Maintaining Plasticity for Scalable Deep Reinforcement Learning

KAIST AI, PhD

Hojoon Lee

12.9, 2025

DAVIAN Data and Visual Analytics Lab

KAIST AI Graduate School of AI

Outline

1. Short Bio
2. Introduction
   - Why do we need Scalable RL?
   - Why Scaling is Difficult in Deep RL?
   - Loss of Plasticity: A Key Bottleneck in Scalable Deep RL
3. Maintaining Plasticity for Scalable Deep RL
   - Existing Works on Preventing Plasticity Loss
   - PLASTIC: Smoothing Loss Landscape and Preserving Active Units for Scalable RL (NeurIPS'23)
   - Simba: Simplicity Bias for Scalable RL (ICLR'25 Spotlight)
   - SimbaV2: Hyperspherical Normalization for Scalable RL (ICML'25 Spotlight)
   - FlashSAC: Scaling Off-Policy RL for Speed (RSS'26)
4. Conclusion

Short Bio | Hojoon Lee

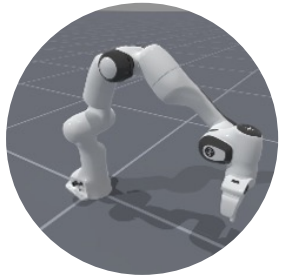
Education

- Ph.D. candidate in Graduate School of AI, KAIST, 2022.03-Current.
- M.S. in Graduate School of AI, KAIST, 2020.03-2022.02.
- B.S in Computer Science, Korea University, 2014.03-2020.02

Work Experience

Meta Reality Lab

May.2025-Nov.2025



Robotic Manipulation

Krafton

Mar.2025-May.2025

Chess Agent

Sony AI

Mar.2024-Sep.2024

GranTurismo Agent

Kakao Enterprise

Sep.2021-Mar.2022

RL Library

Neowiz Games

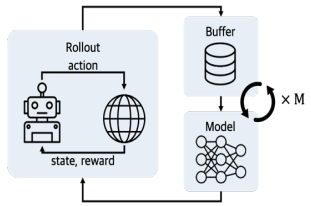
Mar.2019-Aug.2019

BrownDust Agent

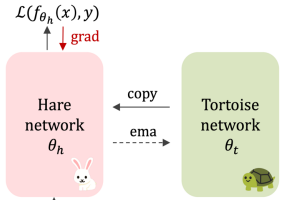
Short Bio | Research Overview

Scalable Deep RL

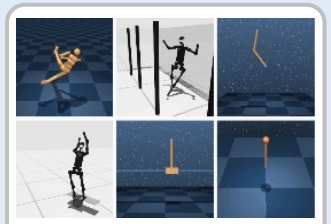
Thesis Topic



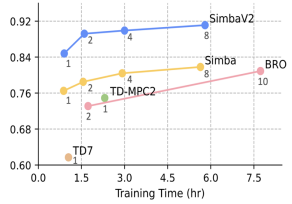
PLASTIC: Adopting SAM optimizer to Scale Compute
NeurIPS'23



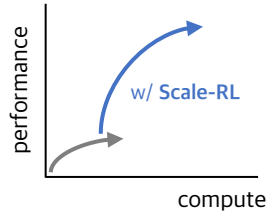
Hare & Tortoise: Maintaining Network Plasticity for CL
ICML'24



Simba: Designing Architecture for Scalable Deep RL
ICLR'25 (spotlight)



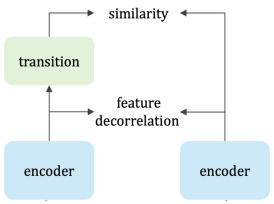
SimbaV2: Designing Architecture for Scalable Deep RL
ICML'25 (spotlight)



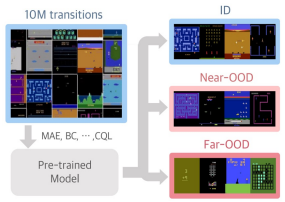
FlashSAC: Scaling Off-Policy RL for Compute Efficiency
on progress

2023 2024 2025

Others

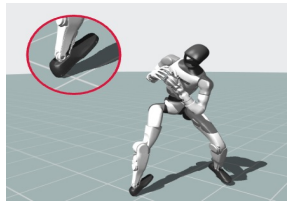


Pre-training with Feature Disentangle Loss in Visual RL
ICML'23

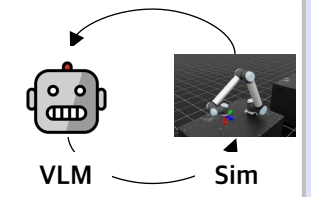


Benchmarking Pre-training Objectives in Visual RL
ICML'24

Vision-Based Autonomous Racing Agent in GT7
RA-L'25



Humanoid Motion Dataset for Robust Locomotion
Under Review



RDA: VLM-Based Reward Design Agent for RL
Under Review

Visual Pre-training for RL

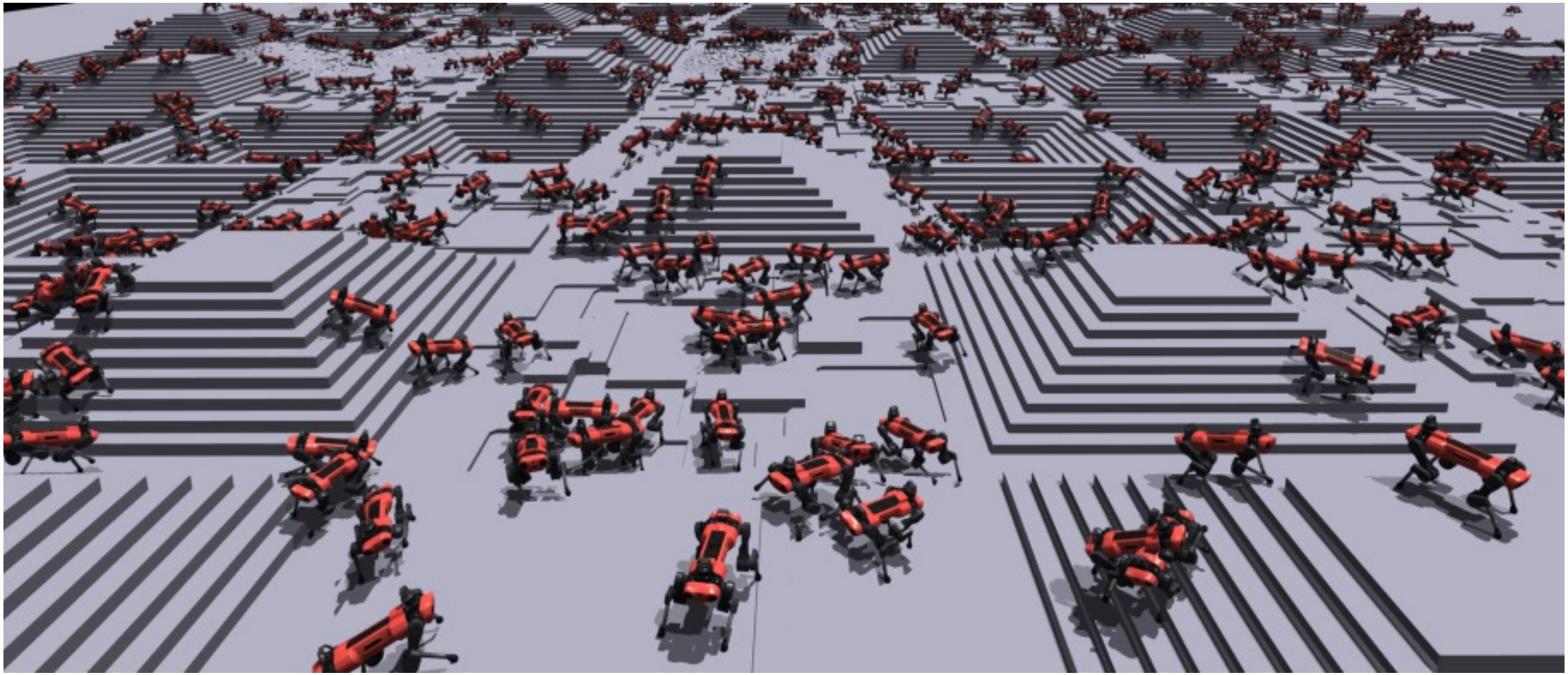
Robotics

Outline

1. Short Bio
2. Introduction
   - Why do we need Scalable RL?
   - Why Scaling is Difficult in Deep RL?
   - Loss of Plasticity: A Key Bottleneck in Scalable Deep RL
3. Maintaining Plasticity for Scalable Deep RL
   - Existing Works on Preventing Plasticity Loss
   - PLASTIC: Smoothing Loss Landscape and Preserving Active Units for Scalable RL (NeurIPS'23)
   - Simba: Simplicity Bias for Scalable RL (ICLR'25 Spotlight)
   - SimbaV2: Hyperspherical Normalization for Scalable RL (ICML'25 Spotlight)
   - FlashSAC: Scaling Off-Policy RL for Speed (preprint)
4. Conclusion

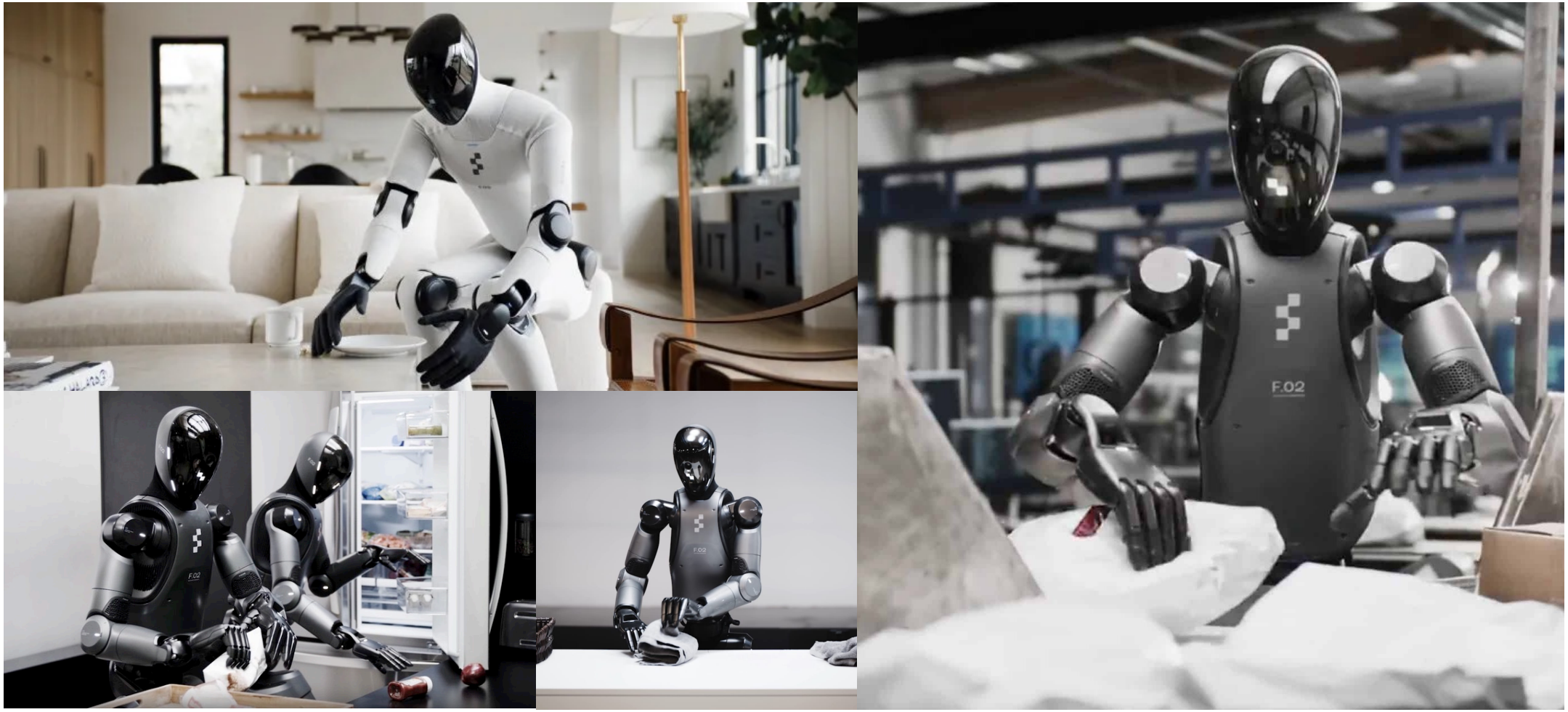
Introduction | Why Do We Need Scalable Deep RL?

Today, robots trained with Deep RL uses small, specialized models.

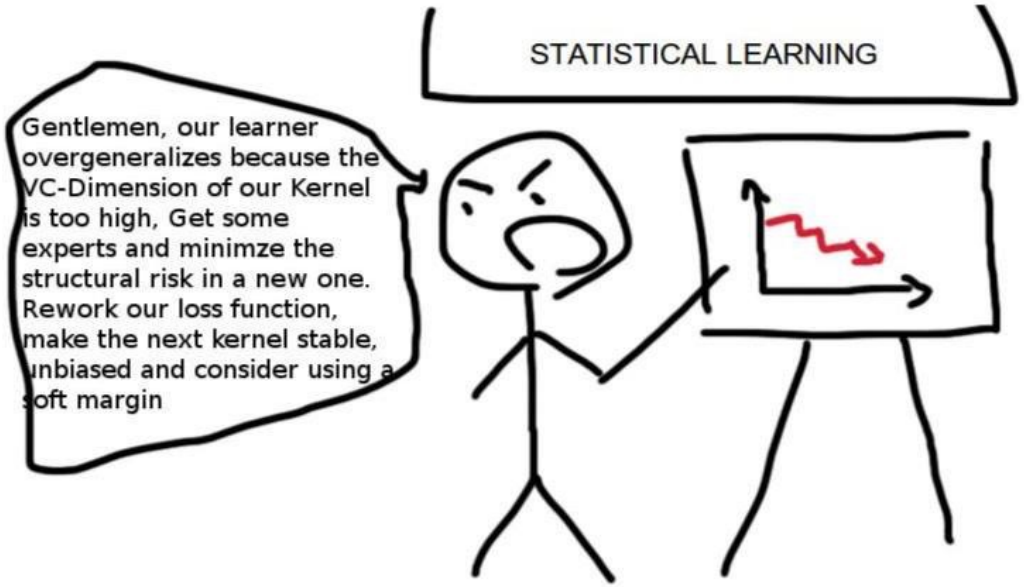


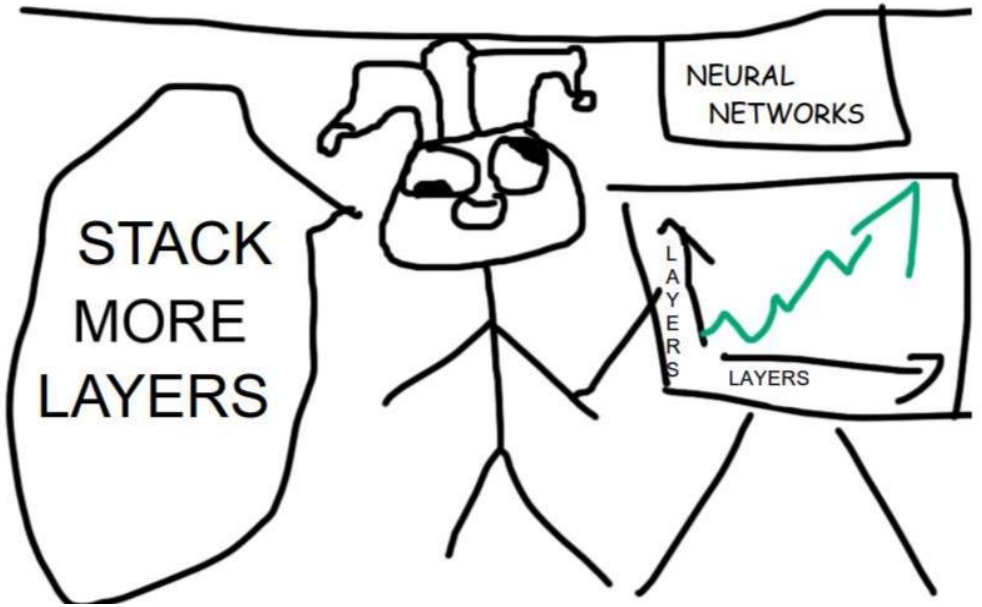
Introduction | Why Do We Need Scalable Deep RL?

