北京大学
PEKING UNIVERSITY

LLM Pretraining through Flat-Direction Dynamics Enhancement

Kun Yuan (袁 坤)

Center for Machine Learning Research @ Peking University

Joint Work with

Shuchen Zhu
(PKU)

Rizhen Hu
(PKU)

Mingze Wang
(PKU)

Mou Sun
(Zhejiang Lab)

Xue Wang
(Meituan)

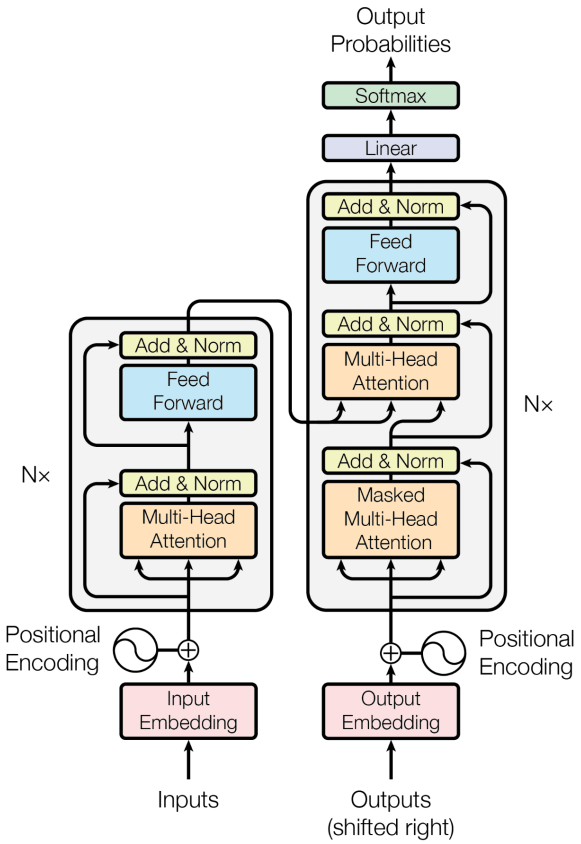
Zaiwen Wen
(PKU)

北京大学
PEKING UNIVERSITY

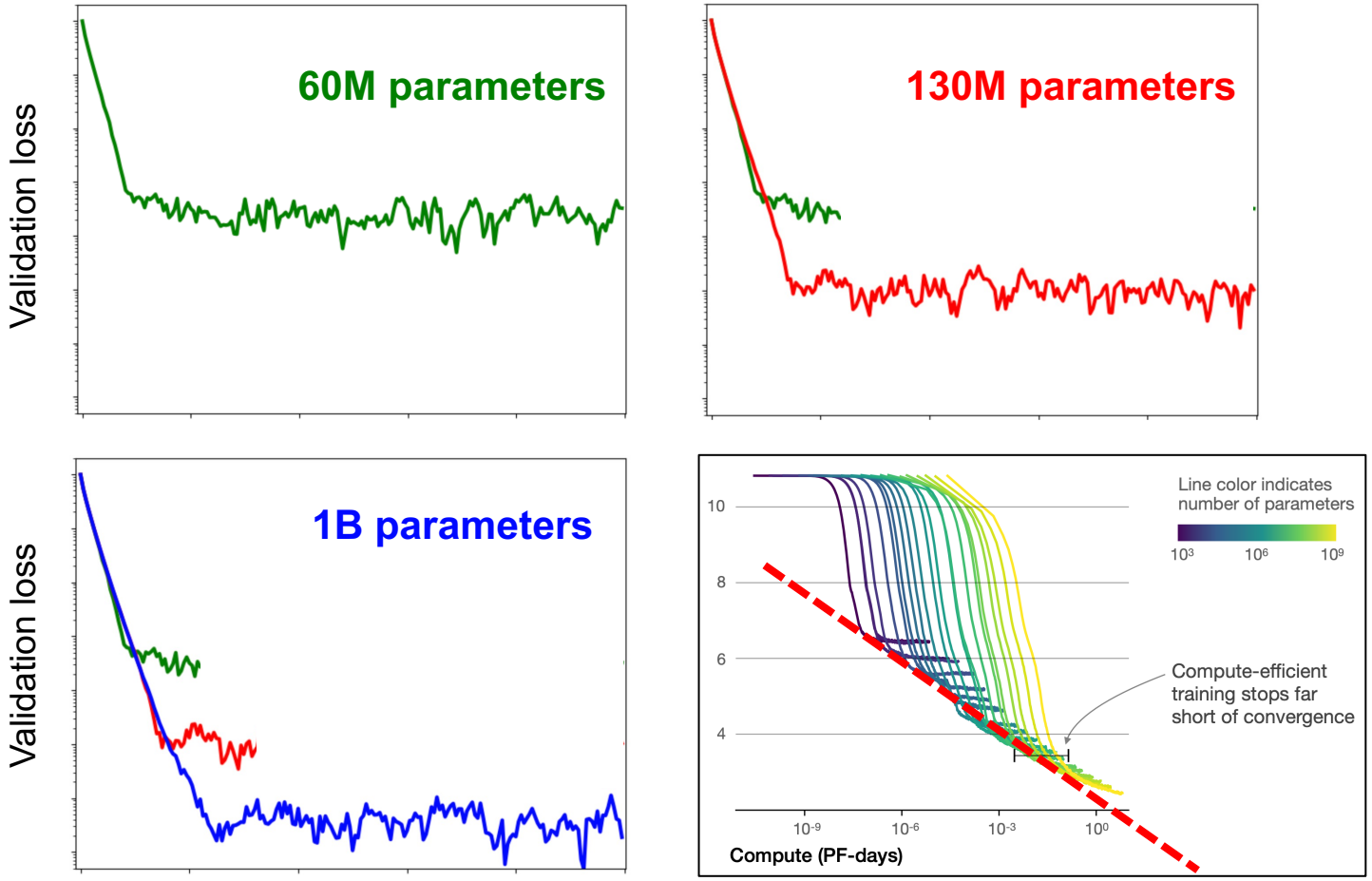
PART 01

Motivation

Scaling Laws for Large Language Models



LLM Performance vs. Model Size

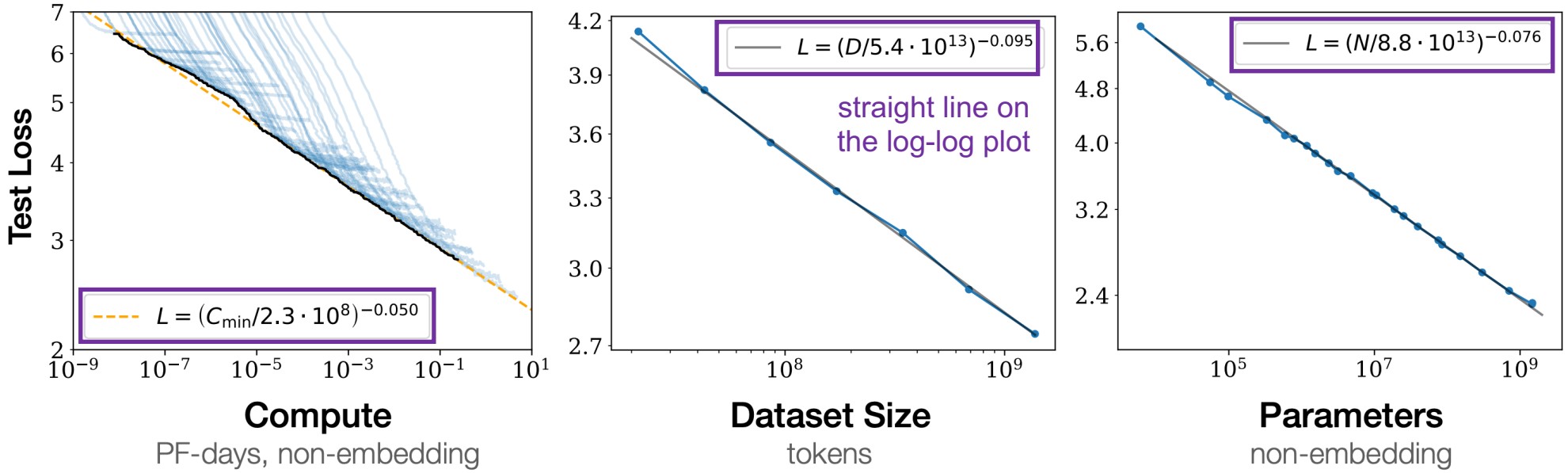


LLM performance improves as parameters get large

Evaluated across models of different scales

Scaling Laws for Large Language Models

[OpenAI, Scaling Laws for Neural Language Models, 2020]

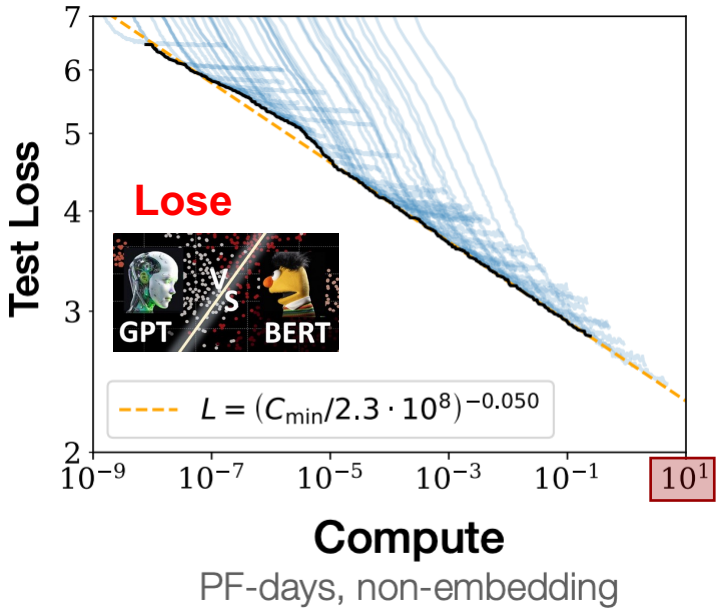


- LLM performance is largely independent of specific architecture
- Performance improves primarily with increases in **data**, **parameters**, and **compute**
- Loss scales as a **power law** with data, parameters, and compute

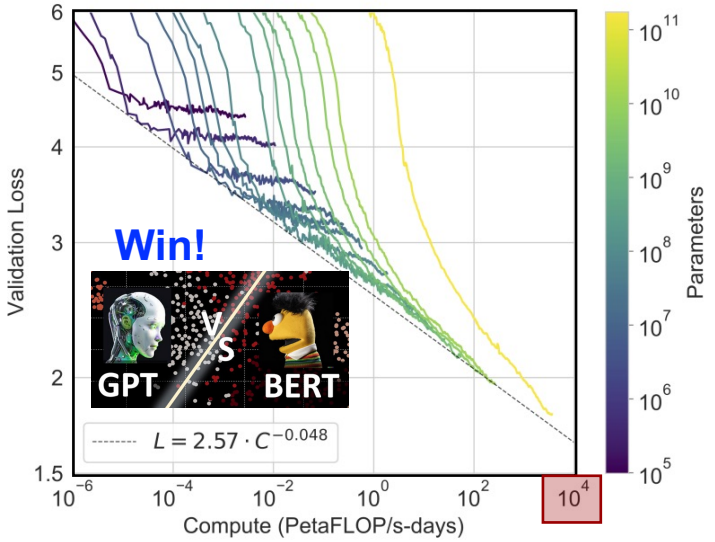
Scaling Laws as a Foundation of ChatGPT's Success

北京大学 PEKING UNIVERSITY

GPT-2, **1.5B** parameters, 2019



GPT-3, **175B** parameters, 2020



- GPT-2 (2019) scaled far beyond BERT in parameters, yet underperformed in practice
- OpenAI exploited scaling laws to build 175B-param GPT-3, launching the large-model era
- Consensus: **Large model** + **Large data** + **Large compute** = **High intelligence**

Scaling Laws Drive Rapid Growth in Computations for LLMs

北京大学 PEKING UNIVERSITY

Training and inference costs rise sharply

Cost grows

Dramatic computation cost; Massive clusters

Computation grows

Increasing parameters and data

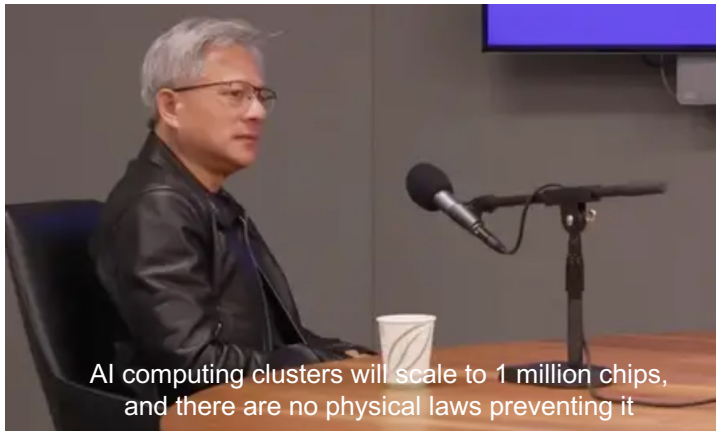
Scaling law

Growing demand for better LLM performance

LLaMA-3 on 20,000 GPUs cluster

To train our largest Llama 3 models, we combined three types of parallelization: data parallelization, model parallelization, and pipeline parallelization. Our most efficient implementation achieves a compute utilization of over 400 TFLOPS per GPU when trained on 16K GPUs simultaneously. We performed training runs on two custom-built 24K GPU clusters. To maximize GPU uptime, we developed an advanced new training stack that automates error detection, handling, and maintenance. We also greatly improved our

XAI built cluster with 100K GPUs



Jensen Huang: AI compute clusters will scale up to 1M chips

Key question: How to save computations?

