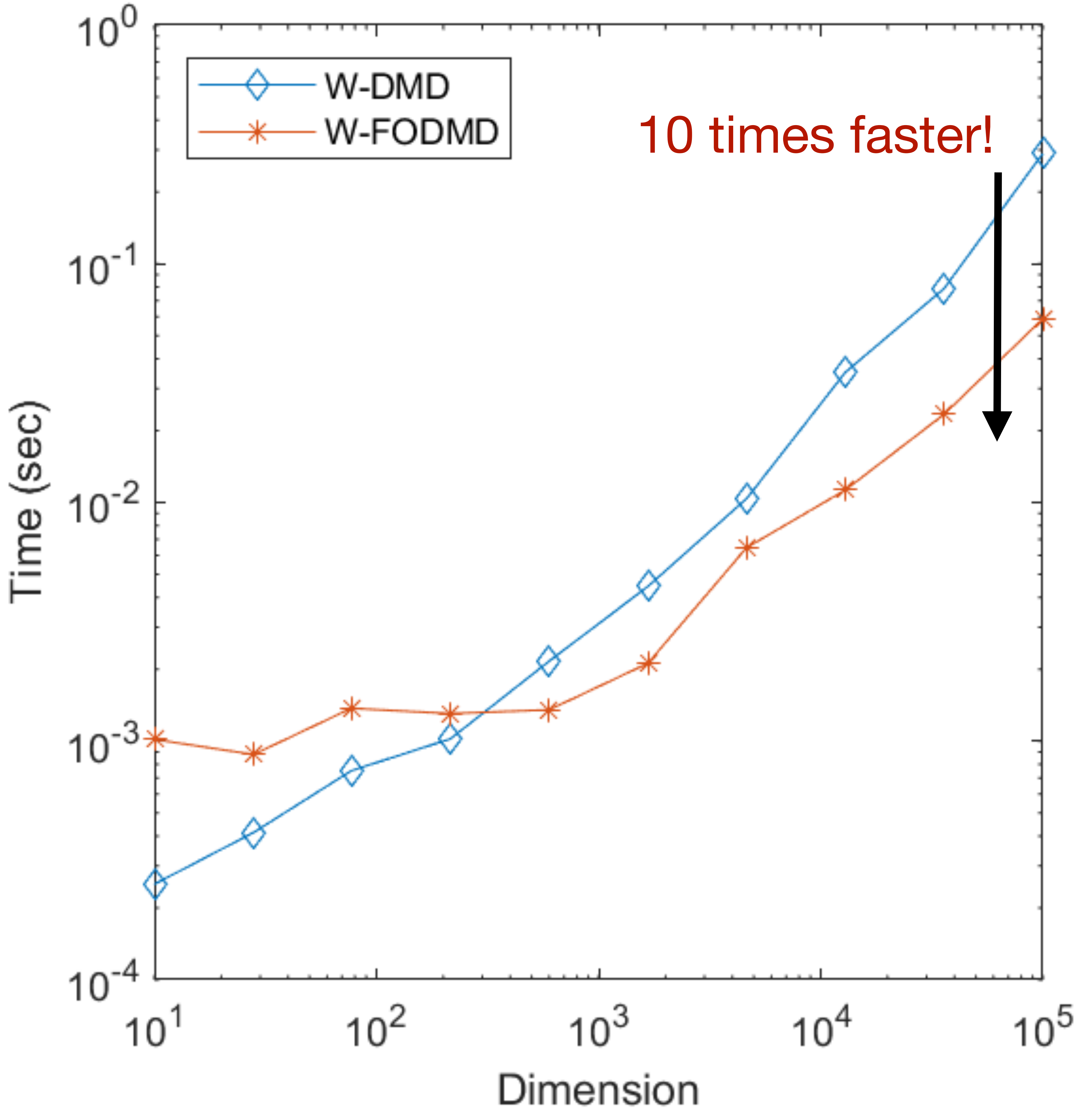
No matrix inverse, No SVD computation

Only a $n \times r$ QR decomposition

(Everything has a closed-form solution)

Faster than Recomputation!