However, real-world robots need large models that can generalize across multiple tasks.



Introduction | Scaling: A Simple Recipe in Supervised Learning

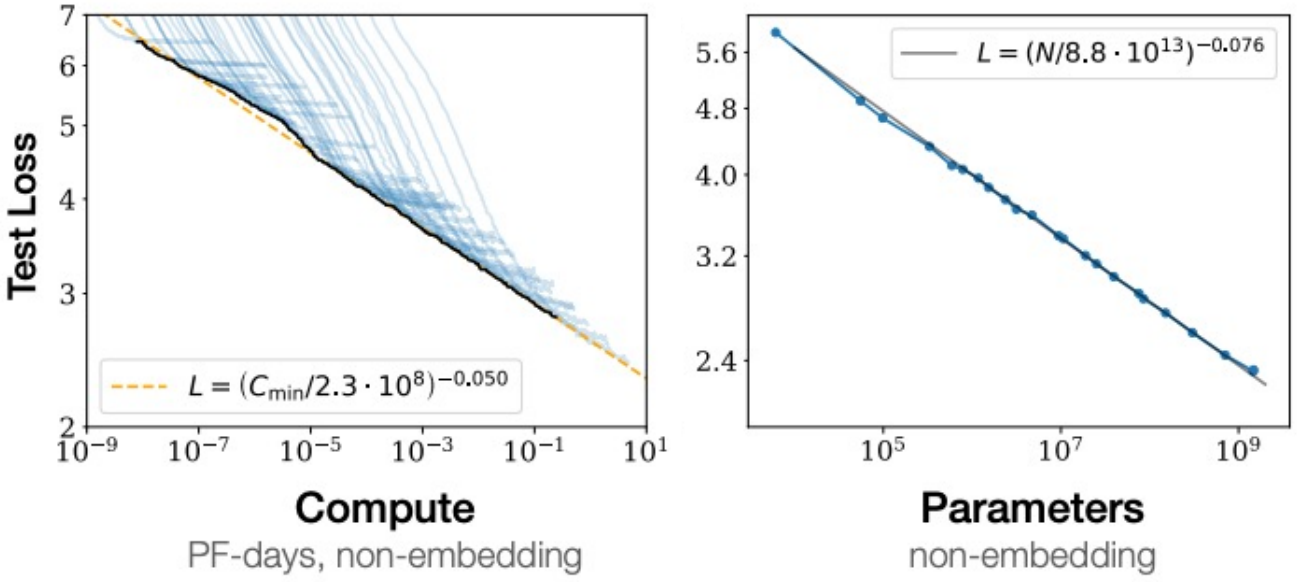


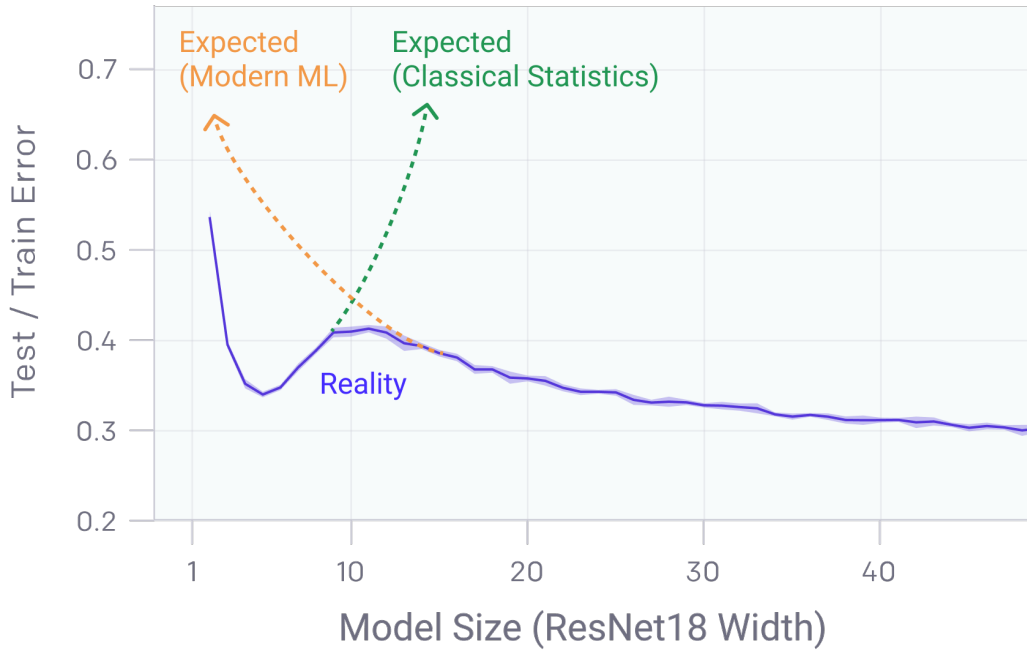


https://www.reddit.com/r/ProgrammerHumor/comments/8c1i45/stack_more_layers/

Introduction | Scaling: A Simple Recipe in Supervised Learning

In supervised learning, neural networks generalize better as model size and compute scales.





[1] Scaling Laws for Neural Language Models, Kaplan et al, arXiv 2020..
[2] Deep Double Descent: Where Bigger Models and More Data Hurt, Nakkiran et al, arXiv 2019.

Introduction | Does Scaling Work in RL?

In general, scaling compute and parameters decreases performance in RL.

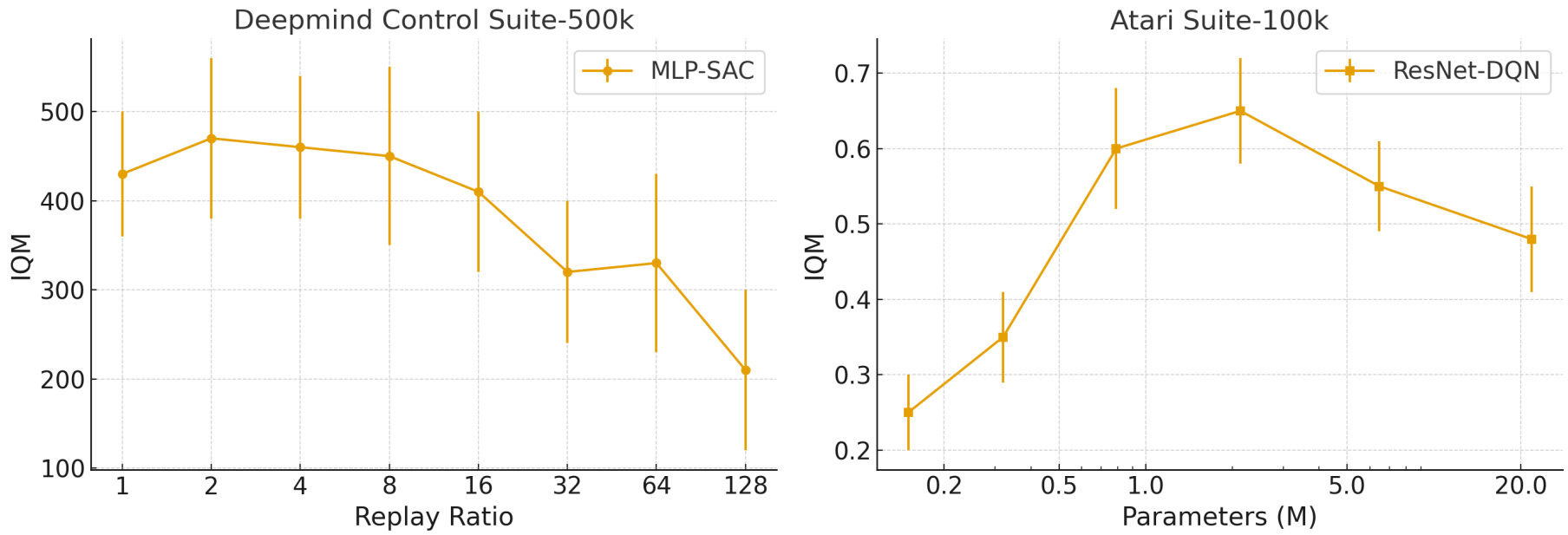


Figure: (Left) Effect of scaling compute for SAC on Deepmind Control Suite with 500K steps.

(Right) Effect of scaling parameters for DQN on Atari tasks with 100k steps.

[1] Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier, D'Oro et al, ICLR 2023.

[2] Bigger, Better, Faster: Human-level Atari with human-level efficiency, Schwarzer et al, ICML 2023

Introduction | Why Scaling is Difficult in RL?

Reinforcement Learning is hard to scale because of:

- Credit Assignment
  - Rewards may be delayed or sparse, hard to identify useful action.
- Reward Design
  - Hard to design rewards that can effectively encode human intention.
- Exploration vs. Exploitation
  - Agent must discover useful behaviors while exploiting what they already know.
- Non-Stationary Optimization
  - The data distribution constantly shifts because the policy generates its own data to train.
  - When learning value function, both the inputs and the targets change over time.

Introduction | Why Scaling is Difficult in RL?

Reinforcement Learning is hard to scale because of:

- Credit Assignment
  - Rewards may be delayed or sparse, hard to identify useful action.
- Reward Design
  - Hard to design rewards that can effectively encode human intention.
- Exploration vs. Exploitation
  - Agent must discover useful behaviors while exploiting what they already know.
- Non-Stationary Optimization
  - The data distribution constantly shifts because the policy generates its own data to train.
  - When learning value function, both the inputs and the targets change over time.

Introduction | Why Scaling is Difficult in RL?

Reinforcement Learning is hard to scale because of:

- Credit Assignment
  - Rewards may be delayed or sparse, hard to identify useful action.
- Reward Design
  - Hard to design rewards that can effectively encode human intention.
- Exploration vs. Exploitation
  - Agent must discover useful behaviors while exploiting what they already know.
- Non-Stationary Optimization
  - The data distribution constantly shifts because the policy generates its own data to train.
  - When learning value function, both the inputs and the targets change over time..

Introduction | Why Scaling is Difficult in RL?

Reinforcement Learning is hard to scale because of:

- Credit Assignment
  - Rewards may be delayed or sparse, hard to identify useful action.
- Reward Design
  - Hard to design rewards that can effectively encode human intention.
- Exploration vs. Exploitation
  - Agent must discover useful behaviors while exploiting what they already know.
- Non-Stationary Optimization
  - The data distribution constantly shifts because the policy generates its own data to train.
  - When learning value function, both the inputs and the targets change over time.

Introduction | Why Scaling is Difficult in RL?

Reinforcement Learning is hard to scale because of:

- Credit Assignment
  - Rewards may be delayed or sparse, hard to identify useful action.
- Reward Design
  - Hard to design rewards that can effectively encode human intention.
- Exploration vs. Exploitation
  - Agent must discover useful behaviors while exploiting what they already know.
- Non-Stationary Optimization
  - The data distribution constantly shifts because the policy generates its own data to train.
  - When learning value function, both the inputs and the targets change over time.

Among many challenges, non-stationary optimization remains relatively less explored.

Introduction | Challenge of Non-Stationary Optimization

Loss of Plasticity [1]

- In non-stationary optimization, neural networks loses its ability to adapt to new data.

[1] Loss of plasticity in deep continual learning, Dohare et al, Nature 2024.

Introduction | Challenge of Non-Stationary Optimization

Loss of Plasticity [1]

- In non-stationary optimization, neural networks loses its ability to adapt to new data.

Synthetic Non-Stationary Optimization Experiment [1]

- **Dataset:** MNIST, CIFAR-100, Tiny-ImageNet.
- **Architecture:** MLP, ViT-Tiny, VGG-16.

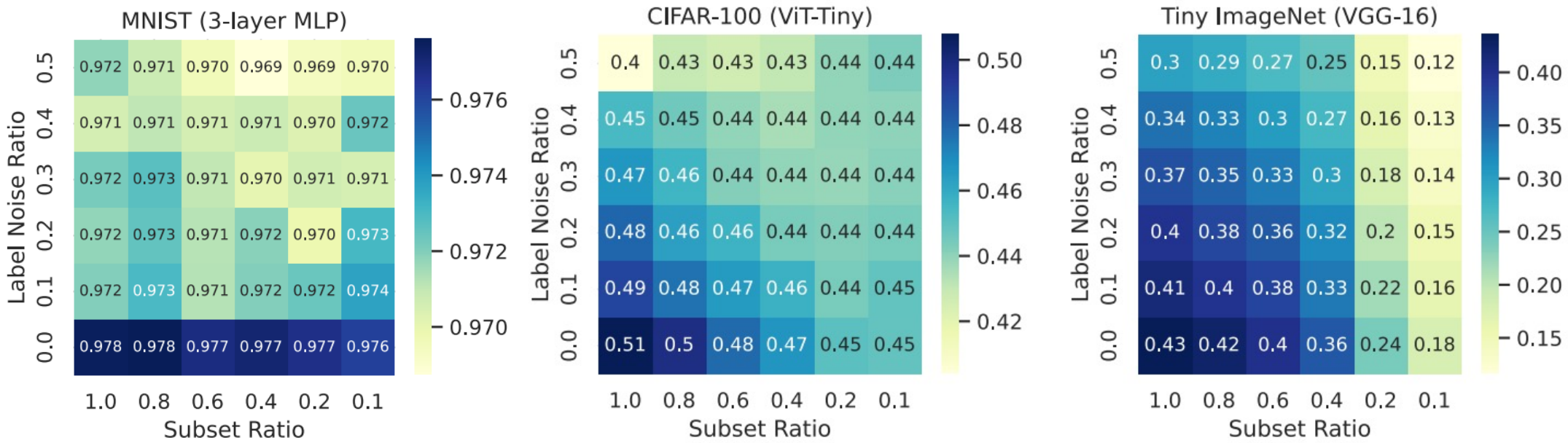
```
# early pre-training
for epoch in range(epochs):
    train the neural network from the subset (p%) of the noisy (q%) dataset.
# late fine-tuning
for epoch in range(epochs):
    train the neural network from the complete noise-free dataset.
evaluate test accuracy.
```

[1] Loss of plasticity in deep continual learning, Dohare et al, Nature 2024.