Scaling Laws Drive Rapid Growth in Computations for LLMs

北京大学 PEKING UNIVERSITY

Training and inference costs rise sharply

Cost scaling

Dramatic computation cost; Massive clusters

Compt. scaling

Increasing parameters and data

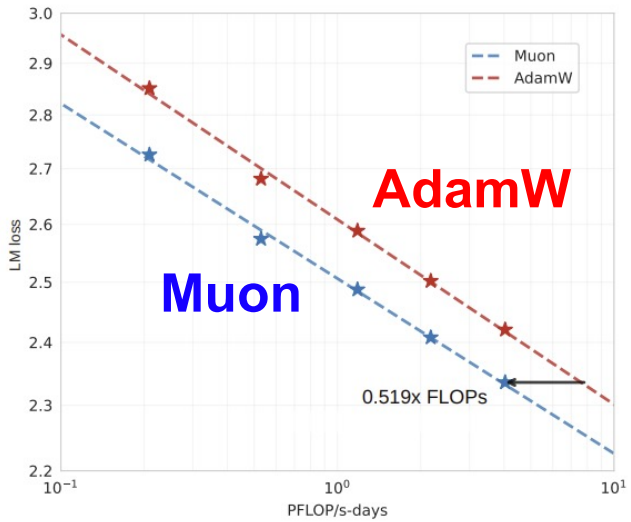
Scaling law

Growing demand for better LLM performance

Approach 1: Design new architectures or optimizers and explore novel scaling laws

Approach 2: Computation-efficient training methods driven by implicit structures inside models

Approach 3: Develop new hardware (e.g., SuperNodes) to reduce training and inference costs



[Muon is Scalable for LLM Training]

Muon cuts the computation by **half**

Smart optimizer → Better scaling

This talk focuses on **optimizers**

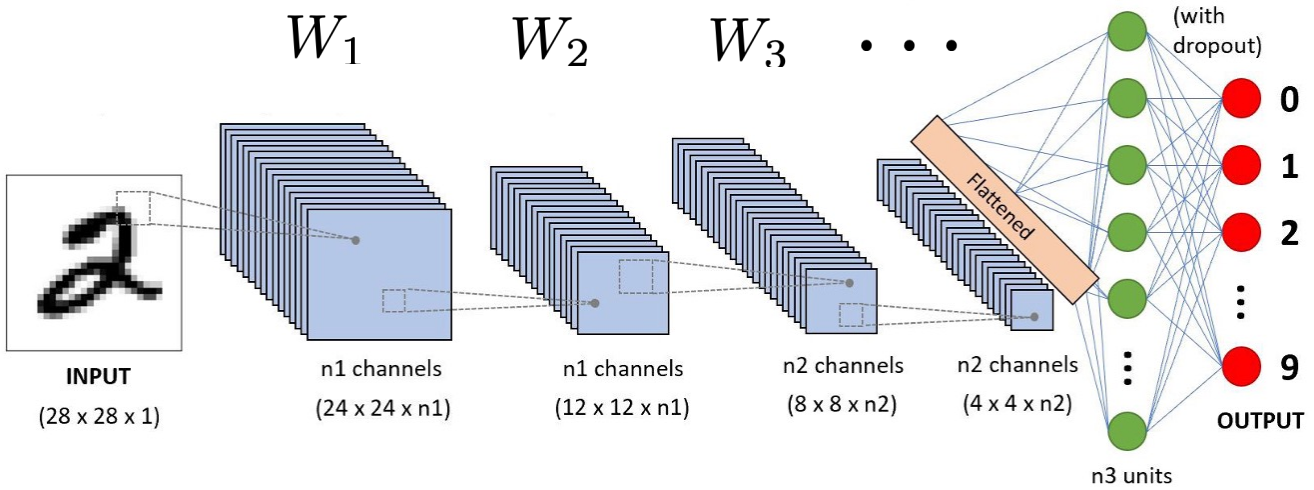
北京大学
PEKING UNIVERSITY

PART 02

Optimizers

LLM Pretraining is Essentially Solving Stochastic Optimization

- The model weights in neural networks are a set of matrices $\boldsymbol{W} = \{W_\ell\}_{\ell=1}^{L}$



- Let $h(\boldsymbol{W};\xi)$ be the language model; $\hat{y} = h(\boldsymbol{W};\xi)$ is the predicted token

$$\boldsymbol{W}^\star = \arg\min_{\boldsymbol{W}} \{\mathbb{E}_{\xi\sim\mathcal{D}}\left[L(h(\boldsymbol{W};\xi), y)\right]\}$$

cross entropy

data distribution

pred. token

real token

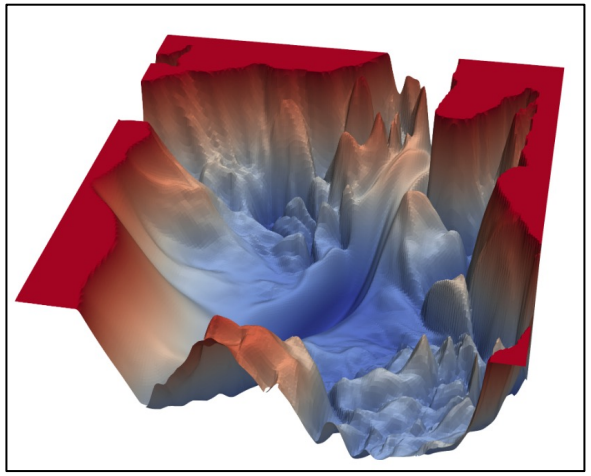
LLM Pretraining is Essentially Solving Stochastic Optimization

- If we define $\boldsymbol{\xi} = (\xi, y)$ and $F(\boldsymbol{W}; \boldsymbol{\xi}) = L(h(\boldsymbol{W}; \xi), y)$, the LLM problem becomes

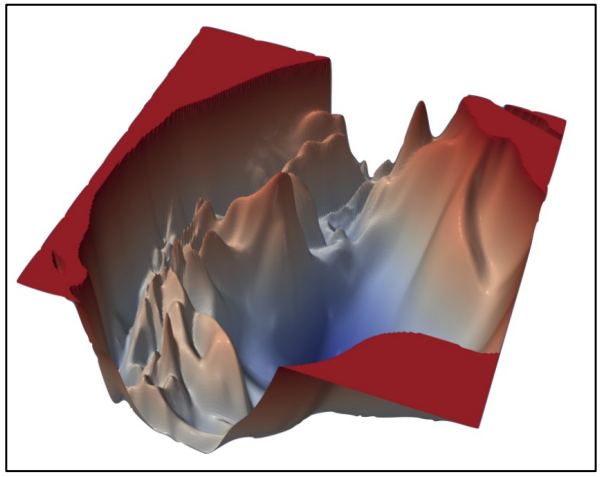
Stochastic optimization: $\boldsymbol{W}^\star = \arg\min_{\boldsymbol{W}} \{\mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{D}} [F(\boldsymbol{W}; \boldsymbol{\xi})]\}$

- Training neural network is extremely challenging due to its non-convexity

VGG-56



VGG-110



[Visualizing the loss landscape of neural nets]

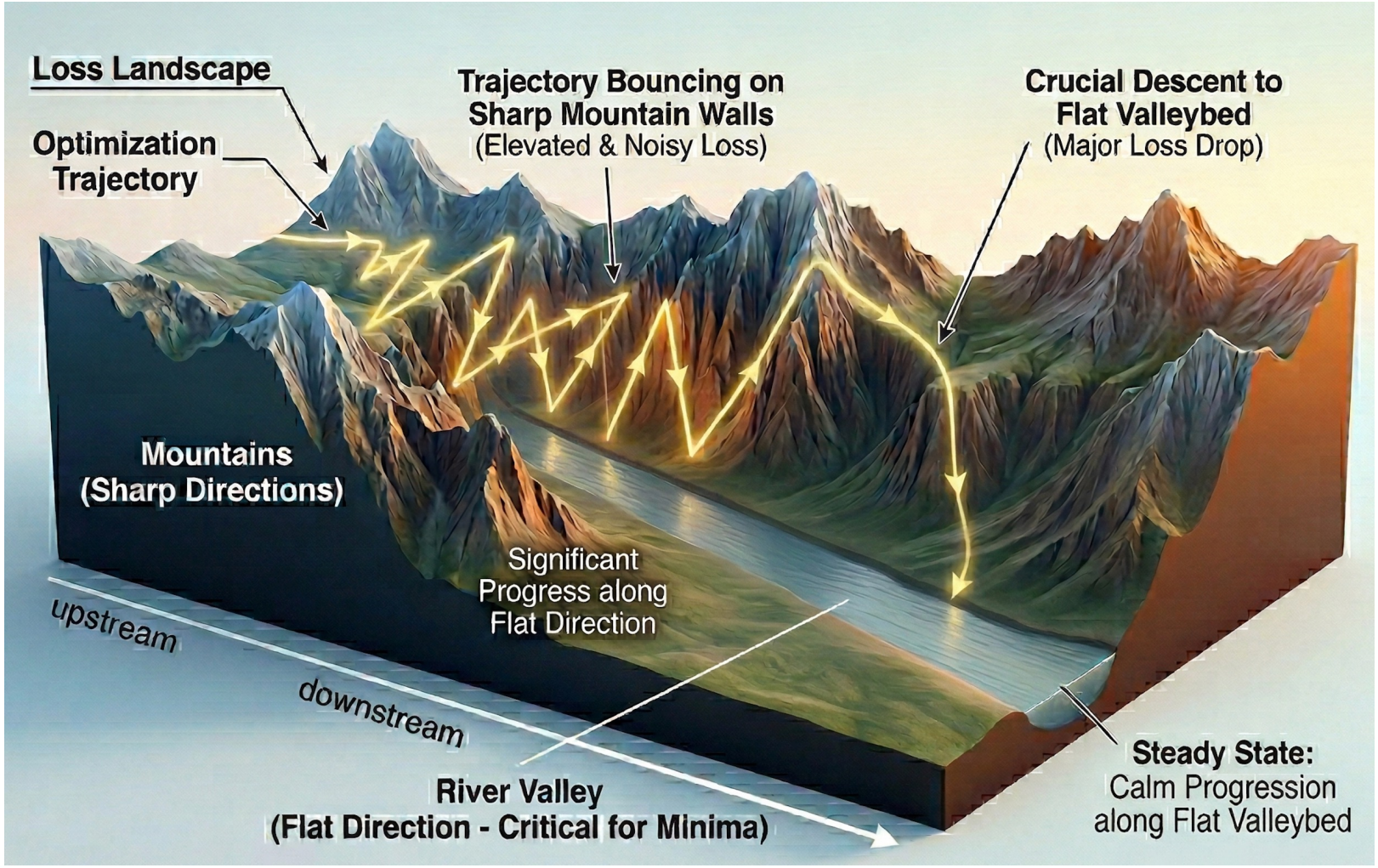
Optimizers for deep learning must adapt to its landscape