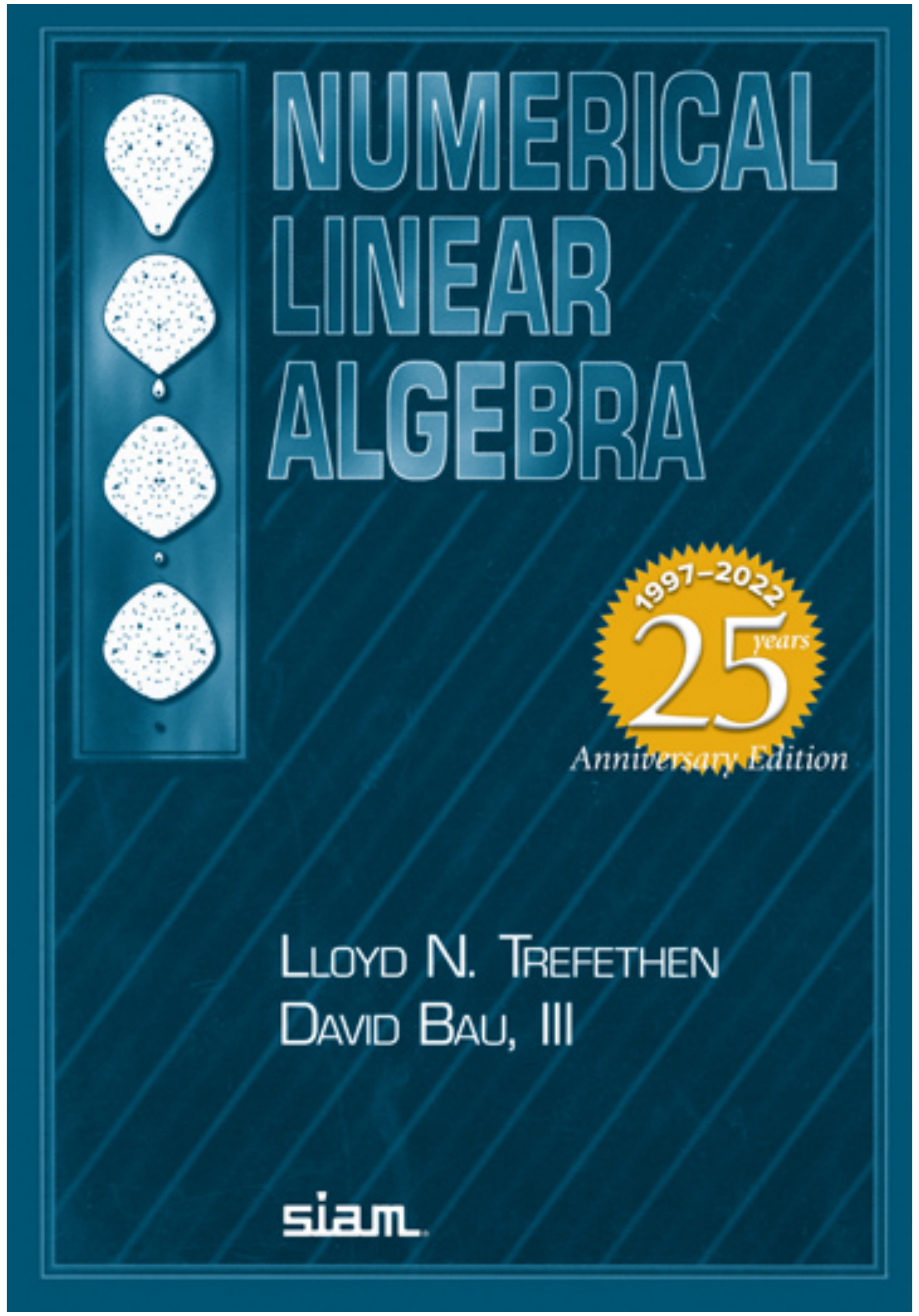




Appendix: Suprising Pre-condition Effect

with a surprising connection with debiasing

Tale 2: Preconditioning



"In ending this book with the subject of preconditioners, we find ourselves at the philosophical center of the scientific computing of the future."

— **L. N. Trefethen and D. Bau III**, `Numerical Linear Algebra` [TB22]

Nothing will be more central to computational science in the next century than the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly.