Introduction | Challenge of Non-Stationary Optimization

Synthetic Non-Stationary Optimization Experiment [1]

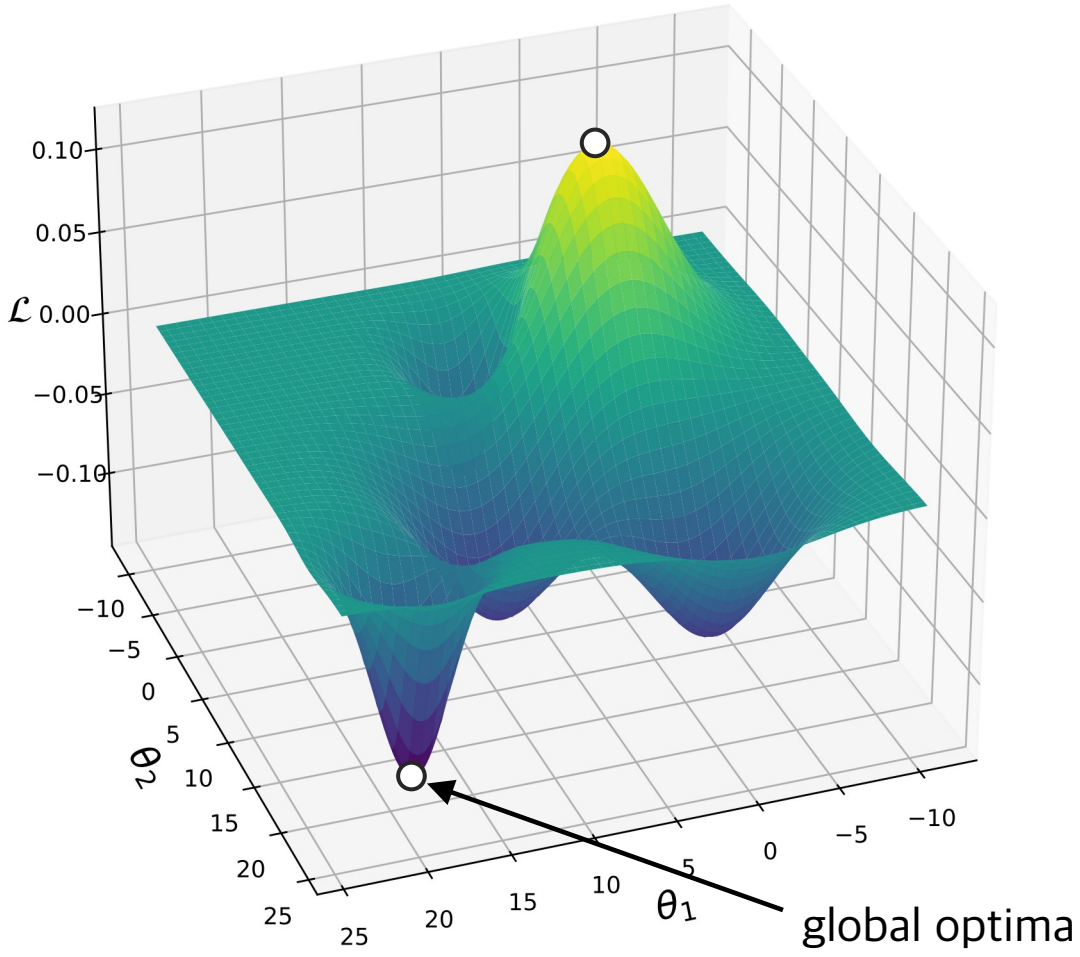
- Network fails to adapt under larger distribution shifts (smaller subsets and noisier labels).
- Larger models (VGG-16) suffer more from these shifts than smaller models (MLP).



*Points in the upper-right corner (↗) indicate smaller subsets and noisier labels.

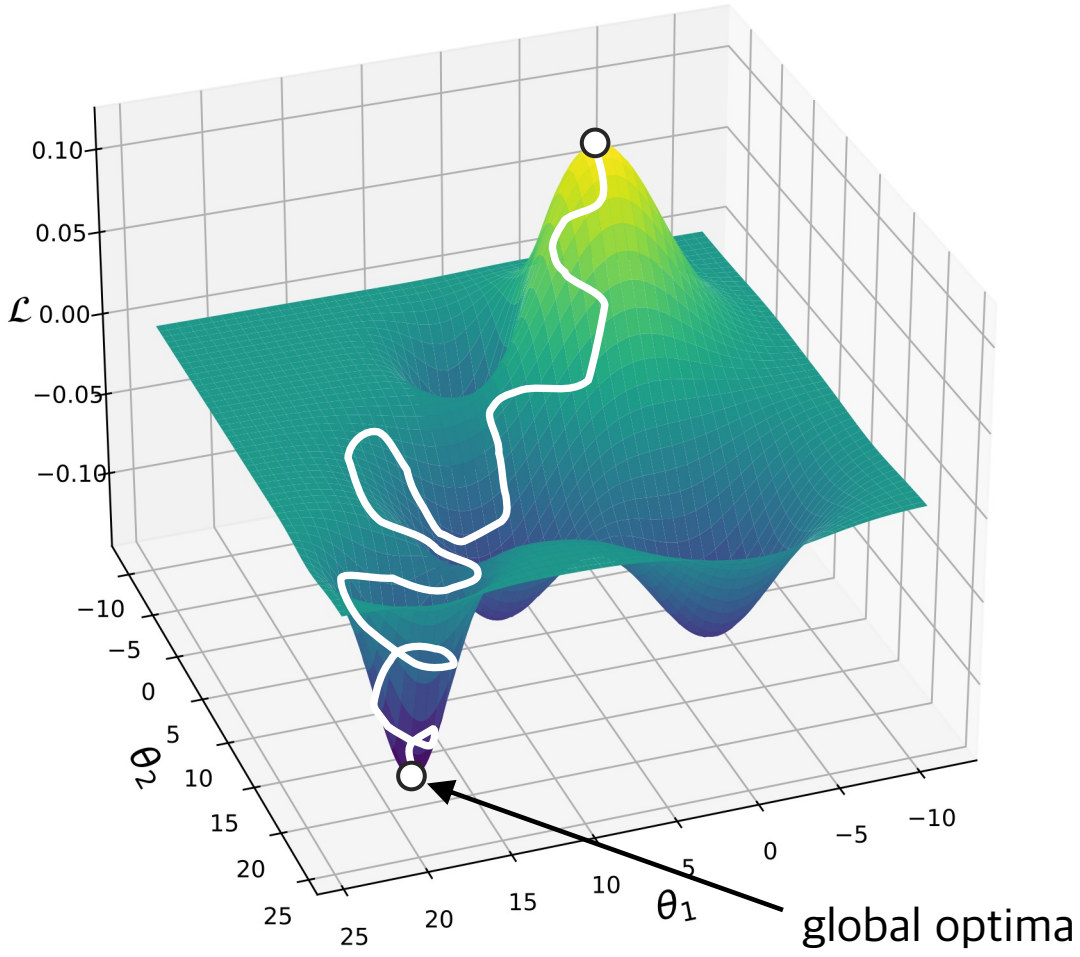
[1] On Warm-Starting Neural Network Training, Ash et al, NeurIPS 2019.