LLM Loss Landscape

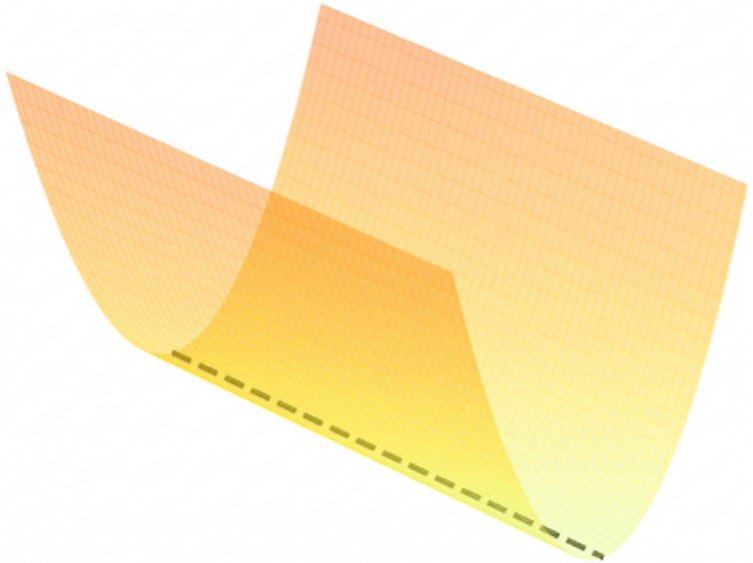
北京大学 PEKING UNIVERSITY

A River Valley Loss Landscape

[Understanding Warmup-Stable-Decay Learning Rates: A River Valley Loss Landscape Perspective, 2024]



Simplified illustration



Flat direction is **critical** for loss drop

Sharp direction results in **oscillation**

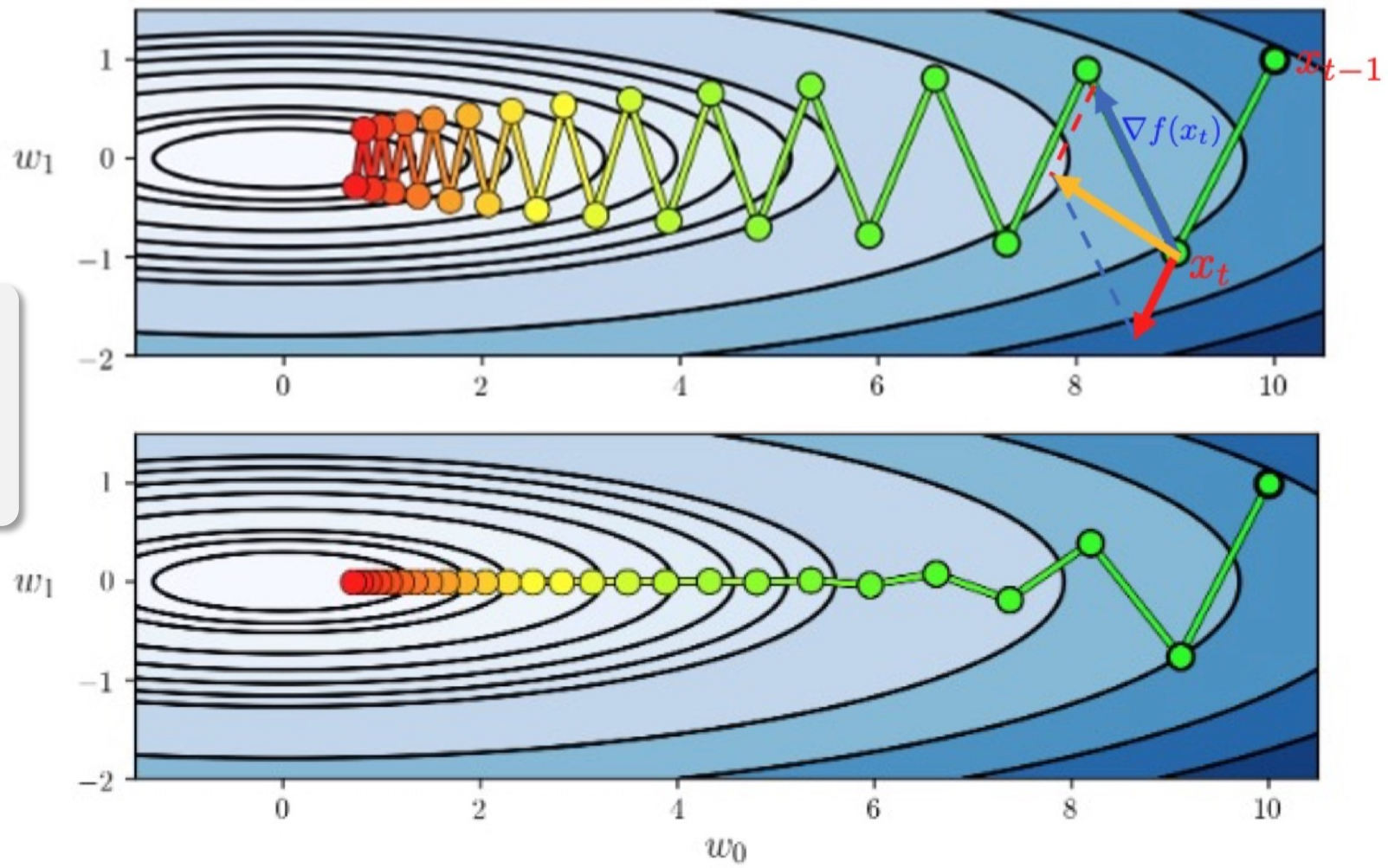
How to Reduce Oscillation? Momentum and Precondition

北京大学 PEKING UNIVERSITY

Momentum
relieves oscillation

$$M_k = (1-\beta_1)M_{k-1} + \beta_1 G_k$$
$$W_{k+1} = W_k - \gamma M_k$$

- Provides inertia
- Stabilizes the trajectory



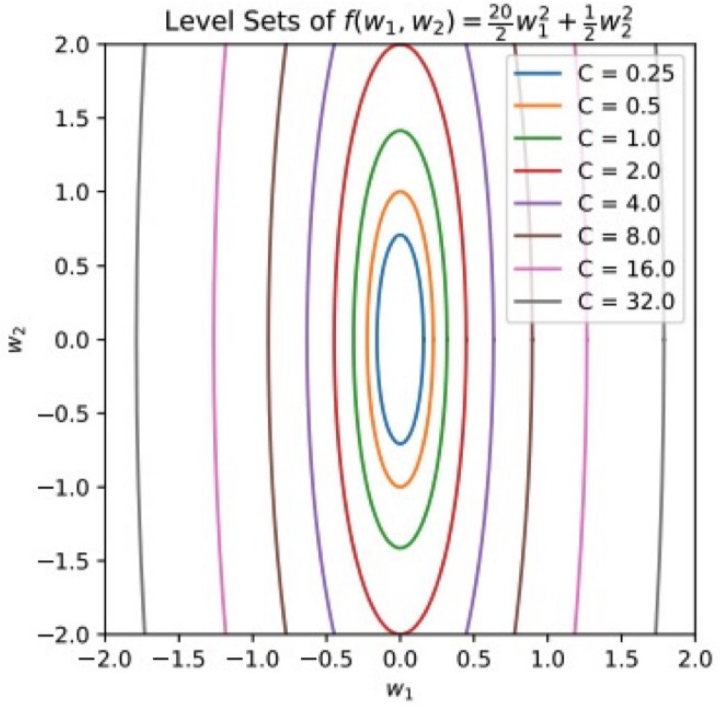
How to Reduce Oscillation? Momentum and Precondition

Precondition
relieves oscillation

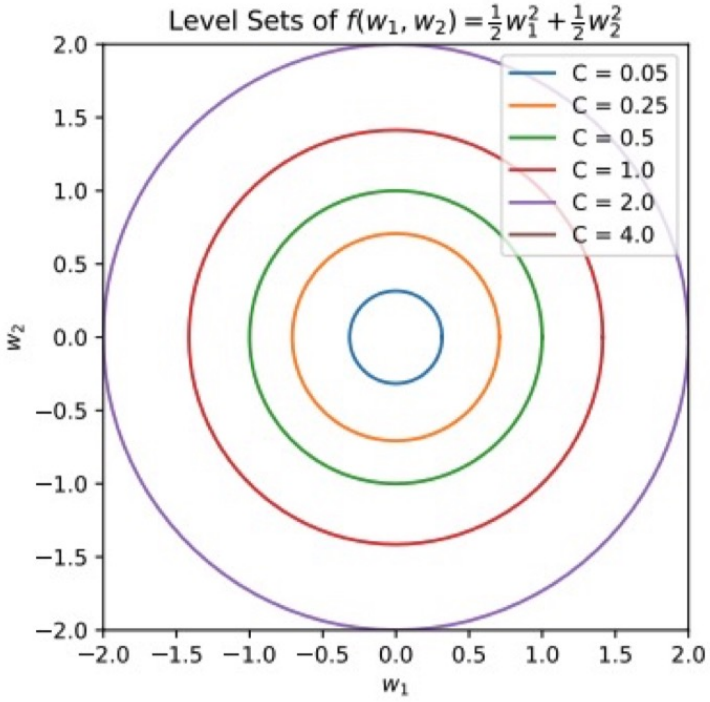
$$W_{k+1} = W_k - \gamma P G_k$$

- Reshape the loss landscape

Without Precondition



With Precondition



Adam = Momentum + Precondition

- In LLM, the preconditioner will **never** be perfect because
  - **Too heavy to calculate** due to billions of parameters
  - **Stochastic gradient noise** obscures the true local curvature

Need momentum to boost performance

- **Adam = Momentum + Precondition**

$$M_k = (1-\beta_1)M_{k-1} + \beta_1 G_k$$
$$V_k = (1-\beta_2)V_{k-1} + \beta_2 G_k \odot G_k$$
$$U_k = M_k \odot V_k^{-1/2} \quad \text{(momentum)} \quad \text{(precondition)}$$
$$W_{k+1} = W_k - \gamma U_k$$

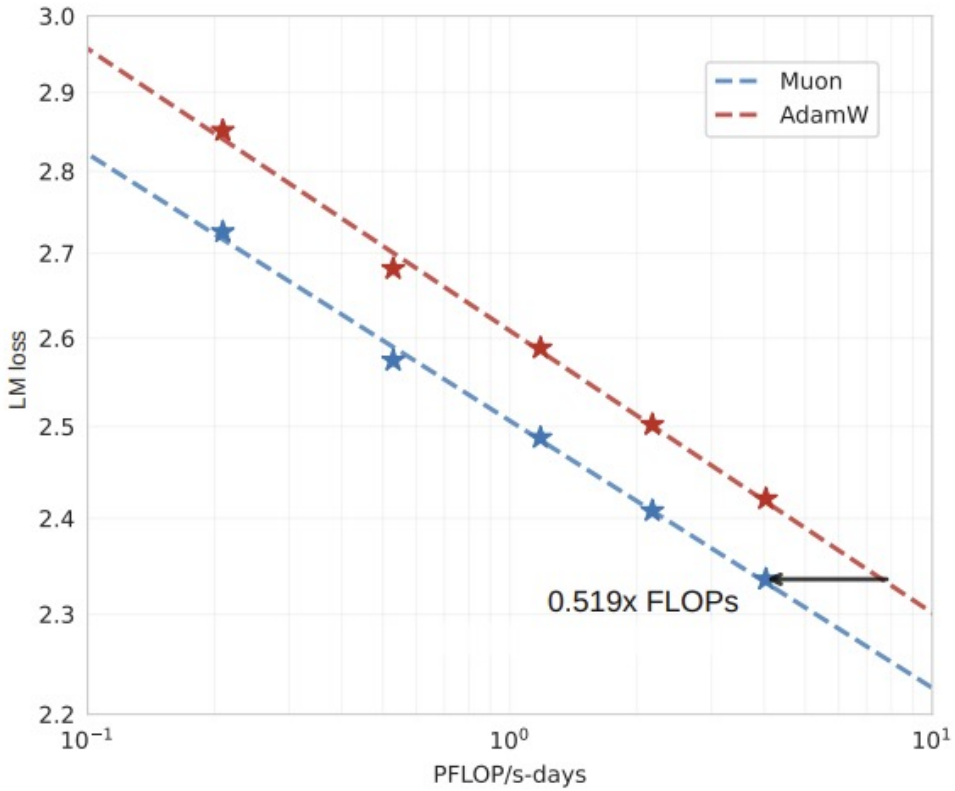
[Adam: A Method for Stochastic Optimization]