What is precondition

- Solving $Ax = b$ is equivalent to solving $B^{-1}Ax = B^{-1}b$

hardness depend on $\kappa(A)$

hardness depend on $\kappa(B^{-1}A)$

Become easier when $B \approx A$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
  - Using an approximate solver $Bx_1 = b$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
  - Using an approximate solver $Bx_1 = b$
  - $x - x_1$ satisfies the equation $A(x - x_1) = b - Ax_1$
  - Using the approximate solver to approximate $x - x_1$ via $Bx_2 = b - Ax_1$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
  - Using an approximate solver $Bx_1 = b$

Iterative Refinement Algorithm

- $x - \sum_{i=1}^{t} x_i$ satisfies the equation $A(x - \sum_{i=1}^{t} x_i) = b - A\sum_{i=1}^{t} x_i$
- Using the approximate solver to approximate $x - \sum_{i=1}^{t} x_i$ via $Bx_{i+1} = b - A\sum_{i=1}^{t} x_i$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
  - Using an approximate solver $Bx_1 = b$

Iterative Refinement Algorithm

- $x - \sum_{i=1}^{t} x_i$ satisfies the equation $A(x - \sum_{i=1}^{t} x_i) = b - A \sum_{i=1}^{t} x_i$
- Using the approximate solver to approximate $x - \sum_{i=1}^{t} x_i$ via $Bx_{i+1} = b - A \sum_{i=1}^{t} x_i$

$$x_{i+1} = (I - B^{-1}A)x_i + B^{-1}b$$

Preconditioned Jacobi Iteration

This Talk: A New Way to Implement Precondition

Via Debiasing

- **Step 1:** Aim to solve (potentially nonlinear) equation $A(u) = b$

use Machine Learning

Unrealiable approximate solver as preconditioner

- **Step 2:** Build an approximate solver $A(\hat{u}) \approx b$
  - Via machine learning/sketching/finite element….
- **Step 3:** Solve $u - \hat{u}$

Connection with control variate, doubly robust estimator, Multifidelity Monte Carlo

AIM: Debiasing a Learned Solution = Using Learned Solution as preconditioner!

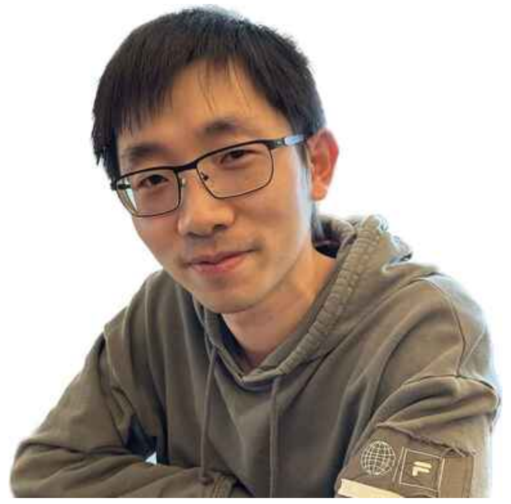
Thank You
And Questions?

Northwestern | McCORMICK SCHOOL OF ENGINEERING

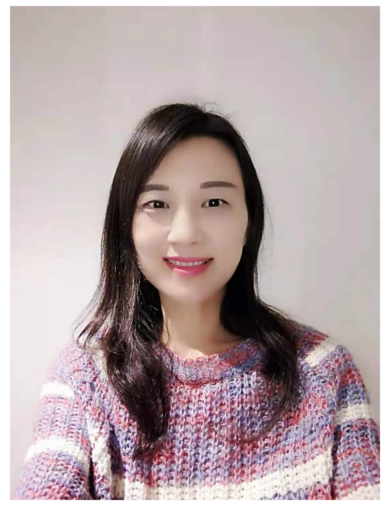
Lexing Ying (Stanford)

Jose Blanchet (Stanford)

Shihao Yang (Gatech)



Sifan Wang (Yale)



Chunmei Wang (UF)

Jiajin Li (UBC)

Students: Haoxuan Chen, Yinuo Ren(Stanford), Youheng Zhu, Kailai Chen (Northwestern), Jasen Lai (UF), Zhaoyan Chen, Weizhong Wang (FDU), Kaizhao Liu (PKU->MIT), Zexi Fan (PKU), Ruihan Xu (Uchicago) …

Scaling in Training:

Jasen Lai, Sifan Wang, Chunmei Wang, **Yiping Lu.** Unveiling the scaling law of PINN under Non-Euclidean Geometry

Scaling in Inference

Zexi Fan, Yan Sun, Shihao Yang, and **Yiping Lu.** Physics-Informed Inference Time Scaling via Simulation-Calibrated Scientific Machine Learning

Eigenvector Computation:

Ruihan Xu, **Yiping Lu.** What is a Sketch-and-Precondition Derivation for Low-Rank Approximation? Inverese Power Error or Inverse Power Estimation?