Introduction | Challenge of Non-Stationary Optimization



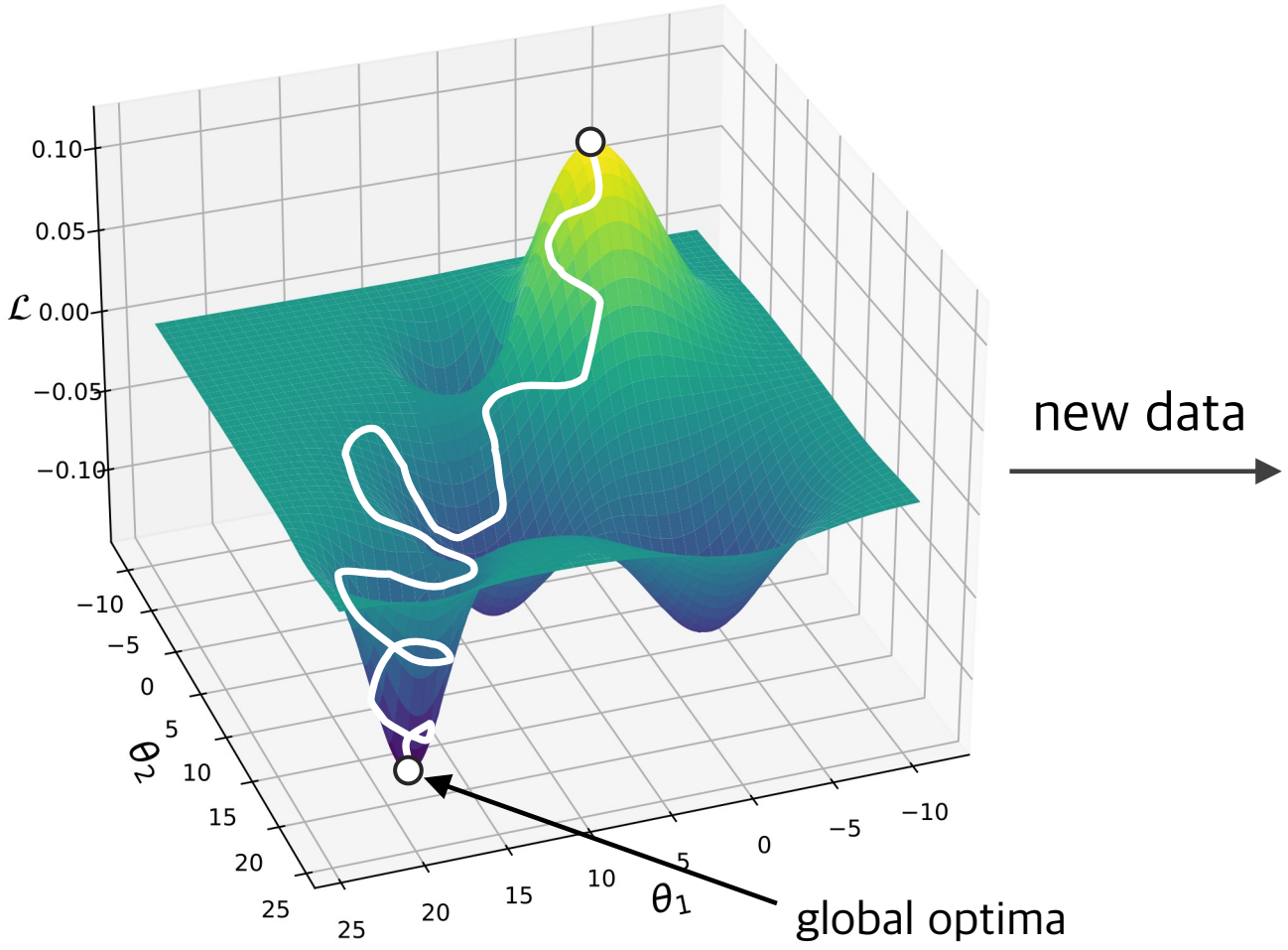
Early-Training Phase

Introduction | Challenge of Non-Stationary Optimization



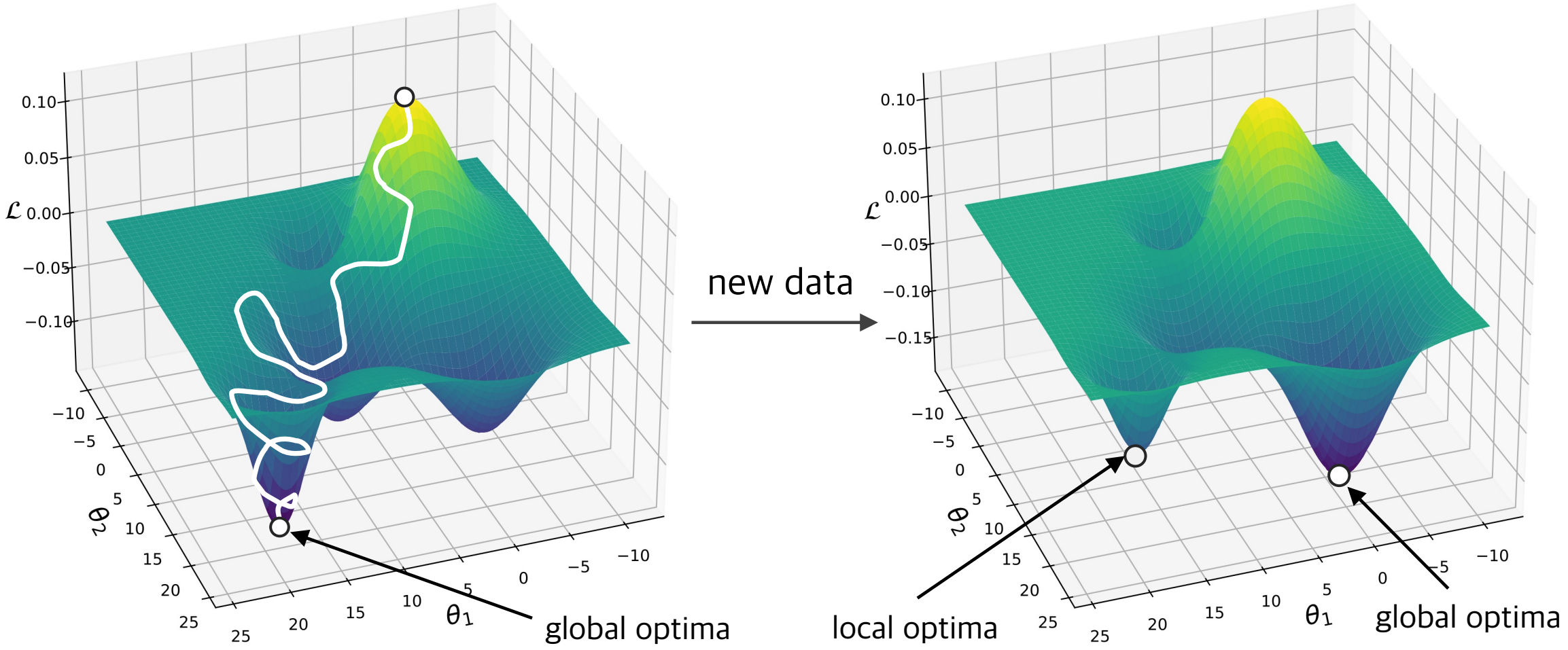
Early-Training Phase

Introduction | Challenge of Non-Stationary Optimization



Early-Training Phase

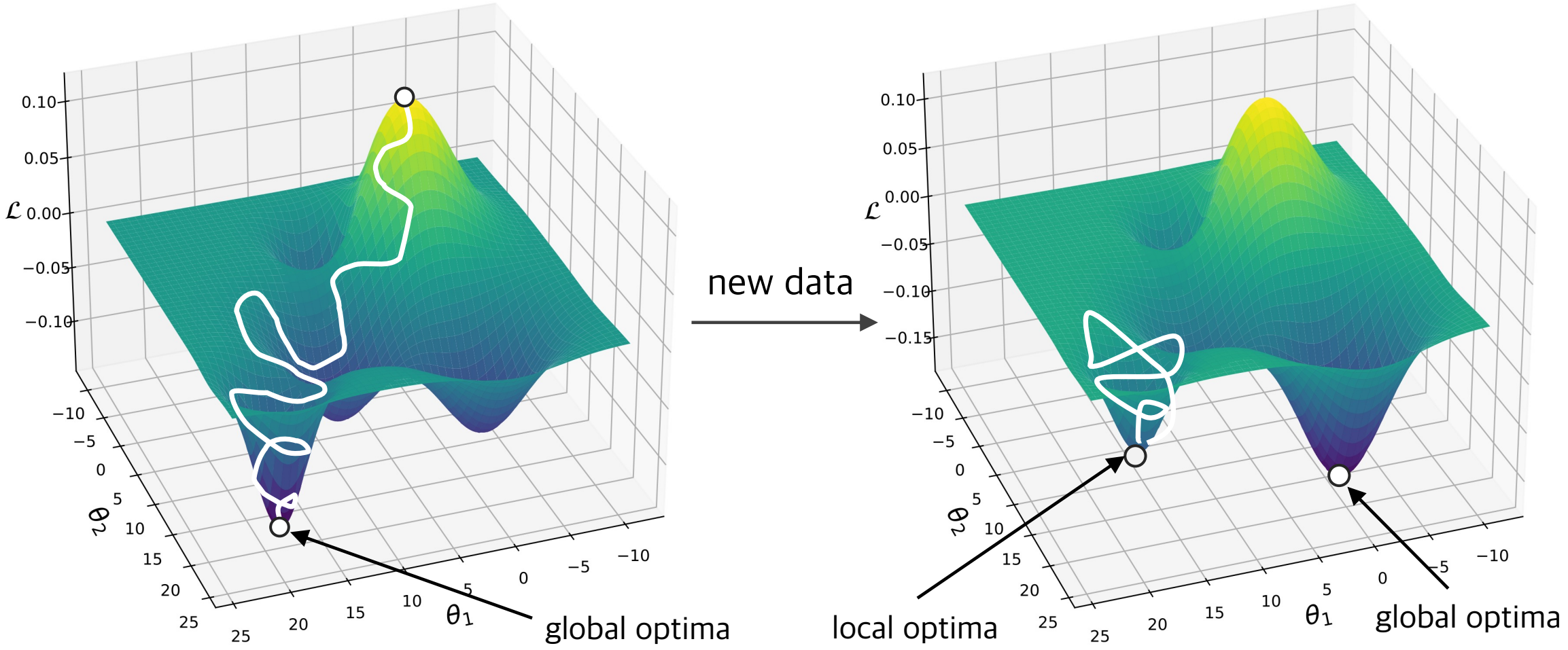
Introduction | Challenge of Non-Stationary Optimization



Early-Training Phase

Late-Training Phase

Introduction | Challenge of Non-Stationary Optimization



Early-Training Phase

Late-Training Phase

Introduction | Loss of Plasticity

Loss of Plasticity [1]

- In non-stationary optimization, neural networks loses its ability to adapt to new data.

[1] Loss of plasticity in deep continual learning, Dohare et al, Nature 2024.

Introduction | Loss of Plasticity

Loss of Plasticity [1]

- In non-stationary optimization, neural networks loses its ability to adapt to new data.

Scaling intensifies Loss of Plasticity

- Deeper networks create sharper loss surface and fall into local optima more easily.
- Scaling the number of updates makes model fall into local optima faster..

[1] Loss of plasticity in deep continual learning, Dohare et al, Nature 2024.

Introduction | Loss of Plasticity

Loss of Plasticity [1]

- In non-stationary optimization, neural networks loses its ability to adapt to new data.

Scaling intensifies Loss of Plasticity

- Deeper networks create sharper loss surface and fall into local optima more easily.
- Scaling the number of updates makes model fall into local optima faster..

Research Goal

- Maintain network plasticity to enable Scalable Deep RL.

[1] Loss of plasticity in deep continual learning, Dohare et al, Nature 2024.

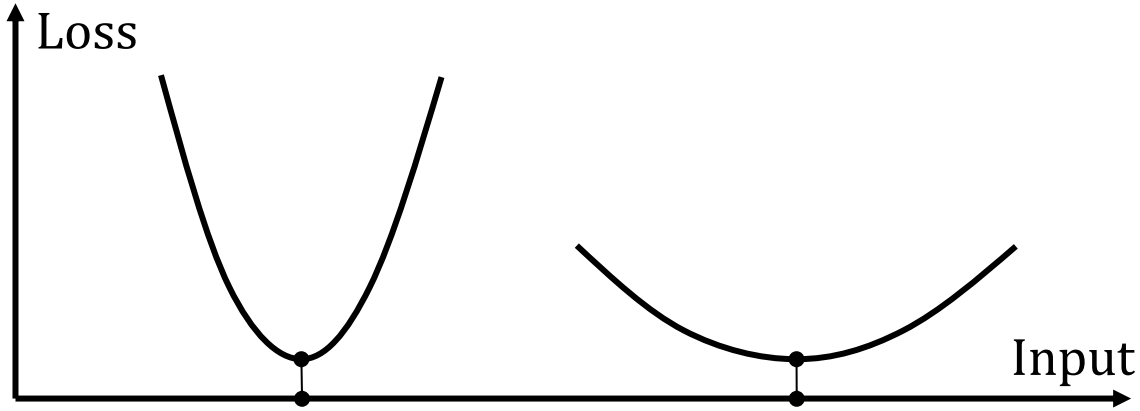
Outline

1. Short Bio
2. Introduction
   - Why do we need Scalable RL?
   - Why Scaling is Difficult in Deep RL?
   - Loss of Plasticity: A Key Bottleneck in Scalable Deep RL
3. Maintaining Plasticity for Scalable Deep RL
   - Existing Works on Preventing Plasticity Loss
   - PLASTIC: Smoothing Loss Landscape and Preserving Active Units for Scalable RL (NeurIPS'23)
   - Simba: Simplicity Bias for Scalable RL (ICLR'25 Spotlight)
   - SimbaV2: Hyperspherical Normalization for Scalable RL (ICML'25 Spotlight)
   - FlashSAC: Scaling Off-Policy RL for Speed (preprint)
4. Conclusion

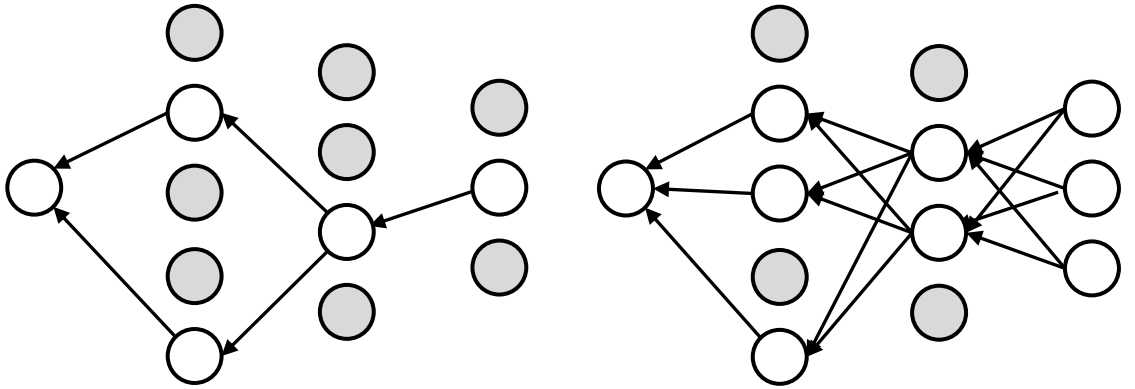
Why Do Neural Network Suffers from Plasticity Loss?

Why Do Neural Network Suffers from Plasticity Loss?

- **Sharp Local Optima**: Networks get trapped and struggle to escape.
- **Gradient Vanishing**: After prolonged training, gradients weaken and fail to reach upper layers.



Sharp vs Smooth Loss Surface



Ineffective vs Effective Gradient Propagation

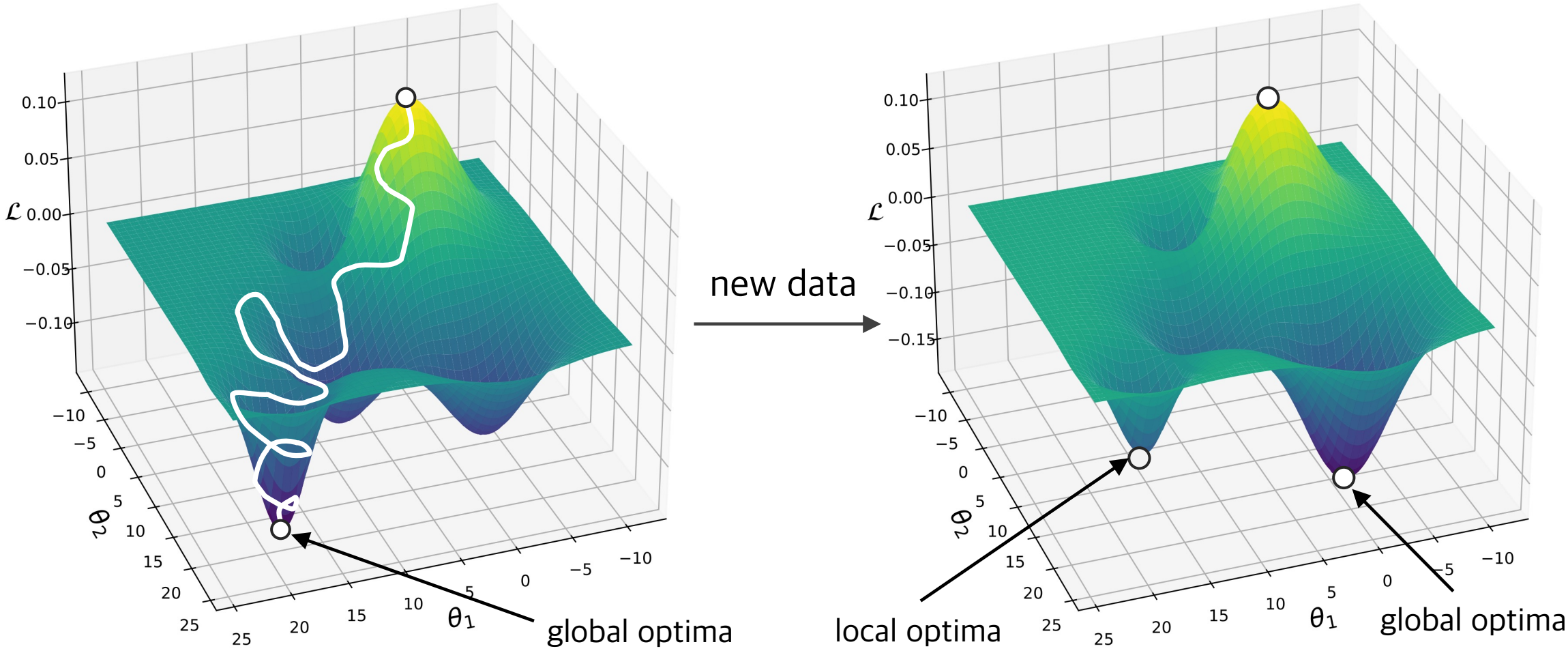
Existing Works on Preventing Plasticity Loss

Network Reinitialization

- Periodically reinitializes the network to restore plasticity.
- Effective, but not scalable.