Muon and SOAP: The New State-of-the-Art

- The preconditioner of Adam is **diagonal**; too simple to capture the landscape
- Muon uses **matrix-valued** preconditioning via **Newton-Schulz** iterations

$$M_k = (1-\beta)M_{k-1} + \beta G_k$$
$$M_k^\beta = (1-\beta)M_k + \beta G_k$$
$$U_k = M_k^\beta \left((M_k^\beta)^\top M_k^\beta\right)^{-1/2}$$

[Keller Jordan blog]

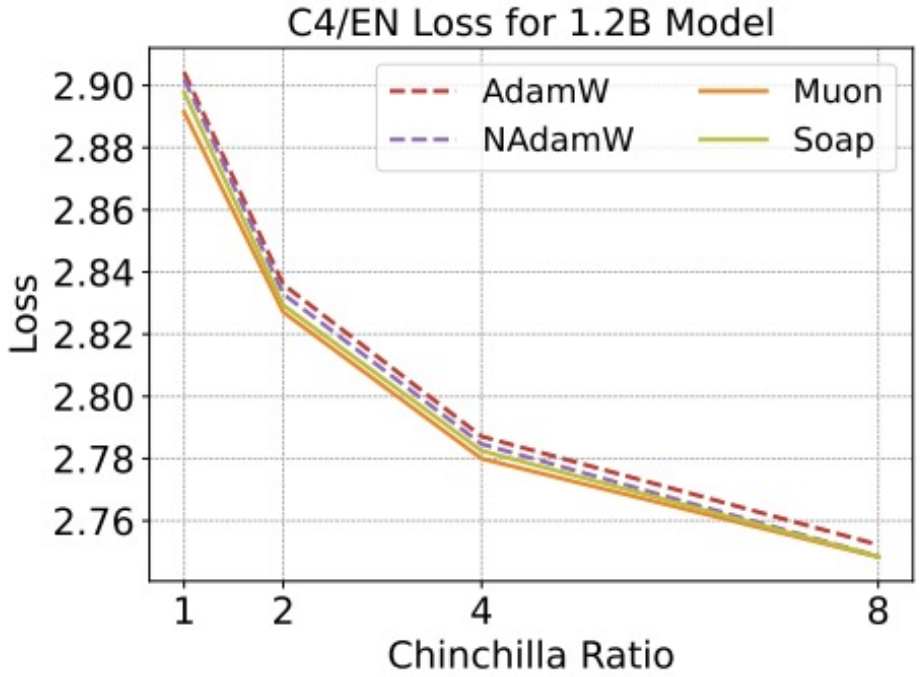


[Muon is Scalable for LLM Training]

Muon and SOAP: The New State-of-the-Art

- SOAP constructs **matrix-valued** preconditioning via the **spectral decomposition** of Kronecker-factored gradient statistics

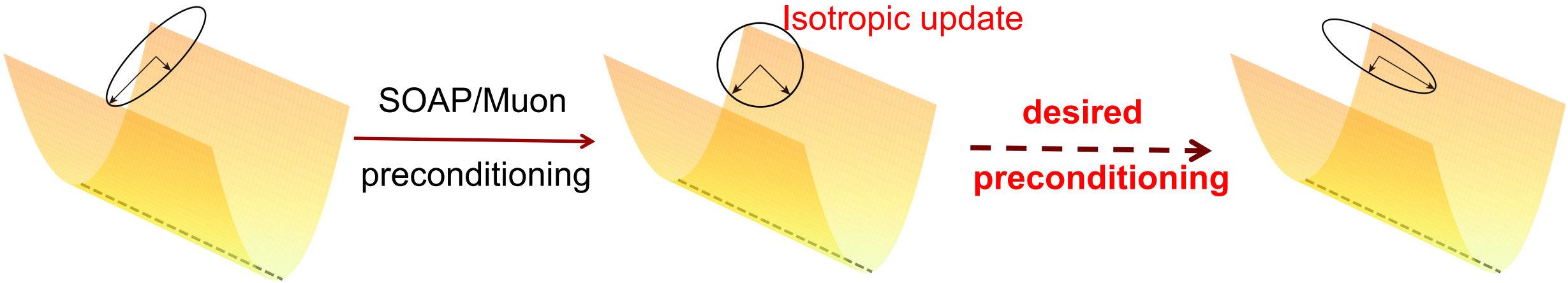
$$
\begin{aligned}
M_k &= (1-\beta_1)M_{k-1} + \beta_1 G_k \\
V_k &= (1-\beta_2)V_{k-1} + \beta_2 (Q_l^\top G_k Q_r)^{\odot 2} \\
L_k &= (1-\beta_s)L_{k-1} + \beta_s G_k G_k^\top \\
R_k &= (1-\beta_s)R_{k-1} + \beta_s G_k^\top G_k \\
U_k &= Q_l \frac{Q_l^\top M_k Q_r}{V_k^{1/2}} Q_r^\top
\end{aligned}
$$



[SOAP: Improving and Stabilizing Shampoo using Adam]

Limitations of Existing Optimizers

1. Preconditioner-induced **isotropic** update [Staib et al., 2019; Zhou et al., 2020; Liu et al., 2025a; Lu et al., 2025; Wang et al., 2025b]



2. **Inadequate** momentum update

$$M_k = (1-\beta)M_{k-1} + \beta G_k$$

desired momentum

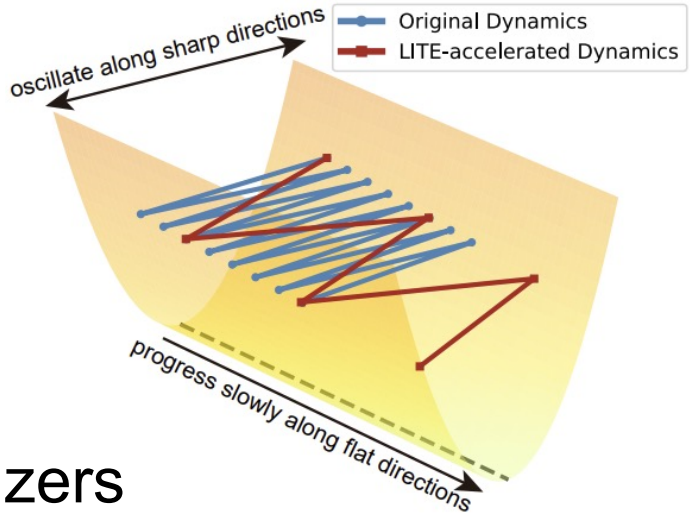
High damping in **sharp** direction

Low damping in **flat** direction

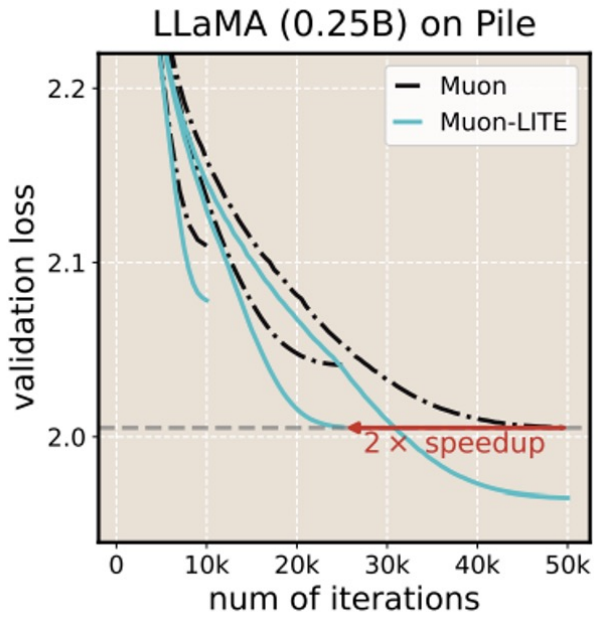
β applies the same friction across all parameters

Our Goal

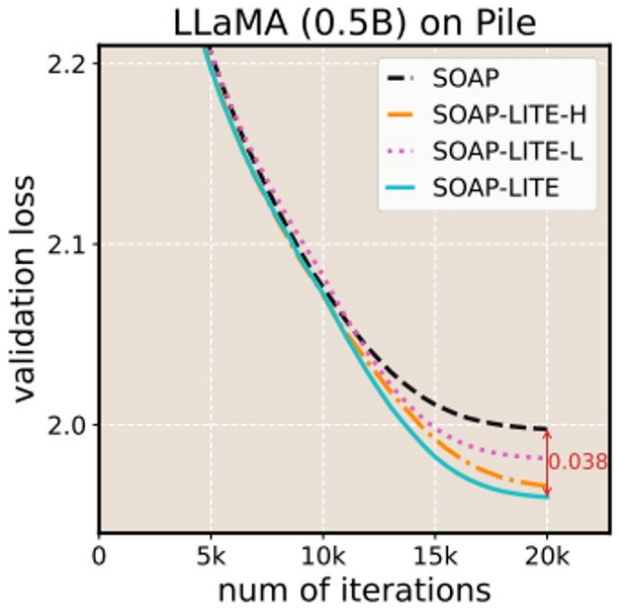
1. Smarter Preconditioner and Momentum to enhance flat-direction dynamics



2. A unified strategy to accelerate various optimizers



Muon + **LITE**
= 2x Speedup



SOAP + **LITE**
= Huge Loss Improvement

北京大学
PEKING UNIVERSITY

PART 03

Riemannian ODE Framework

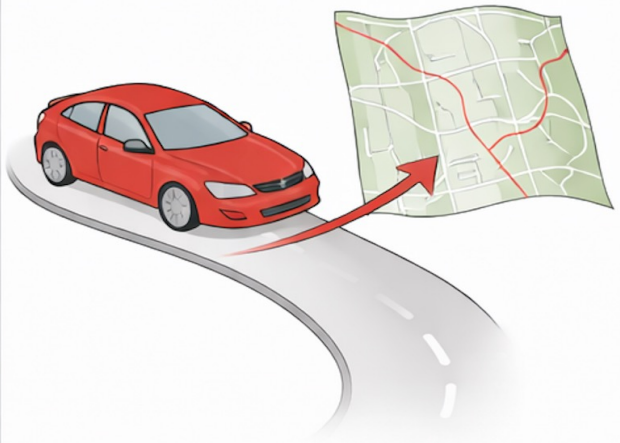
ODE Framework for First Order Momentum Methods

The intuitive role of **gradient**, **preconditioner**, and **momentum**

Gradient = engine + steering wheel

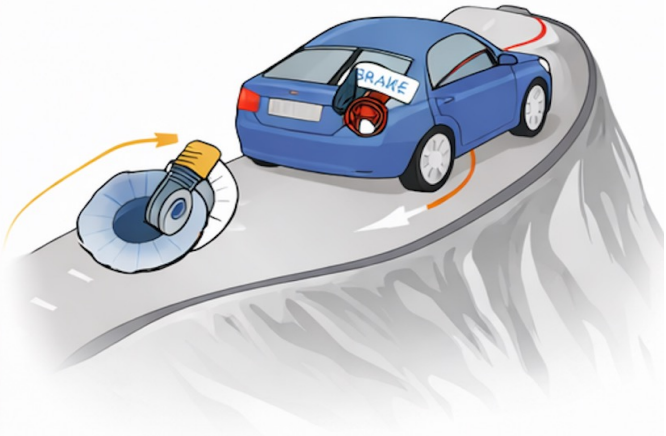
Decides the immediate direction and provides force to descent

Preconditioner = map



Decides the long-run (and more effective) direction

Momentum = brake



Controls the engine; heavy brake in sharp direction; light brake in flat direction

How to characterize them with Math?

ODE Framework for First Order Momentum Methods

- **ODE** can rigorously characterize the role of gradient and momentum

A unified form of momentum methods

$$\begin{cases} m_k = (1-\alpha)m_{k-1} + \nabla f(w_k), \\ w_{k+1} = w_k - \eta(m_k + \beta \nabla f(w_k)). \end{cases}$$

continuization →

← discretization

First order ODE system

$$\begin{cases} \dot{w}_t = -\eta(m_t + \beta \nabla f(w_t)), \\ \dot{m}_t = -\alpha m_t + \nabla f(w_t). \end{cases}$$

- $\beta = 0$. Heavy ball/Polyak momentum
- $\beta > 0$. Nesterov momentum

↓ Second order ODE system

$$\ddot{w}_t + \alpha \dot{w}_t + \beta\eta \nabla^2 f(w_t)\dot{w}_t + \eta(\alpha\beta + 1)\nabla f(w_t) = 0$$

(Su et al. 2016, Shi et al., 2021; Attouch et al., 2022)

ODE Framework for First Order Momentum Methods

北京大学 PEKING UNIVERSITY

Inertia System with Hessian Damping (ISHD)

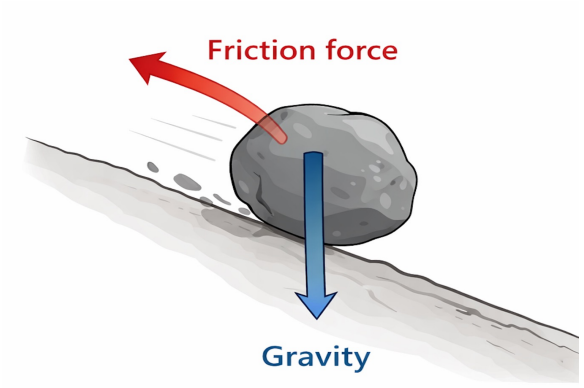
(Su et al. 2016, Shi et al., 2021; Attouch et al., 2022)

$$\ddot{w}_t + \alpha\dot{w}_t + \beta\eta\nabla^2 f(w_t)\dot{w}_t + \eta(\alpha\beta+1)\nabla f(w_t) = 0$$

acceleration velocity

Force = Mass x Acceleration

$$\text{Acceleration}(\ddot{w}_t) = \underbrace{-\alpha\dot{w}_t - \beta\eta\nabla^2 f(w_t)\dot{w}_t}_{\text{Damping or momentum (opposite to velocity)}} \underbrace{- \eta(\alpha\beta+1)\nabla f(w_t)}_{\text{driving force}}$$



ISHD characterizes the role of gradient and momentum (but not preconditioner)

Riemannian ODE Framework for Adaptive Optimizers

北京大学 PEKING UNIVERSITY

A unified form for adaptive optimizers

$$\begin{cases} m_k = (1-\alpha)m_{k-1} + \nabla f(w_k), \\ w_{k+1} = w_k - \eta_k \boxed{F(w_k)^{-1}}(m_k + \beta \nabla f(w_k)), \end{cases}$$

Preconditioner

This form subsumes various adaptive optimizers:

Algorithms	Matrix Form	Vector Form	F (up to a constant factor)
N-AdamW	$\frac{M+\beta g}{\sqrt{V}}$	$\operatorname{diag} v^{-\frac{1}{2}}(m+\beta g)$	$(\operatorname{diag} \mathbb{E}[gg^\top])^{\frac{1}{2}}$
Muon	$M_\beta (M_\beta^\top M_\beta)^{-\frac{1}{2}}$	$((M_\beta^\top M_\beta) \otimes I)^{-\frac{1}{2}}(m+\beta g)$	$(\mathbb{E}[G^\top G])^{\frac{1}{2}} \otimes I$
SOAP	$Q_l \frac{Q_l^\top M Q_r}{\sqrt{V_{\text{rot}}}} Q_r^\top$	$(Q_r \otimes Q_l)(\operatorname{diag} v_{\text{rot}})^{-\frac{1}{2}}(Q_r^\top \otimes Q_l^\top)m$	$(Q_r \otimes Q_l)(\operatorname{diag} \mathbb{E}[g_{\text{rot}} g_{\text{rot}}^\top])^{\frac{1}{2}}(Q_r^\top \otimes Q_l^\top)$

Riemannian ODE Framework for Adaptive Optimizers

- In parallel with the Euclidian ISHD, we have

A unified form of adaptive methods

$$\begin{cases} m_k = (1-\alpha)m_{k-1} + \nabla f(w_k), \\ w_{k+1} = w_k - \eta_k F(w_k)^{-1}(m_k + \beta \nabla f(w_k)), \end{cases}$$

continuization →

← discretization

First order Riemannian ODE system

$$\begin{cases} \dot{w}_t = -\eta_t F(w_t)^{-1}(m_t + \beta \nabla f(w_t)), \\ \dot{m}_t = -\alpha m_t + \nabla f(w_t), \end{cases}$$

Second order ODE system

Proposition 1. Adaptive optimizers can be formulated into **Riemannian ISHD**

$$\nabla^F_{\dot{w}_t} \dot{w}_t + \alpha_t \dot{w}_t + \beta_t \operatorname{Hess}_F(w_t)\dot{w}_t + \gamma_t \operatorname{grad}_F f(w_t) = 0$$

Riemannian acceleration

Riemannian Hessian

Riemannian gradient

Riemannian ODE Framework for Adaptive Optimizers

北京大学 PEKING UNIVERSITY

- Momentum methods (Polyak/Heavy ball/Nesterov) are discrete approximation of **Euclidian ISHD**

$$\ddot{w}_t + \alpha\dot{w}_t + \beta\eta\nabla^2 f(w_t)\dot{w}_t + \eta(\alpha\beta+1)\nabla f(w_t) = 0$$

- Adaptive methods (Adam/Muon/SOAP) are discrete approximation of **Riemannian ISHD**

$$\nabla^F_{\dot{w}_t}\dot{w}_t + \alpha_t\dot{w}_t + \beta_t \underbrace{\mathrm{Hess}_F(w_t)}_{F^{-1}(w)\nabla^2 f(w)}\dot{w}_t + \gamma_t \underbrace{\mathrm{grad}_F f(w_t)}_{F^{-1}(w)\nabla f(w)} = 0$$

The first result to clarify the continuous dynamics for adaptive methods

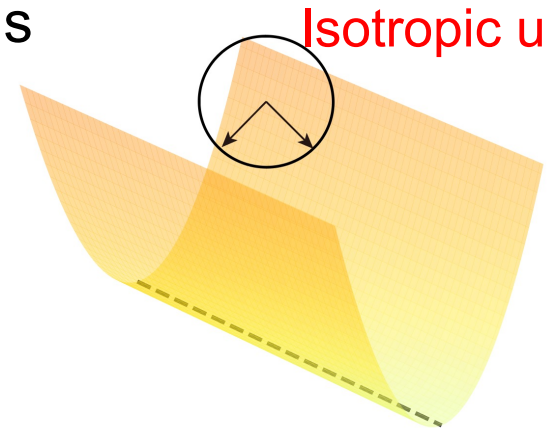
Understanding Preconditioner & Momentum via Riemannian ODE

北京大学 PEKING UNIVERSITY

Newton's second law: F=ma

$$\nabla^F_{\dot{w}_t} \dot{w}_t = \underbrace{-\alpha_t \dot{w}_t - \beta_t \mathrm{Hess}_F(w_t)\dot{w}_t}_{\text{Friction or damping (opposite to velocity)}} \underbrace{- \gamma_t \mathrm{grad}_F f(w_t)}_{\text{Driving force}}$$

- The **preconditioner *F*** induces a Riemmanian metric. Under this geometry, the loss landscape is less anistropic ($\mathrm{Hess}(w) \approx F^{-1}(w)\nabla^2 f(w)$ is less ill-conditioned).
- In this *F-induced* geometry, **momentum** acts as a *damping*.



北京大学
PEKING UNIVERSITY

PART 04

LITE: AcceLerating adaptIve opTimizers in prE-training

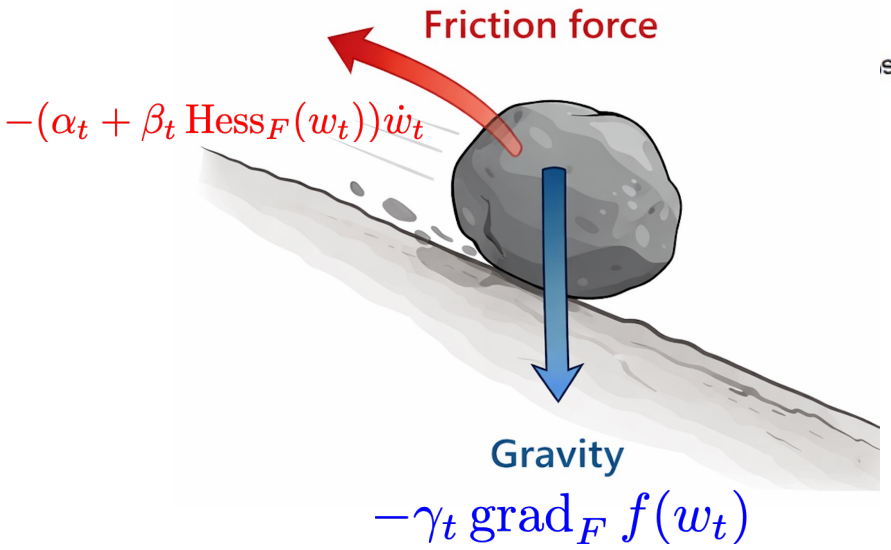
Acceleration Methods by ODE Viewpoints

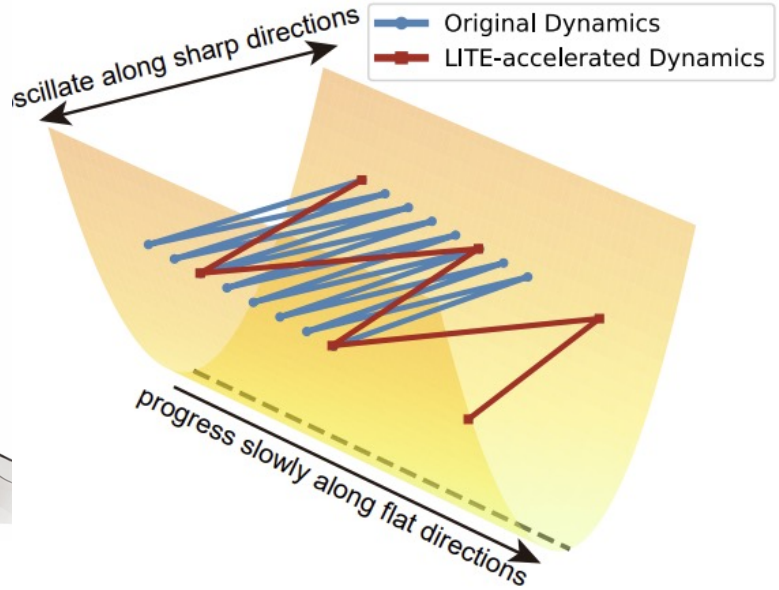
$$\nabla^F_{\dot{w}_t} \dot{w}_t = \underbrace{-\alpha_t \dot{w}_t - \beta_t \mathrm{Hess}_F(w_t)\dot{w}_t}_{\text{Friction or damping (opposite to velocity)}} \underbrace{- \gamma_t \mathrm{grad}_F f(w_t)}_{\text{Driving force}}$$