Existing Works on Preventing Plasticity Loss

Network Reinitialization

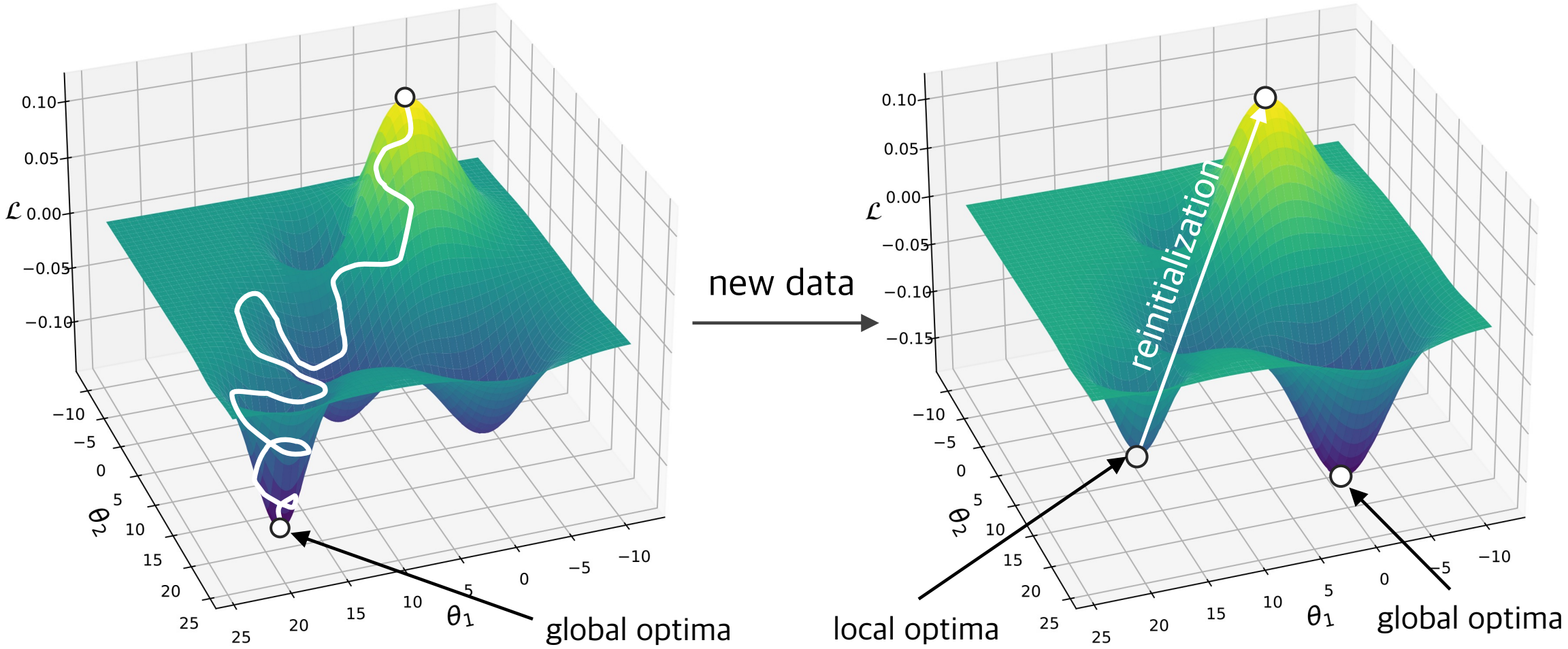


Early-Training Phase

Late-Training Phase

Existing Works on Preventing Plasticity Loss

Network Reinitialization

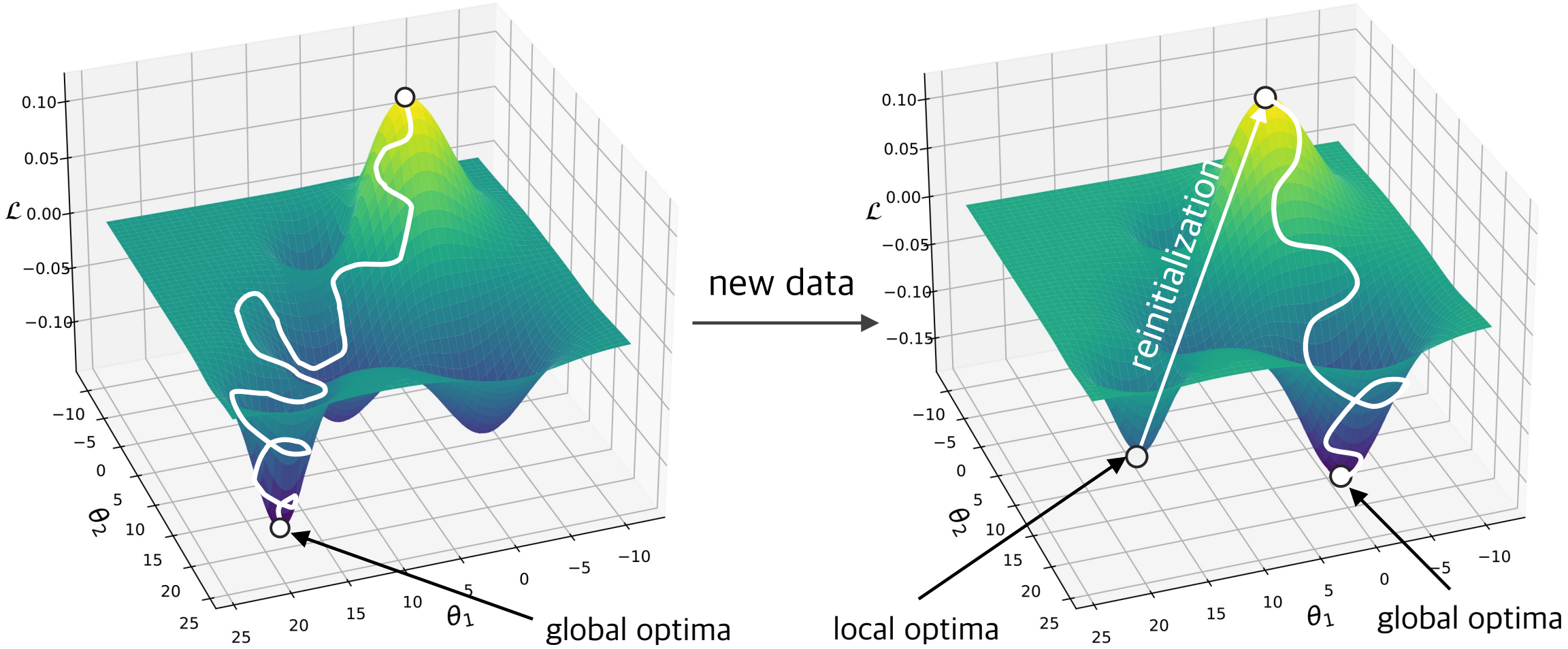


Early-Training Phase

Late-Training Phase

Existing Works on Preventing Plasticity Loss

Network Reinitialization



Early-Training Phase

Late-Training Phase

Existing Works on Preventing Plasticity Loss

Network Reinitialization

- Periodically reinitializes the network to restore plasticity.
- Effective, but not scalable (imagine reinitializing GPT-5).

Existing Works on Preventing Plasticity Loss

Network Reinitialization

- Periodically reinitializes the network to restore plasticity.
- Effective, but not scalable (imagine reinitializing GPT-5).

Avoiding Sharp Local Optima

- LayerNorm: Normalizes intermediate features to smooth the loss landscape.
- SAM Optimizer: Encourages convergence to smoother minima.
- Residual Connection, Data Augmentation, ….

Existing Works on Preventing Plasticity Loss

Network Reinitialization

- Periodically reinitializes the network to restore plasticity.
- Effective, but not scalable (imagine reinitializing GPT-5).

Avoiding Sharp Local Optima

- **LayerNorm**: Normalizes intermediate features to smooth the loss landscape.
- **SAM Optimizer**: Encourages convergence to smoother minima.
- **Residual Connection, Data Augmentation, ….**

Maintaining Gradient Propagation

- **CReLU activation**: Uses $\mathrm{CReLU}(x) = [\mathrm{ReLU}(x), \mathrm{ReLU}(-x)]$ to avoid dead units.
- **Parseval Regularization**: Constrain weight matrices to remain near-orthogonal.

Existing Works on Preventing Plasticity Loss

How to Measure a Network's Plasticity?

- Loss Sharpness
  - Condition Number: Ratio of the maximum to minimum eigenvalue of Hessian.
- Gradient Propagation
  - Dead / Dormant neurons: Number of neurons no longer active in forward or backward propagation.
  - Effective Learning Rate: Gradient Norm divided by Parameter Norm.

PLASTIC: Improving Input and Label Plasticity for Sample Efficient Reinforcement Learning

Hojoon Lee*, Hanseul Cho*, Hyunseung Kim*, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, Chulhee Yun

NeurIPS'23

KAIST AI
Graduate School of AI

PLASTIC | Maintaining Plasticity for Sample-Efficient RL

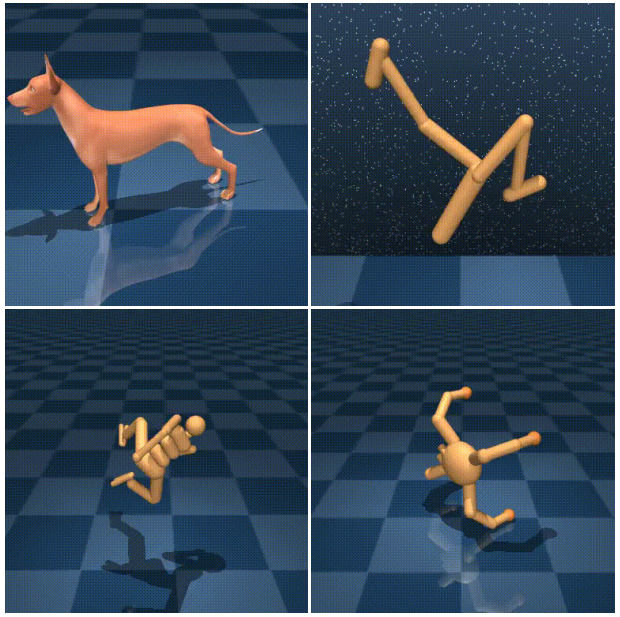
PLASTIC

- Why don’t we combine complementary methods to maintain plasticity?.
- PLASTIC = LayerNorm + SAM + CReLU + Periodic Reinitialization.

PLASTIC | Maintaining Plasticity for Sample-Efficient RL

Benchmark

- Deepmind Control Suite [1]
- Atari Games [2]



[1] DeepMind Control Suite, Tassa et al, arXiv 2018.

[2] The Arcade Learning Environment: An Evaluation Platform for General Agents, Bellmare et al, JAIR, 2013.

PLASTIC | Maintaining Plasticity for Sample-Efficient RL

Plasticity Analysis

- PLASTIC converges to smooth loss landscape with maintaining higher number of active units.

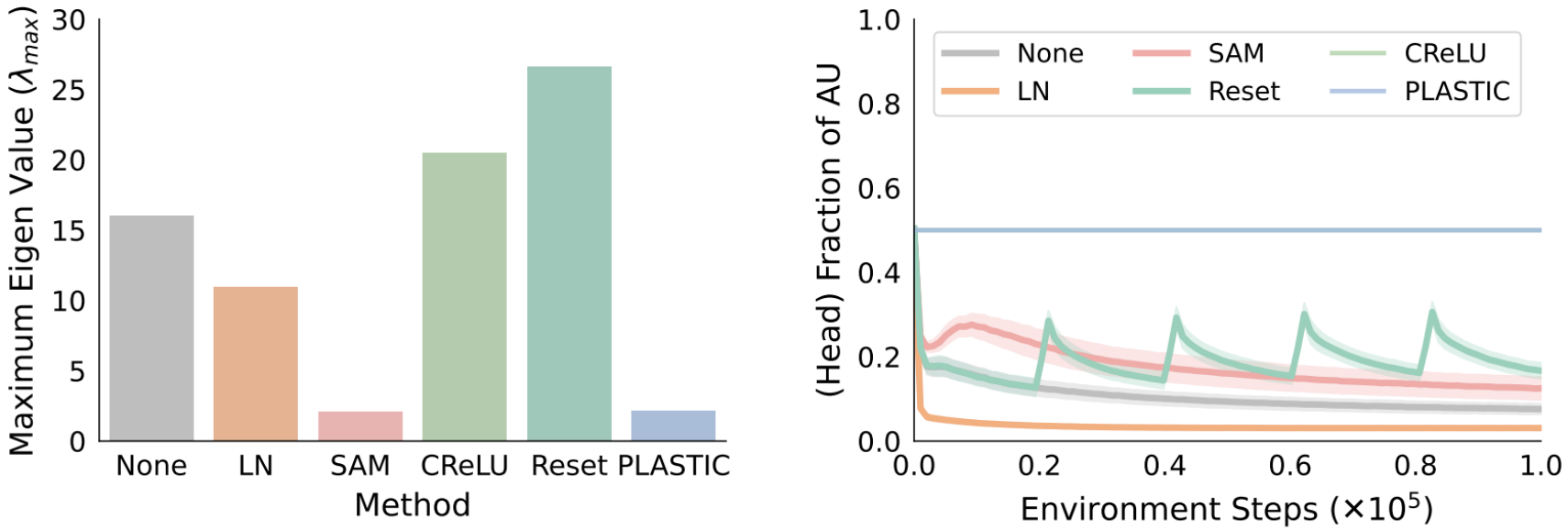


Figure: LayerNorm and the SAM optimizer lead to a smoother loss landscape, as measured by the maximum eigenvalue. CReLU and Reset achieve a higher fraction of active units, which is also restored through periodic head reinitialization.

PLASTIC | Maintaining Plasticity for Sample-Efficient RL

Benchmark Results

- PLASTIC outperforms any single method used in isolation.

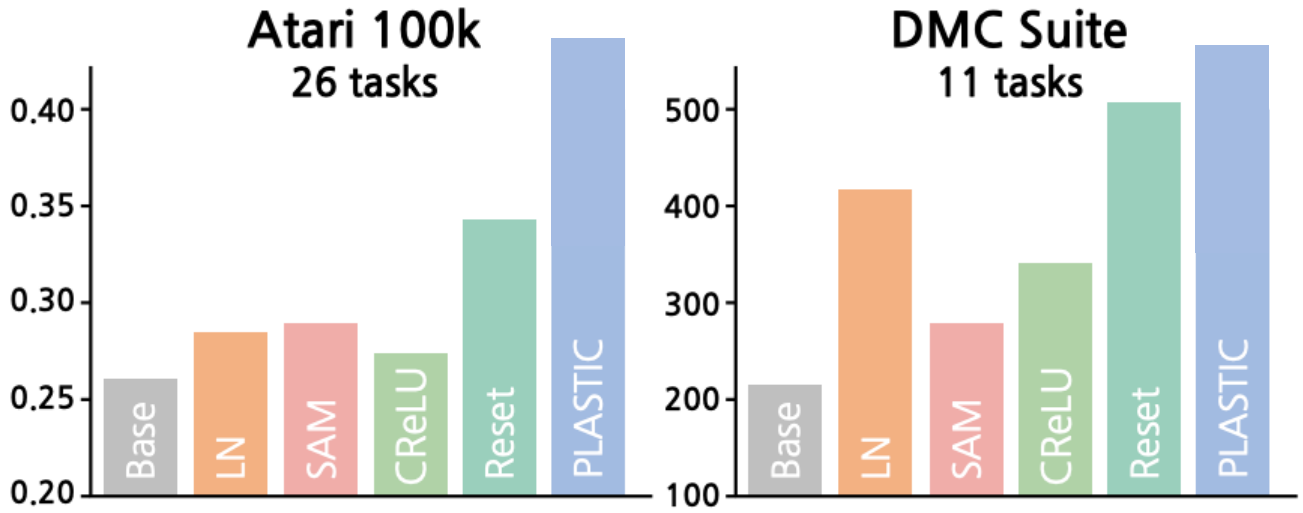


Figure: (Left) In both the Atari-100k and DMC-500k benchmarks, integrating different plasticity preserving methods yields substantially higher performance.

PLASTIC | Maintaining Plasticity for Sample-Efficient RL

Benchmark Results

- PLASTIC outperforms any single method used in isolation.

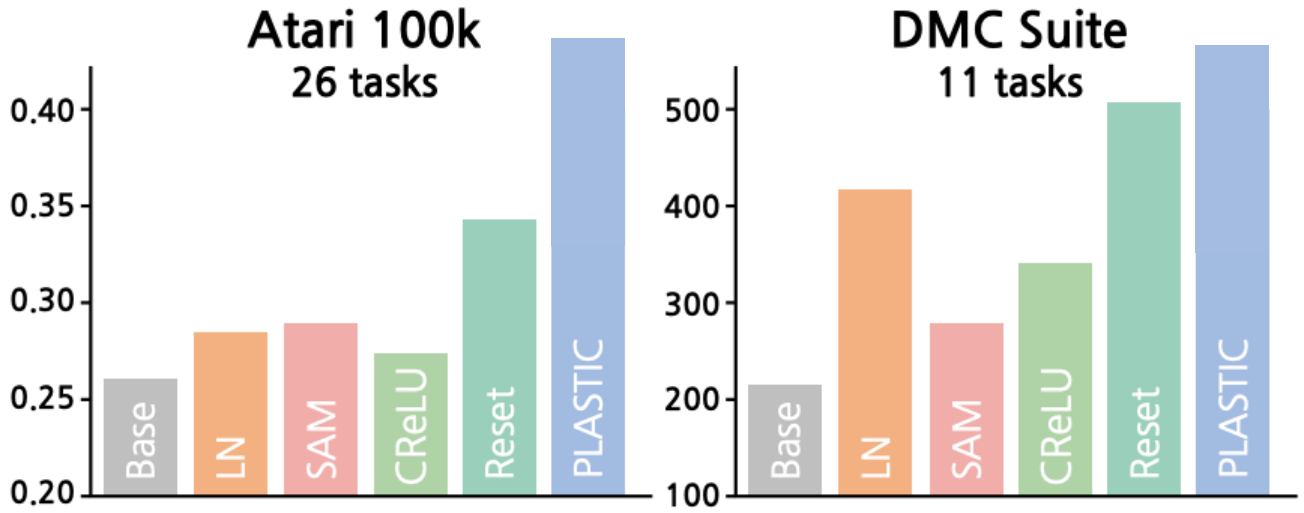


Figure: (Left) In both the Atari-100k and DMC-500k benchmarks, integrating different plasticity preserving methods yields substantially higher performance.

LN	SAM	CReLU	Reset	IQM
				0.258 (0.224, 0.292)
✓				0.259 (0.235, 0.293)
	✓			0.325 (0.296, 0.354)
		✓		0.256 (0.224, 0.287)
			✓	0.343 (0.314, 0.373)
✓	✓			0.341 (0.325, 0.366)
✓		✓		0.284 (0.257, 0.314)
✓			✓	0.396 (0.365, 0.430)
	✓	✓		0.372 (0.357, 0.408)
	✓		✓	0.411 (0.377, 0.447)
		✓	✓	0.373 (0.344, 0.402)
✓	✓	✓	✓	**0.421 (0.388, 0.457)**

Table: Although Combining various methods is effective, Periodic Reset remains essential.

Simba: Simplicity Bias for Scaling Up Parameters in Deep Reinforcement Learning

Hojoon Lee*, Dongyoon Hwang*, Donghu Kim, Hyunseung Kim, Jet Tai,
Kaushik Subramanian, Peter Wurman, Jaegul Choo, Peter Stone, Takuma Seno

ICLR'24 (spotlight)

KAIST AI Sony AI KRAFTON TEXAS

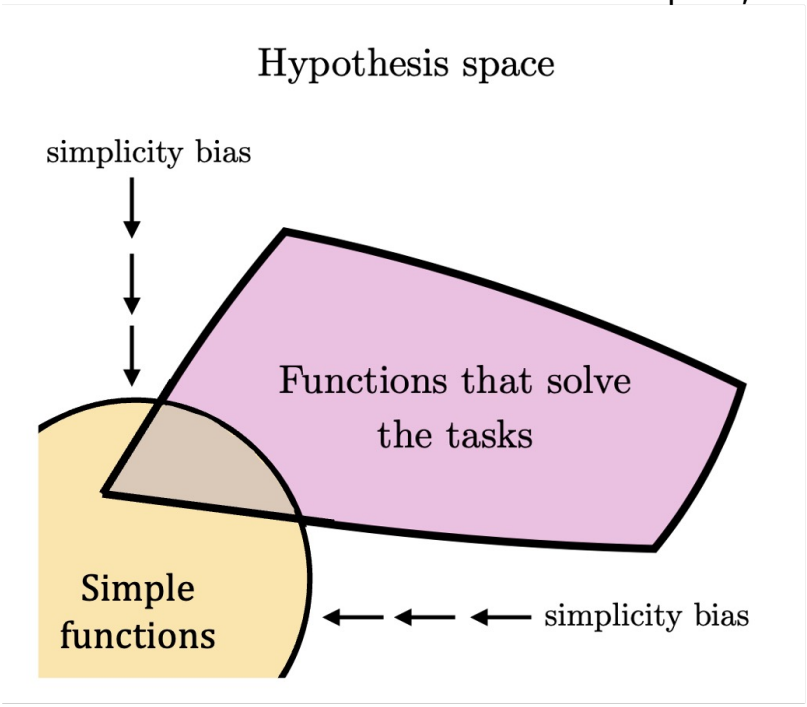
What Makes Supervised Learning Scalable?

Plasticity Loss can Still Occur in Supervised Learning

- Even with mild non-stationarity, large supervised models can still get fall into sharp local optima.

Simplicity Bias Hypothesis [1]

- Modern architectures favors simpler, smoother function, in line with Occam's Razor.



[1] The Platonic Representation Hypothesis, Huh et al, ICML 2024.

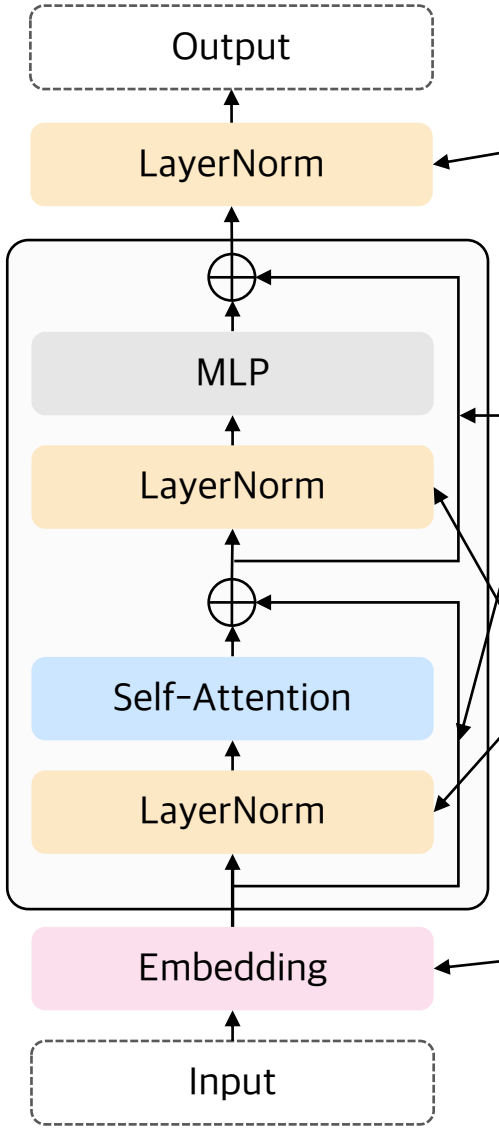
What Makes Supervised Learning Scalable?

Components that Contribute to Simplicity Bias

- Gradient Noise in Stochastic Gradient Descent [1].
- Weight Decay [2].
- Cross-Entropy Loss [3].
- ReLU-like activations [4].
- Small weights/activations [4].
- Batch/Layer/Instance Normalizations [4].
- Residual Connections [5].

[1] On the Generalization Benefit of Noise in Stochastic Gradient Descent, Smith et al, ICML 2020.
[2] SGD and Weight Decay Secretly Minimize the Rank of Your Neural Network, Galanti et al, arXiv 2024.
[3] Cross-Entropy Loss and Low Rank Features Have Responsibility for Adversarial Examples, Nar et al, arXv 2019.
[4] Neural Redshift: Random Networks are not Random Functions, Teney et al, CVPR 2024.
[5] Visualizing the Loss Landscape of Neural Nets, Li et al, NeurIPS 2018.

Demystifying Simplicity Bias in Transformer



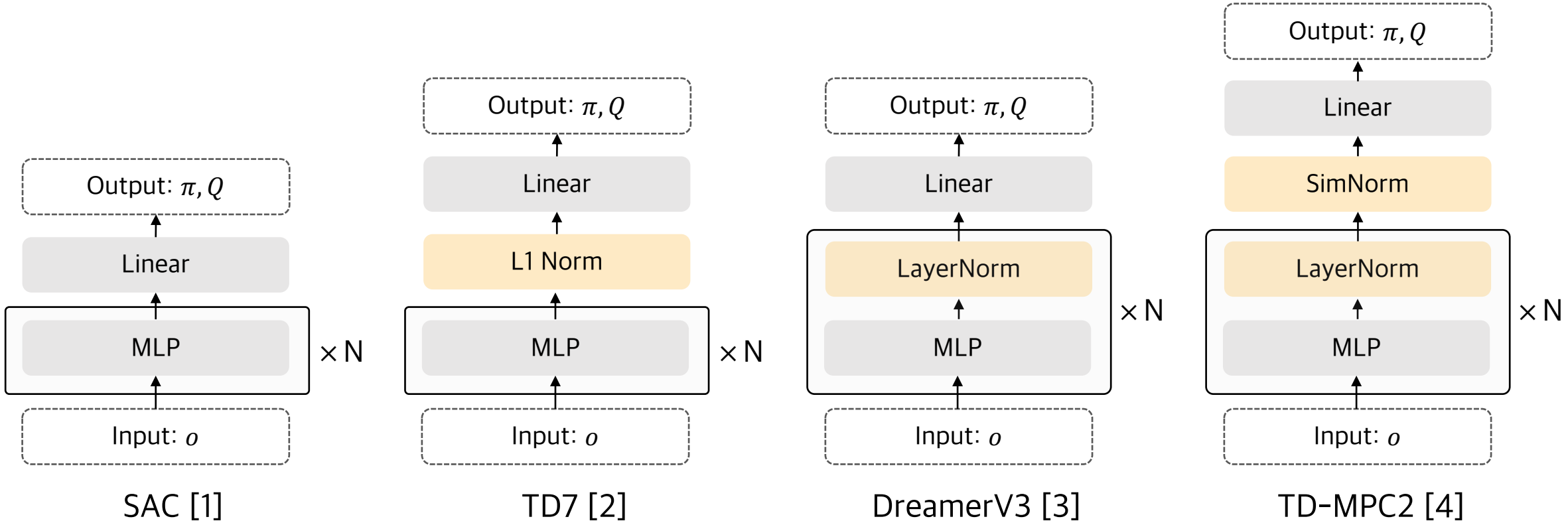
(4) **Post LayerNorm:** Normalizes aggregated features before the classifier.

(3) **Residual Connection:** Provide a direct linear pathway from input to output.

(2) **Pre LayerNorm:** Normalizes feature before applying non-linear transform.

(1) **Embedding:** Embeds inputs with small, scaled weights.

Standard Network Architectures in Deep RL



*World model components have been omitted for ease of understanding

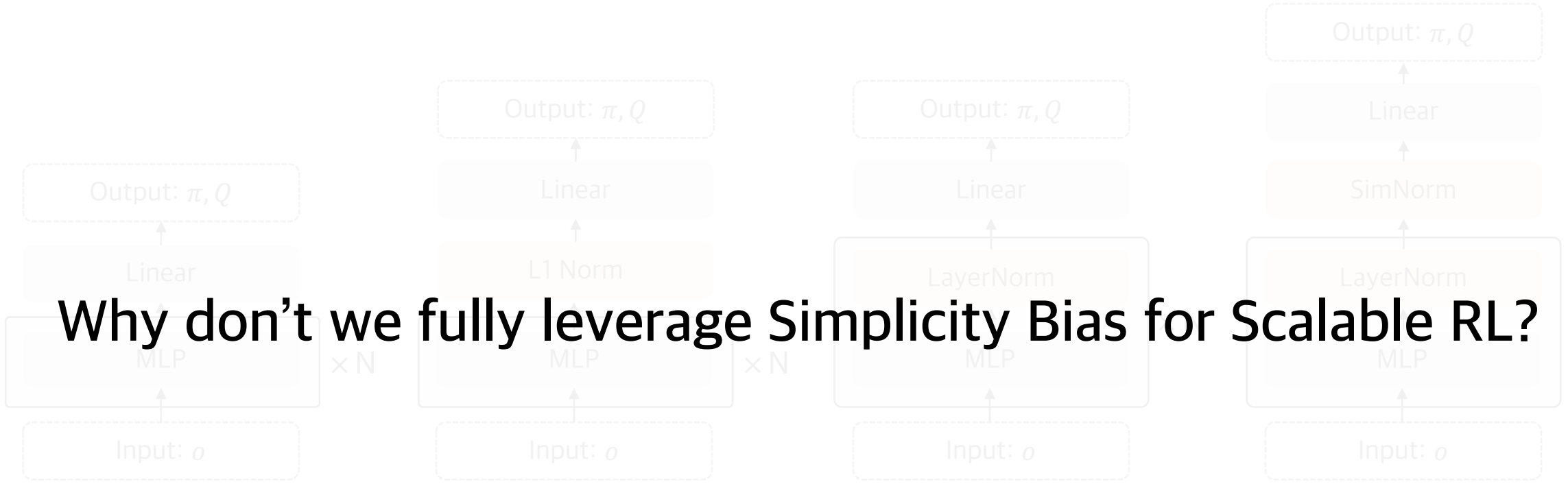
[1] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Haarnoja et al, ICML 2018.

[2] For SALE: State-Action Representation Learning for Deep Reinforcement Learning, Fujimoto et al, NeurIPS 2023.

[3] DreamerV3: Mastering Diverse Domains through World Models, Hafner et al, arXiv 2023.

[4] TD-MPC2: Scalable, Robust World Models for Continuous Control, Hansen et al, ICLR 2024.

Standard Network Architectures in Deep RL

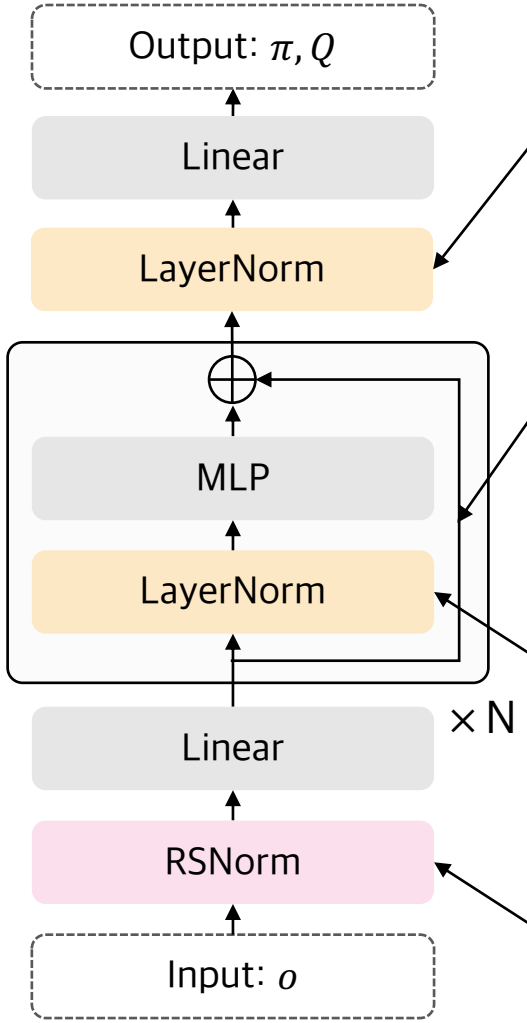


Why don’t we fully leverage Simplicity Bias for Scalable RL?