In the "flat river" directions, $\mathrm{Hess}_F(w)$ is negative or close to 0

To accelerate convergence

- **amplify force: increase γ**
- **reduce damping: increase β**

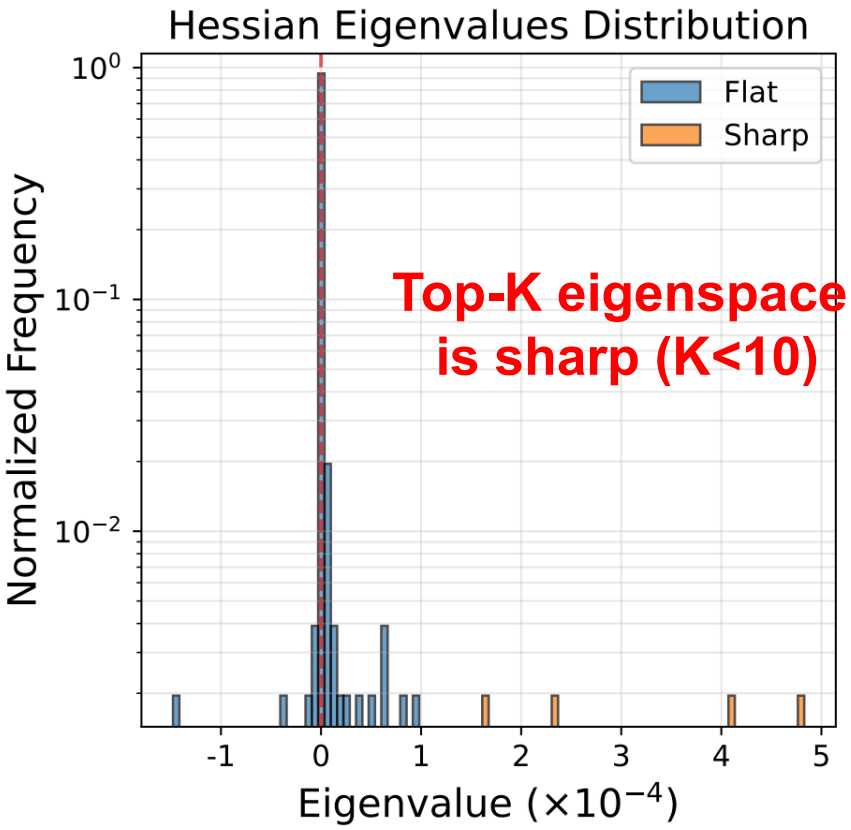




LITE: acceLerating adaptIve opTimizers in LLM prE-training

北京大学 PEKING UNIVERSITY

1. Identify the sharp and flat directions



Top-K eigenspace of GG^T aligns with Hessian well

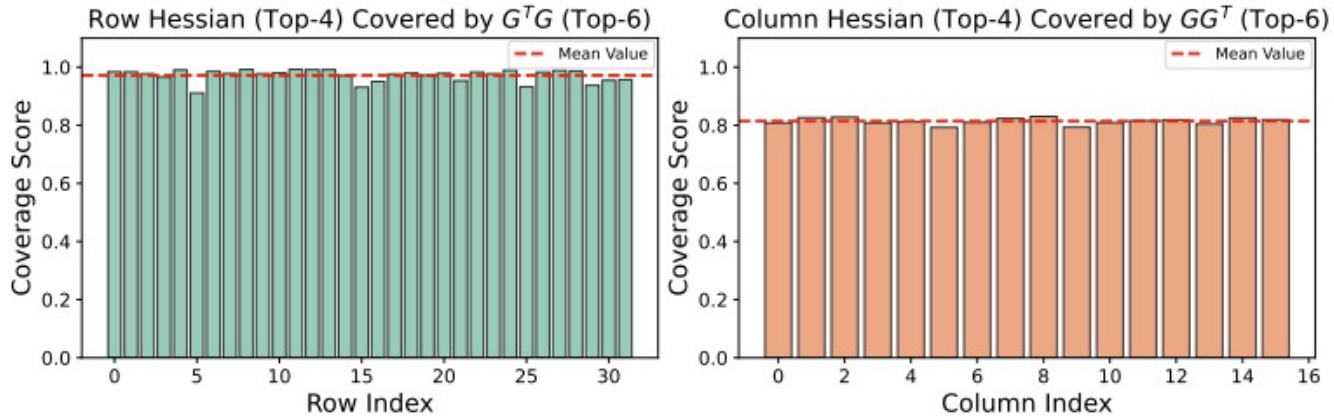


Figure 3. Coverage of the top eigenspaces of row (column) Hessians by those of $G^\top G$ ($GG^\top$) for the `up_proj` block of an FFN layer. A higher score indicates a greater degree of containment.

LITE: acceLerating adaptIve opTimizers in LLM prE-training

北京大学 PEKING UNIVERSITY

2. Determine the acceleration strategy in the flat direction

- **amplify force: increase γ**
- **reduce damping: increase β**

$$\nabla^F_{\dot{w}_t}\dot{w}_t + \alpha_t\dot{w}_t + \beta_t \mathrm{Hess}_F(w_t)\dot{w}_t + \gamma_t \mathrm{grad}_F f(w_t) = 0$$

From second-order to first-order

$$\begin{cases} \dot{w}_t = -\eta_t F(w_t)^{-1}(m_t + \beta\nabla f(w_t)), \\ \dot{m}_t = -\alpha m_t + \nabla f(w_t), \end{cases}$$

discretization

$$\begin{cases} m_k = (1-\alpha)m_{k-1} + \nabla f(w_k), \\ w_{k+1} = w_k - \eta_k F(w_k)^{-1}(m_k + \beta\nabla f(w_k)), \end{cases}$$

Strategy: increase η and β in the flat direction

LITE: acceLerating adaptIve opTimizers in LLM prE-training

北京大学 PEKING UNIVERSITY

Algorithm 1 LITE Strategy

Input: $w_0, m_0, \{\eta_k\}, \alpha, \chi \geq 1, \beta_2 \geq \beta_1 \geq 0$.
for $k = 0$ **to** K **do**
  Update momentum $m_k = (1-\alpha)m_{k-1} + \nabla f(w_k)$.
  Estimate projection Q_k onto the flat direction.
  Estimate the update direction

$$u_k = (I - Q_k)F(w_k)^{-1}(m_k + \beta_1 \nabla f(w_k)) \quad \triangleright \text{ Sharp direction}$$
$$+ \chi Q_k F(w_k)^{-1}(m_k + \beta_2 \nabla f(w_k)). \quad \triangleright \text{ Flat direction (enhance)}$$

  Update parameters $w_{k+1} = w_k - \eta_k u_k$.
end for

Example I: Accelerating Muon (Muon-LITE)

Muon update: $U_k = (M_k + \beta G_k)(M_k^\top M_k)^{-1/2}$

MUON-LITE

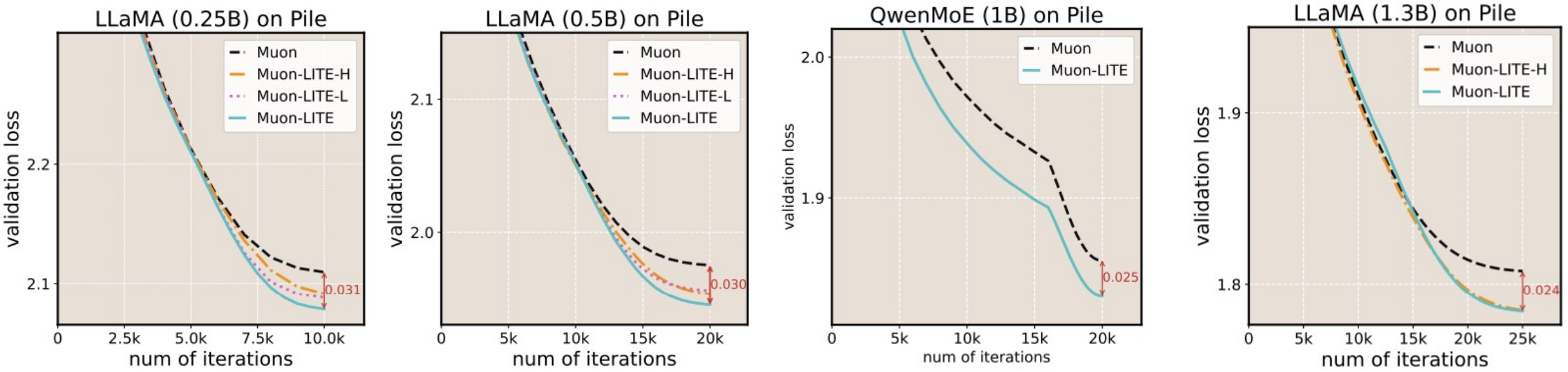
Sharp directions are estimated by Top-d eigenspace of $M_k^\top M_k$.

$$U_k = (M_k + \beta_1 G_k)(M_k^\top M_k)^{-1/2} P_k + \chi(M_k + \beta_2 G_k)(M_k^\top M_k)^{-1/2}(I_n - P_k)$$

Sharp direction **Flat direction**

- $\beta_2 \geq \beta_1$: Reducing damping
- $\chi \geq 1$: Increasing learning rate (and hence the driving force)

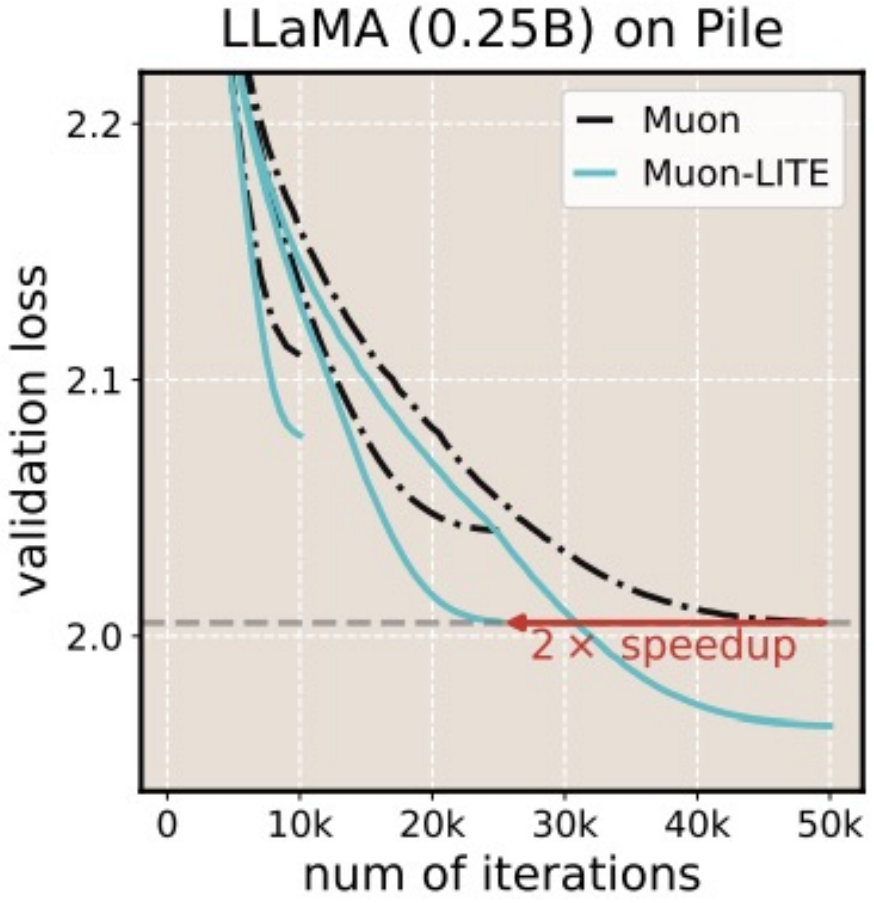
Experiments: Muon-LITE on LLaMA2 and Qwen2MoE Models

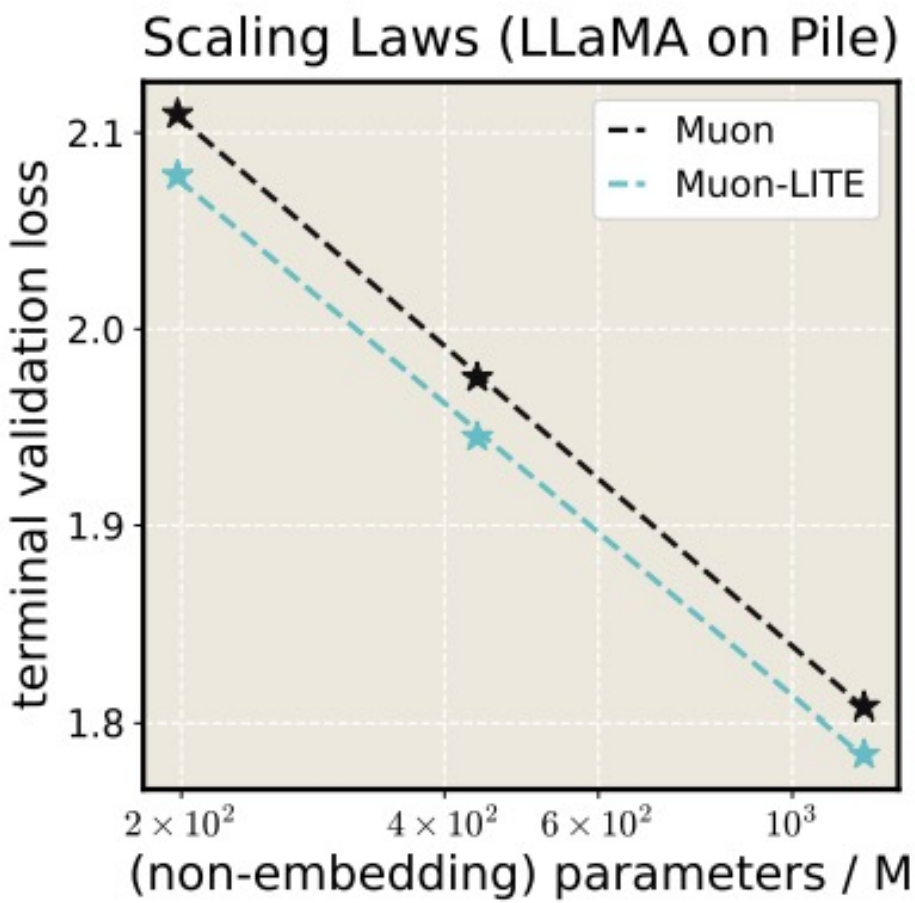


- Either increase the learning rate (L) or Hessian damping coefficient (H) can yield faster training.
- Increasing both the learning rate (L) and Hessian damping coefficient (H) simultaneously is better than increasing either one individually.

Remark: The muon baseline utilizes the RMS norm alignment with Nesterov momentum version from Moonshot.

Experiments: Muon-LITE shows superior scaling laws over Muon





LITE shows superior scaling laws on training data budgets and model sizes.

Experiments: Downstream Tasks

北京大学 PEKING UNIVERSITY

Table 4. Evaluation results on downstream tasks (0-shot with lm-evaluation-harness) of LLaMA models (1.3B) pre-trained on Pile using MUON-LITE and Muon. The best scores in each column are bolded. Abbreviations: T_mc2=TruthfulQA_mc2, AVG=Average score.

Method	ARC_E	ARC_C	PIQA	HellaSwag	BOOLQ	WinoGrande	MMLU	T_mc2	AVG
Muon	52.99	22.10	67.41	35.30	52.45	53.35	23.76	40.14	43.44
MUON-LITE	**54.38**	**23.89**	**67.63**	**36.72**	**60.52**	**54.78**	**25.15**	**40.83**	**45.49**

Enhance model performance across diverse domains ranging from **common sense reasoning** to **truthful QA**

Efficient Implementation of Muon-LITE

An efficient method for estimating Top-d_s eigenspace of $\widetilde{M}_k^\top \widetilde{M}_k$:

1. At iteration *k*, suppose we have a threshold estimate τ_k.

2. Computing the filtering matrix

Highly efficient, only induce a throughput drop **≈1%** for training a LLaMA 1.3B model with token batch size 8192×1024 on 8×A100 80G GPU.

$$T_k = \frac{1}{2}\,\mathrm{NS}(\widetilde{M}_k) + \frac{1}{2}\,\mathrm{NS}\left(\frac{\widetilde{M}_k}{\tau_k} - \mathrm{NS}(\widetilde{M}_k)\right)$$

where $NS(M) = M(M^TM)^{-1/2} = UV$ is conputed by Newton-Schulz iterations, and $M = U\Sigma V$ (SVD)

3. Compute the projection to the Top-d_s eigenspace by $P_k = T_k^{\mathrm{T}} T_k$, the set τ_{k+1} by adjust τ_k based on the trace of P_k.

Example II: Accelerating SOAP (SOAP-LITE)

SOAP update:

Matrix Form

$$Q_l \frac{Q_l^\top M Q_r}{\sqrt{V_{\text{rot}}}} Q_r^\top$$

$\longleftrightarrow$

Vector Form

$$(Q_r \otimes Q_l)(\operatorname{diag} v_{\text{rot}})^{-\frac{1}{2}}(Q_r^\top \otimes Q_l^\top)m$$

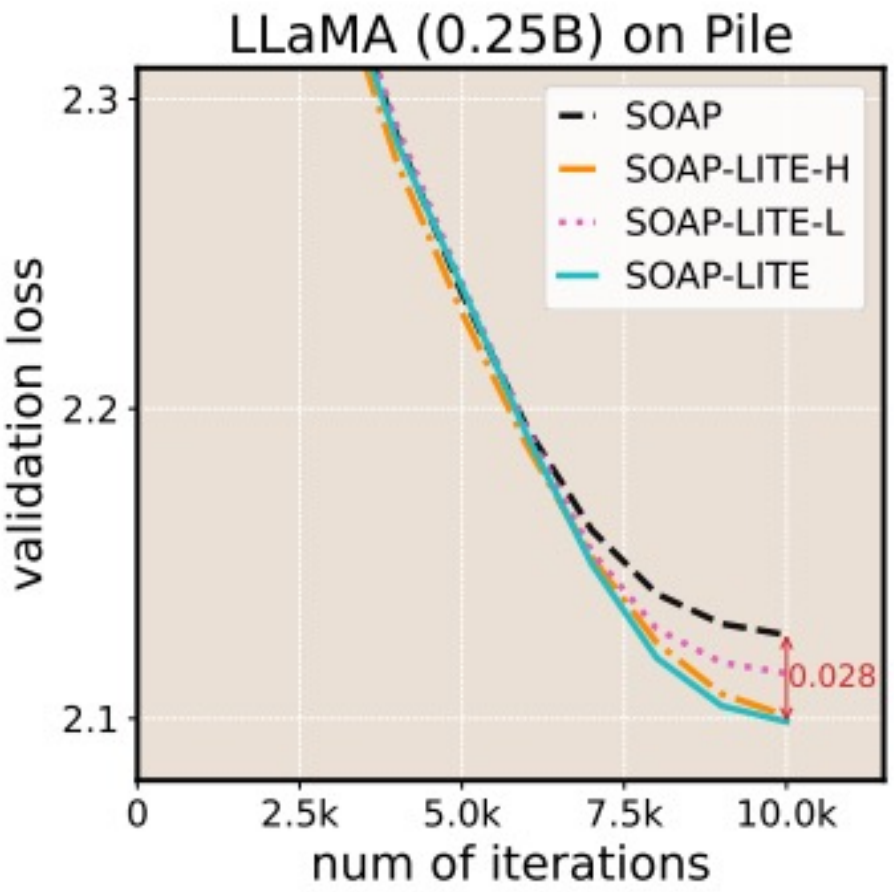
where Q_l and Q_r are estimated eigen-basis of $EMA(GG^T)$ and $EMA(G^TG)$ respectively.

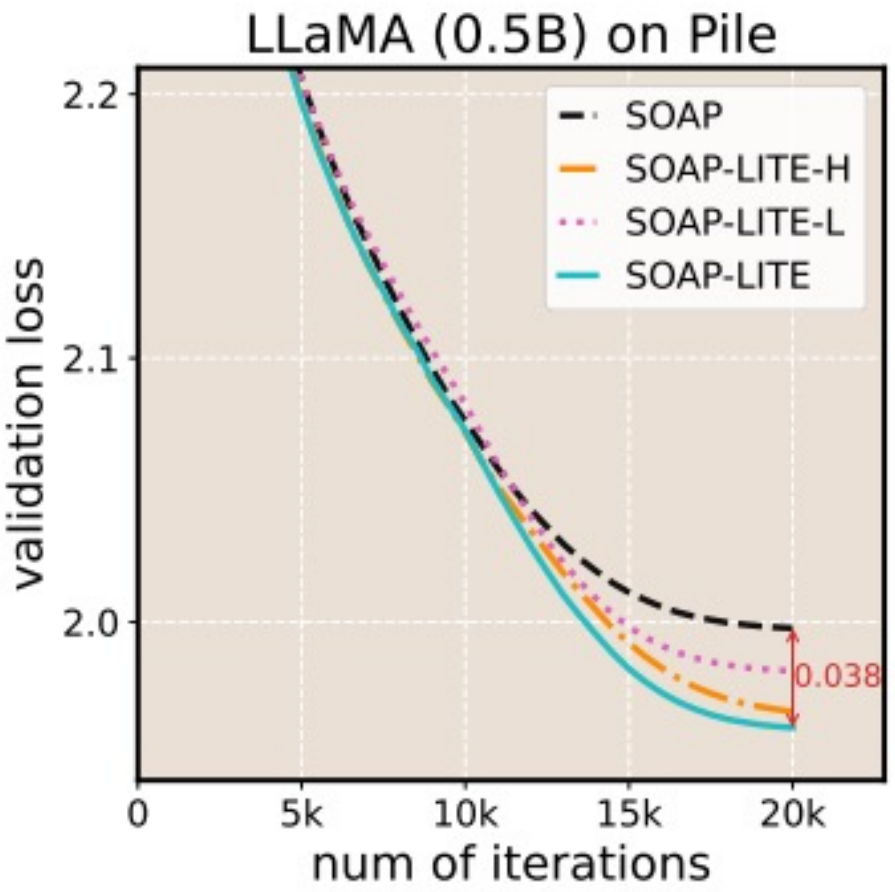
SOAP-LITE

$$U_k = Q_l \left(V_k^{-\frac{1}{2}} \odot (\beta_1 P_k + \chi\beta_2 Q_k) \odot (Q_l^\top G_k Q_r) \right) Q_r^\top$$
$$+ Q_l \left(V_k^{-\frac{1}{2}} \odot (P_k + \chi Q_k) \odot (Q_l^\top M_k Q_r) \right) Q_r^\top,$$

where P_k is the Top-d_s coordinates of V_k and $Q_k = 1_m 1_n^\top - P_k$

Experiments: SOAP-LITE on LLaMA2 Models





PART 05

Applications: OptProver

ByteDance | Seed

Scaling up Multi-Turn Off-Policy RL and Multi-Agent Tree Search for LLM Step-Provers

Ran Xin[1,†,¶], **Zeyu Zheng**[2,*,¶], **Yanchen Nie**[3,*,¶], **Kun Yuan**[3], **Xia Xiao**[1,†]

[1]ByteDance Seed, [2]Carnegie Mellon University, [3]Peking University

[¶]Equal contribution, [*]Work done at ByteDance Seed, [†]Corresponding authors

Open-Source 7B and 32B Theorem Provers

北京大学 PEKING UNIVERSITY

BFS-Prover-V2-7B (this work)	accumulative	**82.4%**	-	-
BFS-Prover-V2-32B (this work)	accumulative	**86.1%**	**85.5%**	**41.4%**
w/ Planner	accumulative	**95.1%**	**95.5%**	-
Whole-proof provers				
DeepSeek-Prover-V2-671B [23]	8192	88.9%	90.6%	37.1%
Kimina-Prover-72B† [30]	1024	87.7%	-	-
w/ TTRL search	accumulative	92.2%	-	-
Goedel-Prover-32B† [17]	8192	92.2%	-	-
w/ Self-correction	1024	92.6%	-	-
Delta-Prover† [43]	accumulative	95.9%	-	-
Seed-Prover† [6]	accumulative	99.6%	-	-

Table 1 Comparison between BFS-Prover-V2 and other leading theorem provers. † denotes concurrent work.

Comparable with whole-proof provers

```
import Mathlib
import LLMlean


open Real


example (a b : ℝ) (f : ℝ → ℝ) : f ((a+b)^2) = f (a^2 + 2*a*b + b^2) := by
  sorry


example (a b c : ℝ) (h : a = b + c) : exp (2 * a) = (exp b) ^ 2 * (exp c) ^ 2 := by
  sorry


def even_fun (f : ℝ → ℝ) := ∀ x, f (-x) = f x
example (f g : ℝ → ℝ) (hf : even_fun f) (hg : even_fun g) : even_fun (f + g) := by
  sorry


example (a b c : ℤ) (h₁ : a ∣ b) (h₂ : b ∣ c) : a ∣ c := by
  sorry


def conjugate {G : Type*} [Group G] (x : G) (H : Subgroup G) : Subgroup G where
  carrier := {a : G | ∃ h, h ∈ H ∧ a = x * h * x⁻¹}
  one_mem' := by
    dsimp
    sorry
  inv_mem' := by
    dsimp
    sorry
  mul_mem' := by
    dsimp
    sorry
```

No info found.

▶ All Messages (7)

From Olympiad Prover to Optimization Prover

北京大学 PEKING UNIVERSITY

- BFS Prover is trained to solve Olympiad problems; **it is not an expert on optimization**
- We build an optimization problem set: **OptBench**

- **Basics** (121 problems): fundamental theorems essential for proving optimization properties

Let E be a real inner product space. Assume that $f : E \to \mathbb{R}$ is differentiable and let $f'(y)$ denote the gradient of f at point y. Prove that for any vectors $x, p \in E$, there exists a real number t with $0 < t < 1$ such that:

$$f(x+p) = f(x) + \langle f'(x+t \cdot p), p \rangle.$$

- **Convexity** (135 problems): properties for convex optimization

Definition: Let E be a real inner product space, $s \subseteq E$ be a set, and $f : E \to \mathbb{R}$ be a function. For a point $x \in E$, the *subdifferential* of f at x within s, is defined as the set of all vectors $g \in E$ such that: $\forall y \in s, f(y) \geq f(x) + \langle g, y - x \rangle$.
Theorem: Let E be a real inner product space, s be a set in E and $f : E \to \mathbb{R}$ be a function. If f is convex on s and continuous on the interior of s, then for every $x \in \text{int}(s)$, the subdifferential is non-empty.