*World model components have been omitted for ease of understanding

[1] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Haarnoja et al, ICML 2018.
[2] For SALE: State-Action Representation Learning for Deep Reinforcement Learning, Fujimoto et al, NeurIPS 2023.
[3] DreamerV3: Mastering Diverse Domains through World Models, Hafner et al, arXiv 2023.
[4] TD-MPC2: Scalable, Robust World Models for Continuous Control, Hansen et al, ICLR 2024.

Simba | Simplicity Bias for Scalable Deep RL



(4) **Post LayerNorm**: Normalizes aggregated features before the classifier.

(3) **Residual Connection**: Provide a direct linear pathway from input to output.

(2) **Pre LayerNorm**: Normalizes feature before applying non-linear transform.

(1) **RSNorm**: Normalizes each input dimension to a normal distribution.

Simba | Simplicity Bias for Scalable Deep RL

Running Statistics Normalization (RSNorm)

Normalizes each input dimension to a normal distribution by **running statistics** of the input dimension.

Input: $o_t \in \mathbb{R}^d$, Running mean: $\mu_t \in \mathbb{R}^d$, Running variance: $\sigma_t \in \mathbb{R}^d$.

$$\delta_t \leftarrow o_t + \mu_{t-1}, \qquad \mu_t \leftarrow \mu_{t-1} + \frac{1}{t}\delta_t\,, \qquad o_t^2 \leftarrow \frac{t-1}{t}\left(\sigma_{t-1}^2 + \frac{1}{t}\delta_t^2\right)$$

$$\overline{o_t} = \mathrm{RSNorm}(o_t) = \frac{\mathrm{o_t} - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}}$$

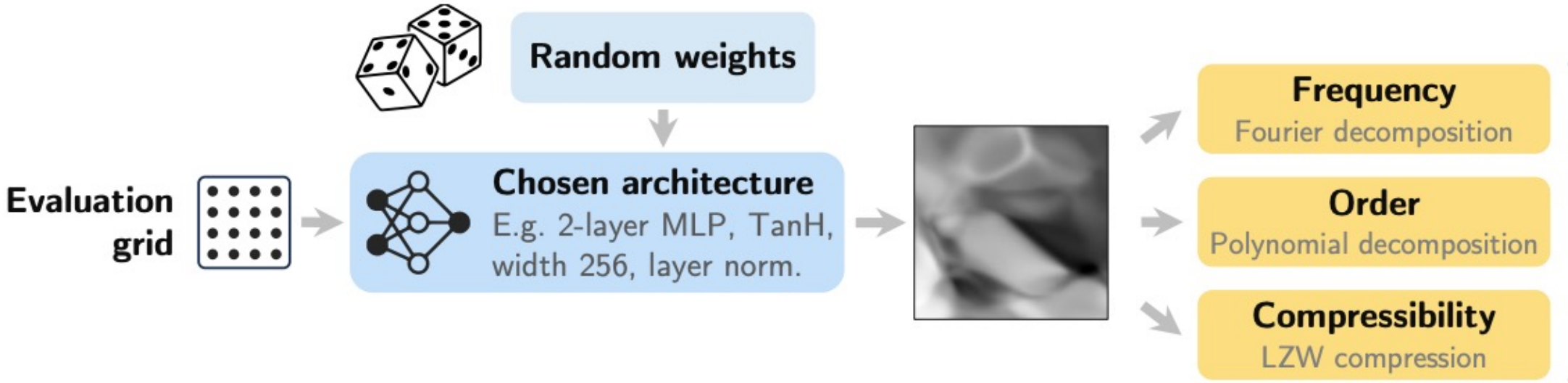
Why RSNorm?

- **LayerNorm**: Struggles with **disproportionately large values** in certain dimensions.
- **BatchNorm**: Unstable updates due to **non-stationary mini-batch statistics.**

What Makes Supervised Learning Scalable?

Measuring Simplicity Bias [1]

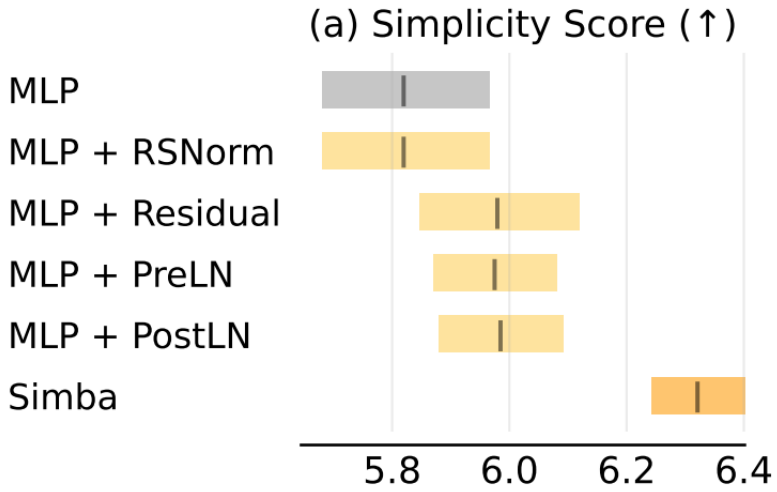
- Generate N, d-dimensional grid inputs.
- Forward them through the network to obtain N, d-dimensional grid outputs.
- Analyze output complexity (e.g., lower polynomial outputs indicate higher simplicity bias).



[1] Neural Redshift: Random Networks are not Random Functions, Teney et al, CVPR 2024.

Simba | Simplicity Bias for Scalable Deep RL

Simplicity Bias Analysis

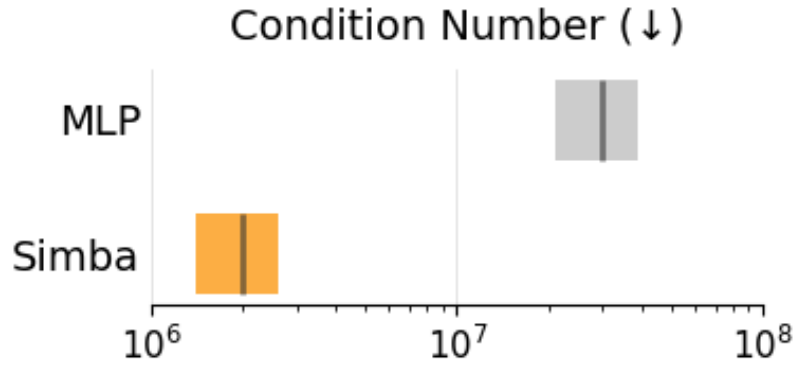


Adding each component increases the Simplicity Bias.

*RSNorm is the only exception.

*Likely because we used a uniformly distributed dataset for this analysis.

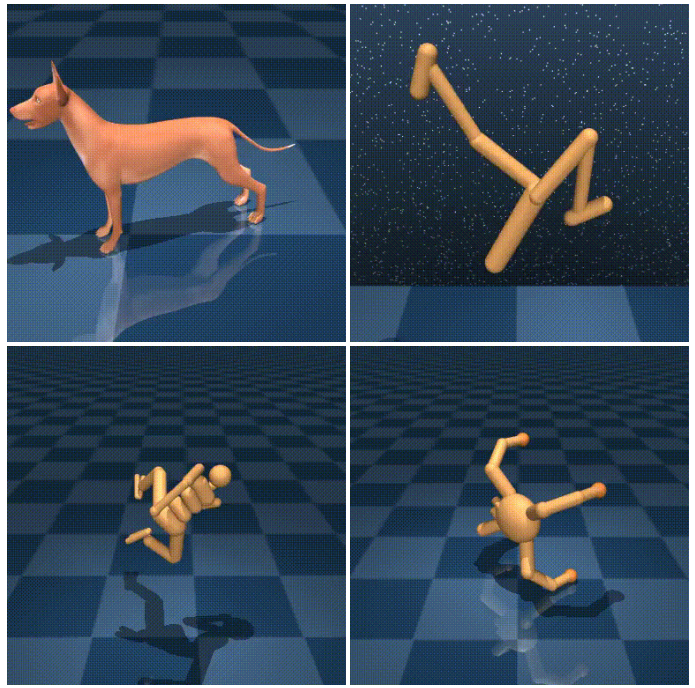
Plasticity Analysis



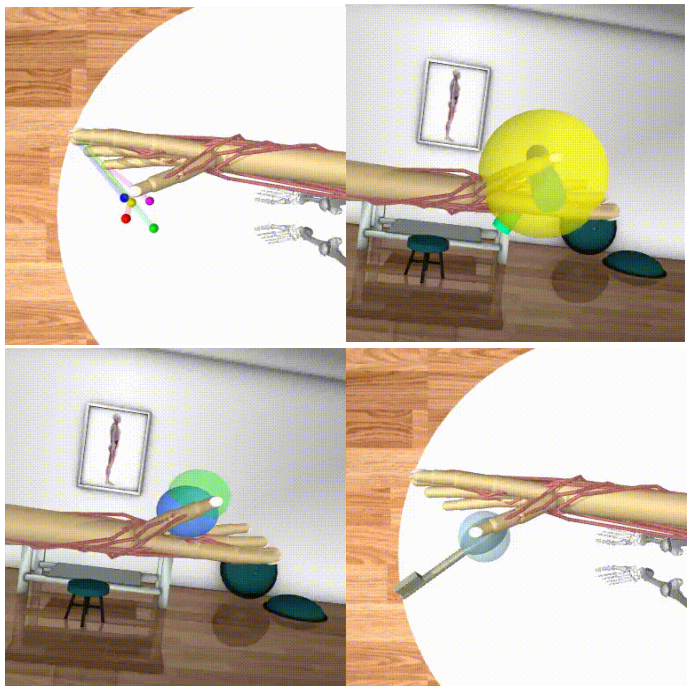
Simba induces much smoother loss landscape.

Simba | Simplicity Bias for Scalable Deep RL

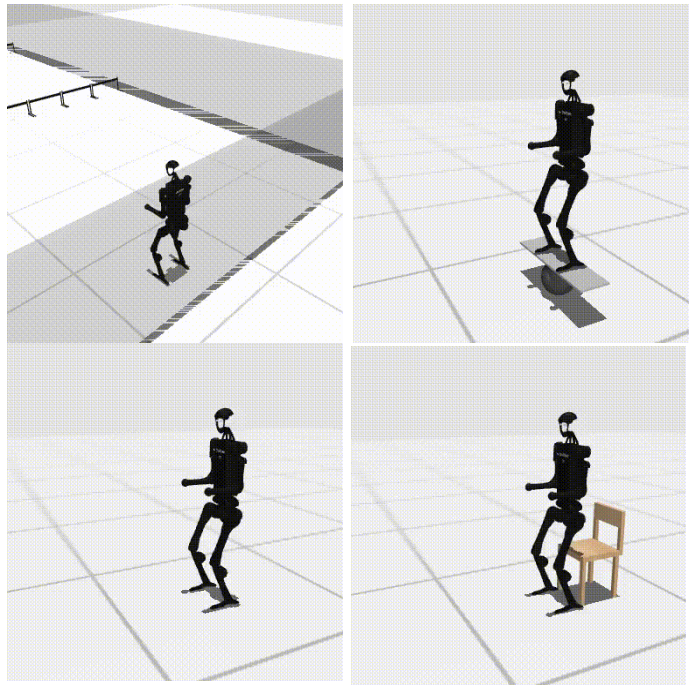
Benchmark



DMC [1]



MyoSuite [2]



Humanoid Bench [3]

[1] DeepMind Control Suite, Tassa et al, arXiv 2018.

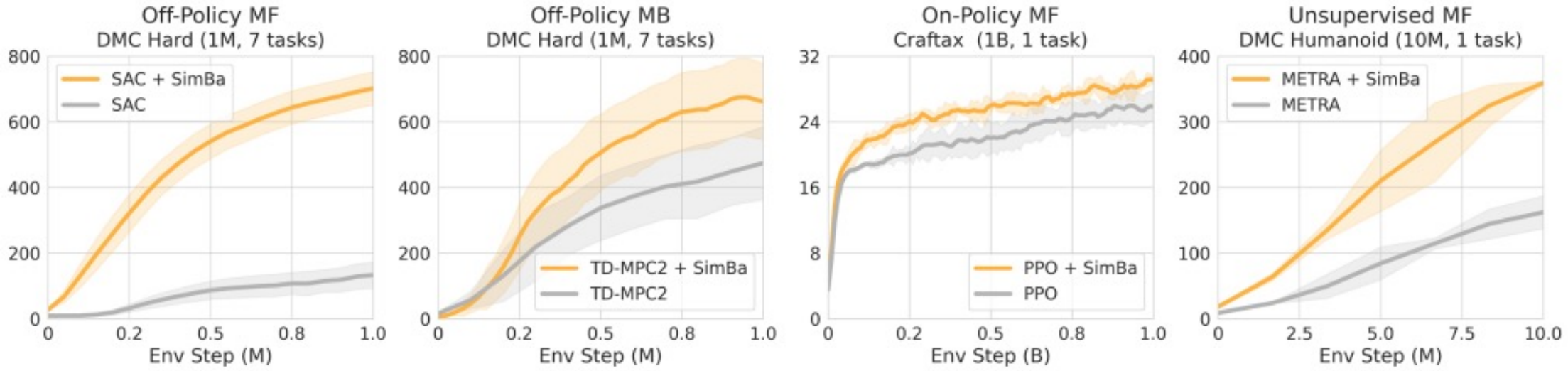
[2] MyoSuite -- A contact-rich simulation suite for musculoskeletal motor control, Caggiano et al, ICML 2022.

[3] HumanoidBench: Simulated Humanoid Benchmark for Whole-Body Locomotion and Manipulation, Sferazza et al, RSS 2024.

Simba | Simplicity Bias for Scalable Deep RL

Benchmark Results

- Simply replacing the architecture from MLP to Simba yields huge improvements.



SAC [1] | TD-MPC2 [2] | PPO [3] | METRA [4]

[1] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Haarnoja et al, ICML 2018.

[2] TD-MPC2: Scalable, Robust World Models for Continuous Control, Hansen et al, ICLR 2024.