- **Algorithmic** (144 problems): problems from numerical algorithm

Let E be a real Hilbert space and let $f : E \to \mathbb{R}$ be a differentiable function. Let $f'(x)$ denote the gradient of f at x. Assume that the gradient f' is Lipschitz continuous with constant $l > 0$. Suppose the step size $a \in \mathbb{R}$ satisfies $0 < a \leq \frac{1}{l}$. Then, for all $x \in E$, the following inequality holds:

$$f(x - a \cdot f'(x)) \leq f(x) - \frac{a}{2}\|f'(x)\|^2.$$

From Olympiad Prover to Optimization Prover

北京大学 PEKING UNIVERSITY

- Existing Lean Provers **cannot** prove optimization well

Table 2. Performance comparison on OptBench. Whole-proof methods use Pass@32/256, while step-level methods use Pass@1/32. Basic, convex, and algorithmic are the three subcategories, while Avg indicates the average accuracy across the full benchmark.

	Basic		Convex		Algorithmic		Avg	
Method	Pass@32	Pass@256	Pass@32	Pass@256	Pass@32	Pass@256	Pass@32	Pass@256
Whole-proof Generation Baselines								
DeepSeek-Prover-V2-7B	14.88%	19.01%	23.70%	31.85%	4.86%	6.25%	14.25%	18.75%
Goedel-Prover-V2-8B	14.05%	19.01%	19.26%	32.59%	4.86%	8.33%	12.50%	19.75%
Kimina-Prover-7B	4.96%	10.74%	10.37%	16.30%	2.08%	5.56%	5.75%	10.75%

	Basic		Convex		Algorithmic		Avg	
Method	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32
Step-level Provers								
BFS-Prover-V2-7B	14.05%	36.36%	20.00%	44.44%	6.25%	16.67%	13.25%	32.00%

- We need to fine-tune BFS-Prover with optimization lean data

From Olympiad Prover to Optimization Prover

北京大学 PEKING UNIVERSITY

Table 2. Performance comparison on OptBench. Whole-proof methods use Pass@32/256, while step-level methods use Pass@1/32. Basic, convex, and algorithmic are the three subcategories, while Avg indicates the average accuracy across the full benchmark.

	Basic		Convex		Algorithmic		Avg	
Method	Pass@32	Pass@256	Pass@32	Pass@256	Pass@32	Pass@256	Pass@32	Pass@256
Whole-proof Generation Baselines								
DeepSeek-Prover-V2-7B	14.88%	19.01%	23.70%	31.85%	4.86%	6.25%	14.25%	18.75%
Goedel-Prover-V2-8B	14.05%	19.01%	19.26%	32.59%	4.86%	8.33%	12.50%	19.75%
Kimina-Prover-7B	4.96%	10.74%	10.37%	16.30%	2.08%	5.56%	5.75%	10.75%

	Basic		Convex		Algorithmic		Avg	
Method	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32	Pass@1	Pass@32
Step-level Provers								
BFS-Prover-V2-7B	14.05%	36.36%	20.00%	44.44%	6.25%	16.67%	13.25%	32.00%
OptProver Variants (Ours)								
OptProver (Only EI)	32.23%	49.59%	39.26%	60.00%	15.97%	40.97%	28.75%	50.00%
OptProver (EI + DPO)	34.71%	52.07%	39.26%	62.22%	20.14%	42.36%	31.00%	52.00%
OptProver (EI + UAPO)	36.36%	53.72%	**45.93%**	60.00%	27.08%	47.22%	36.25%	53.50%
OptProver (EI + PW-DPO)	37.19%	53.72%	45.19%	**62.96%**	27.08%	46.53%	36.25%	54.25%
OptProver (EI + PW-UAPO)	**40.50%**	**55.37%**	45.19%	62.22%	**28.47%**	**48.61%**	**37.75%**	**55.25%**

OptProver achieves state-of-the-art performance among comparably sized model.

From Olympiad Prover to Optimization Prover

北京大学 PEKING UNIVERSITY

Statement

Let E be a real inner product space and $f : E \to \mathbb{R}$ be a convex function on E.
For any fixed vector $x \in E$, the function $h(u) = f(u) + \frac{1}{2}\|u - x\|^2$ is **1-strongly convex** on E.

Proof

```
lemma strongconvex_of_convex_add_sq (f : E → ℝ) (x : E) (hfun : ConvexOn ℝ univ f) :
  StrongConvexOn univ (1 : ℝ) fun u ↦ f u + ‖u - x‖ ^ 2 / 2 := by
 rw [StrongConvexOn]
 simp_rw [add_comm (f _)]
 constructor
 · exact convex_univ
 rintro y - z - a b ha hb hab
 simp only [smul_eq_mul, sub_add, div_eq_inv_mul]
 ring_nf
 replace hfun := hfun.2
 specialize hfun (by simp) (by simp) ha hb hab
 exact y
 · exact z
 simp only [_root_.div_eq_inv_mul] at hfun ⊢
 field_simp
 refine (div_le_div_right two_pos).2 ?_
 rw [norm_sub_sq_real, norm_sub_sq_real, norm_sub_sq_real]
 rcases eq_sub_of_add_eq hab with rfl
 simp [norm_add_sq_real, inner_add_left, inner_smul_left]
 rw [norm_smul, norm_smul, inner_smul_right, real_inner_comm]
 simp only [Real.norm_eq_abs, abs_of_nonneg hb, abs_of_nonneg ha, sub_mul, sub_add,
  add_sub_assoc, mul_assoc]
 norm_num
 on_goal 1 => norm_num at *
 have : 0 ≤ 1 - b := by linarith
 conv_lhs => rw [add_comm, add_assoc, add_left_comm]
 ring_nf
 norm_num [real_inner_comm] at *
 ring_nf at *
 repeat' rw [norm_sub_sq_real]
 rw [mul_comm b] at hfun
 nlinarith [real_inner_comm x y, real_inner_comm x z, real_inner_comm y z,
  sq_nonneg (b - 1), sq_nonneg (b - 2), f y, f z]
```

北京大学
PEKING UNIVERSITY

PART 06

Theoretical Analysis on Training Dynamics

Theory: Loss Landscape

北京大学 PEKING UNIVERSITY

Assumption (informal)

Let $P_s(w)$ be the projection onto the top-d_s eigenspace of $\nabla^2 f(w)$. Assume:

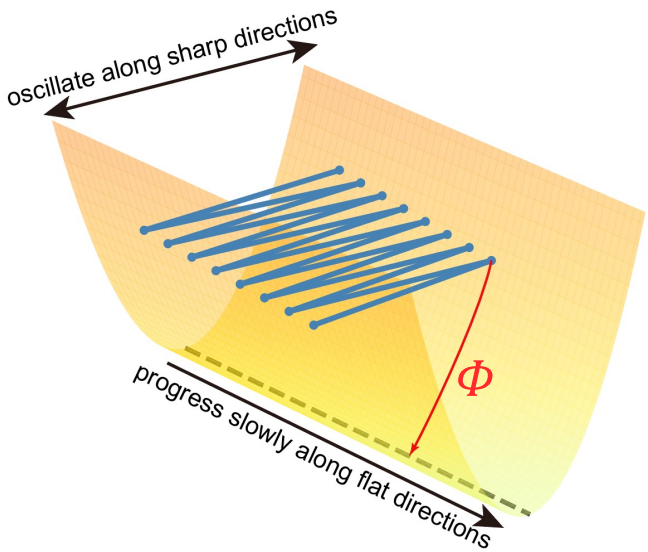
- *the set (referred to as River) $\mathcal{R} = \{w : P_s(w)\nabla f(w) = 0\}$ constitutes a $(p - d_s)$-dimensional manifold.*
- *$\nabla^2 f(w)$ and $F(w)$ share a common eigen-basis in the sharp subspace $\mathrm{Range}(P_s(w))$.*

Define the "projection" Φ to River (descending along Valley):

$$\Phi(w) = \lim_{t\to\infty} \psi_t(w)^1,$$

with $\frac{d}{dt}\psi_t(w) = -P_s(\psi_t(w))F^{-1}(\psi_t(w))\nabla f(\psi_t(w))$ and $\psi_0(w) = w$.

- Intuition: Φ provides a tool to quantify the distance to River.



[1] $\psi_t(w)$ vanishes when $P_s F^{-1}\nabla f(w) = 0$.

The assumptions are based on (Wen et al. 2025).

Theory: Two Stage Training Dynamics

北京大学 PEKING UNIVERSITY

Theorem (Part 1, informal)

Under some regularity assumptions, the trajectory w_t governed by

$$\begin{cases} \dot{w}_t = -\eta_t F(w_t)^{-1} P_t\big(m_t + \beta_1 \nabla f(w_t)\big) \\ \qquad\quad - \chi \eta_t F(w_t)^{-1} Q_t\big(m_t + \beta_2 \nabla f(w_t)\big), \\ \dot{m}_t = -\alpha m_t + \nabla f(w_t). \end{cases}$$

satisfies the following attraction bound for sufficiently small ε:

$$\|w_t - \Phi(w_t)\|_2^2 \lesssim \exp(-\Theta(t)) + \varepsilon.$$

Stage 1: the dynamics of LITE first converge to the $\mathcal{O}(\varepsilon^{1/2})$ neighborhood of River after a $\Theta(\log(\varepsilon^{-1}))$ time interval.

Theory: Two Stage Training Dynamics

北京大学 PEKING UNIVERSITY

Theorem (Part 2, informal)

The "projected" trajectory $z_t = \Phi(w_t)$ *follows*

$$\begin{cases} \dot{z}_t = -\eta_t \Big(\chi P_{\mathcal{R}}(z_t) F(z_t)^{-1} (s_t + \beta_2 \nabla f(z_t)) + \epsilon_{z,t} \Big), \\ \dot{s}_t = -\alpha s_t + \nabla f(z_t), \end{cases}$$

Define

$$b_t = \left\| P_{\mathcal{R}}(z_t) F(z_t)^{-1} \nabla f(z_t) \right\|^2_{F(z_t)} + \left\| P_{\mathcal{R}}(z_t) F(z_t)^{-1} s_t \right\|^2_{F(z_t)}.$$

Then

$$\|\epsilon_{z,t}\|^2_{F(z_t)} \lesssim \exp(-\Theta(t)) + \varepsilon^2 b_t + \varepsilon \int_0^t K(\tau - t) b_\tau \, d\tau = \mathcal{O}(\varepsilon).$$

Stage 2: Both w_t and the projected dynamics $\Phi(w_t)$ track the decoupled acceleration dynamics (set $\epsilon_{z,t} \equiv 0$) on River closely.

Theory: Acceleration by LITE

Larger lr ($\chi \geq 1$) and Hessian damping coefficients ($\beta_2 \geq \beta_1$) foster flat-direction optimization.

Theorem

For the unperturbed projected dynamics ($\varepsilon_{z,t} \equiv 0$)

$$\begin{cases} \dot{z}_t = -\eta_t \Big(\chi P_{\mathcal{R}}(z_t) F(z_t)^{-1} (s_t + \beta_2 \nabla f(z_t)) \Big), \\ \dot{s}_t = -\alpha s_t + \nabla f(z_t), \end{cases}$$

the following bound holds for sufficiently small ε:

$$\frac{1}{T} \int_0^T \eta_t \left\| P_{\mathcal{R}}(z_t) F(z_t)^{-1} \nabla f(z_t) \right\|_{F(z_t)}^2 dt \leq \frac{2(f(z_0) - \inf f)}{\chi \beta_2 T},$$

$$\frac{1}{T} \int_0^T \eta_t \left\| s_t \right\|_{F(z_t)}^2 dt \leq \frac{2(f(z_0) - \inf f)}{\chi \alpha T}.$$

Summary

北京大学 PEKING UNIVERSITY

- Riemannian ODE framework for understanding the joint effect of momentum and preconditioners in the pretraining dynamcis.
- Principled acceleration strategy (LITE) using flat-direction enhancement within the ODE framework.
- Experimental results demonstrate LITE efficiently accelerates pretraining in Muon and SOAP, showing better scaling laws over data and model size.
- Theoretical analysis on the acceleration mechanism of LITE in the anisotropic loss landscapes.

北京大学
PEKING UNIVERSITY

Thank you!

Please refer to more details in *https://arxiv.org/abs/2602.22681*

Paper

Github Code