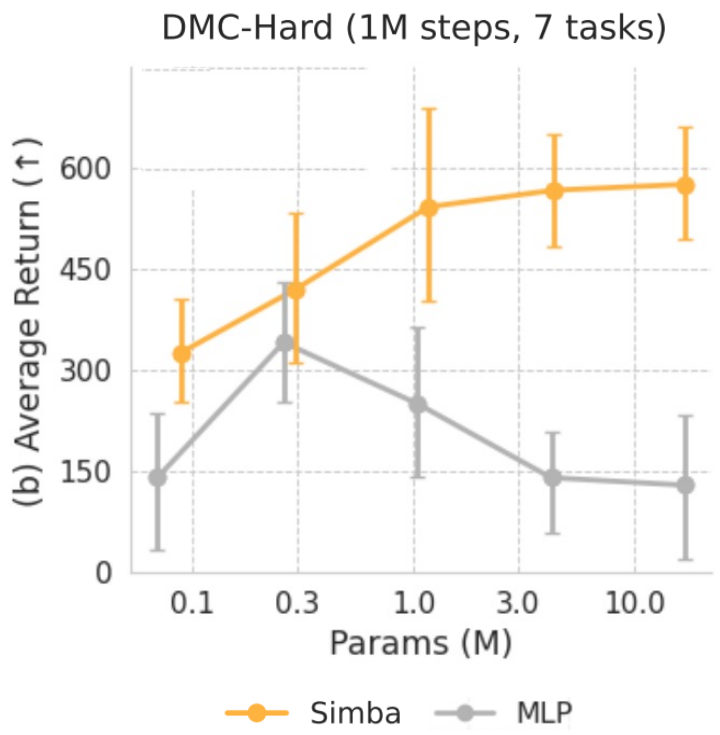
[3] Proximal Policy Optimization Algorithms for Continuous Control, Schulman et al, arXiv 2017.

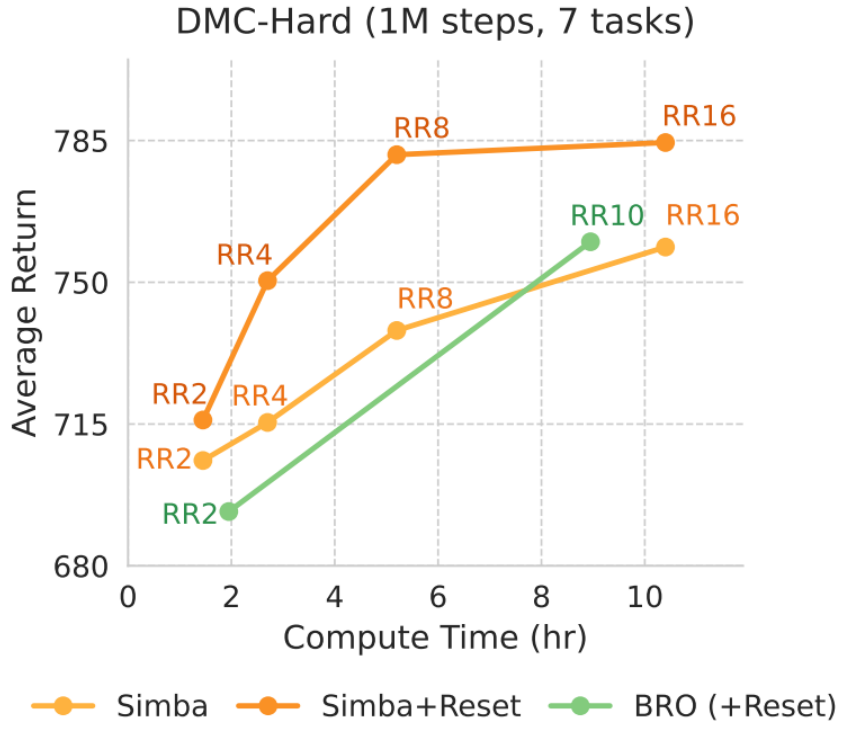
[4] METRA: Scalable Unsupervised RL with Metric-Aware Abstraction, Park et al, ICLR 2024.

Simba | Simplicity Bias for Scalable Deep RL

Scaling Results

- Simba enables scaling parameters and computes in Deep RL.
- However, periodic reinitialization achieves better scaling.



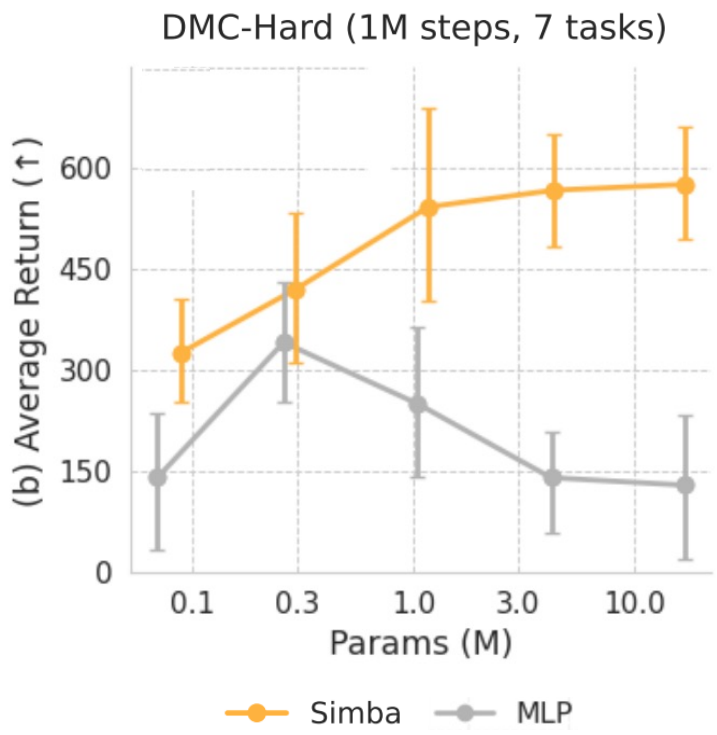


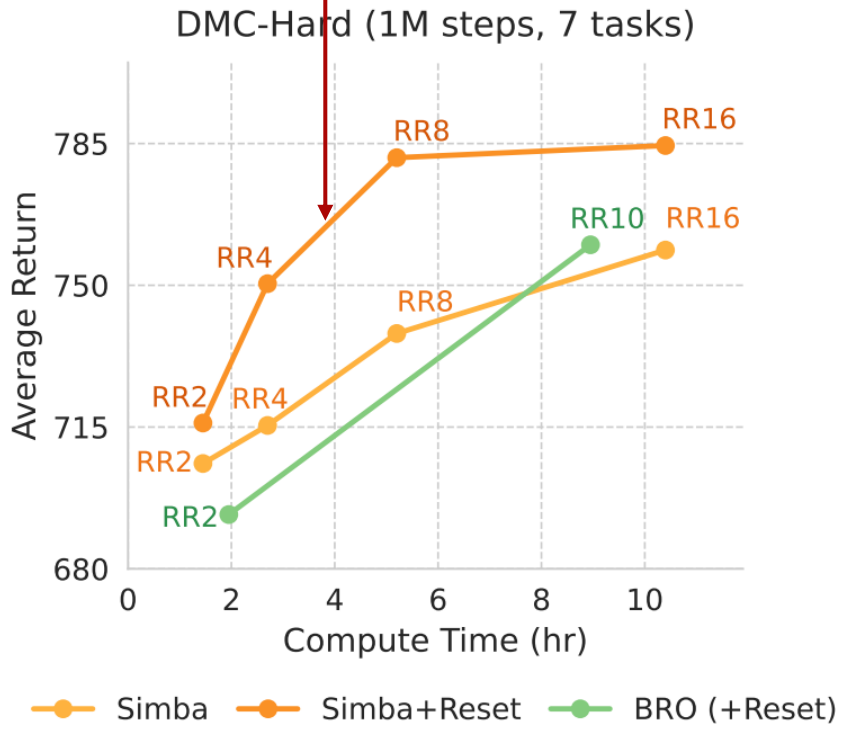
[1] BRO: Bigger, Regularized, Optimistic: scaling for compute and sample-efficient continuous control, Nauman et al, NeurIPS 2024.

Simba | Simplicity Bias for Scalable Deep RL

Scaling Results

- Simba enables scaling parameters and computes in Deep RL.
- However, periodic reinitialization achieves better scaling.





[1] BRO: Bigger, Regularized, Optimistic: scaling for compute and sample-efficient continuous control, Nauman et al, NeurIPS 2024.

SimbaV2: Hyperspherical Normalization for Scalable Deep Reinforcement Learning

Hojoon Lee*, Youngdo Lee*, Takuma Seno, Donghu Kim, Peter Stone, Jaegul Choo

ICML'25 (Spotlight)

KAIST AI Graduate School of AI
Sony AI
TEXAS The University of Texas at Austin

Beyond Simba

Is Simba sufficient to maintain plasticity?

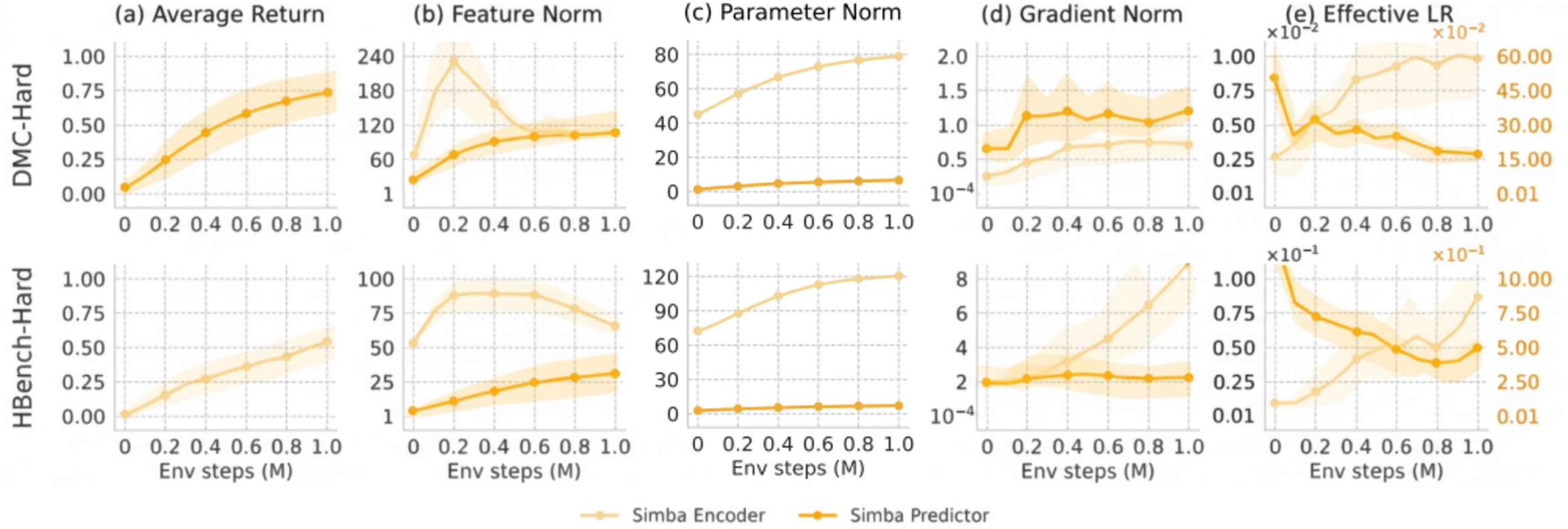


Figure: Training curves of key metrics for Simba with the SAC algorithm on DMC-Hard and Hbench-Hard tasks.

Feature, parameter, and gradient norms fluctuate throughout training, as the effective learning rates (ELR).

Beyond Simba

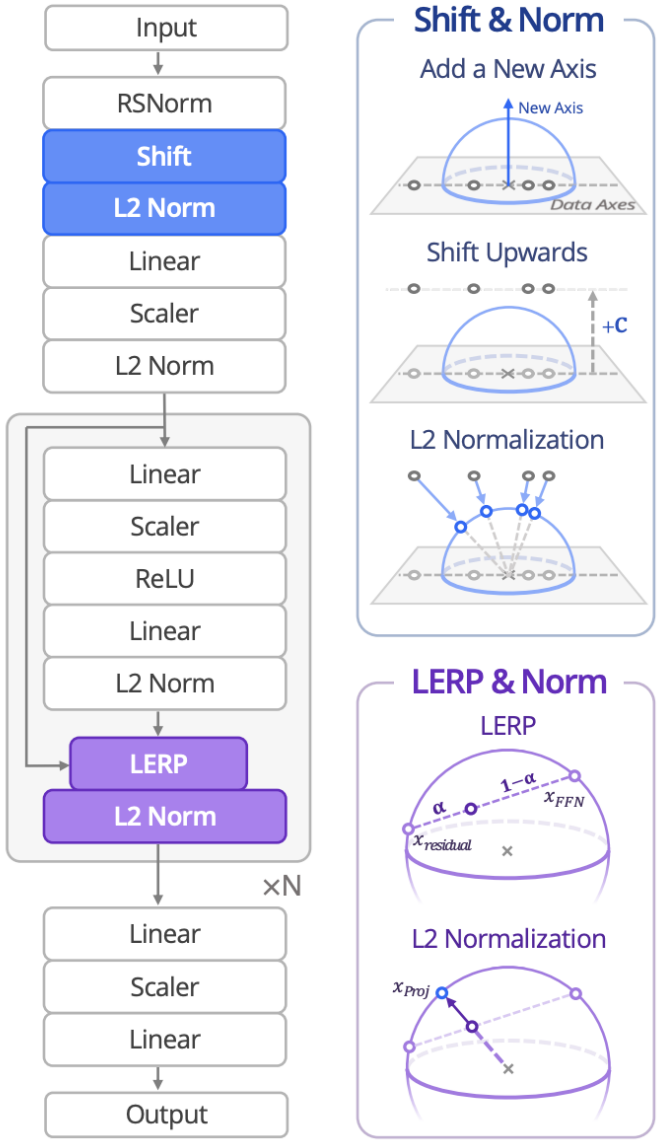
How to Enforce a Smoother Loss Landscape?

- Layernorm allows unbound growth of feature norm.
- Use stricter normalization to bound the feature norm.

How to Maintain Effective Gradient Propagation (Effective LR)?

- Stabilize parameter norms.
- Stabilize feature norms.
- Stabilize gradient norms.

SimbaV2 | Simba → SimbaV2



- Design Principle
  - Smooth Loss Landscape and maintain an Effective LR.
  - Constrain Feature, Parameter, Gradient to Unit Norms.
- Feature Norm
  - Layer Normalization → L2 Normalization.
  - Residual Connection→ Learnable Linear Interpolation (LERP).
- Parameter Norm
  - Parameters are projected onto the unit-hypersphere per update.
  - Linear → Linear (weights on unit hypersphere) + Scaler.
- Gradient Norm
  - MSE Loss → KL-Divergence Loss.
  - Input Scaling: RSNorm → RSNorm.
  - Reward Scaling: None → RSNorm.

SimbaV2 | Hyperspherical Normalization for Scalable Deep RL

Plasticity Analysis

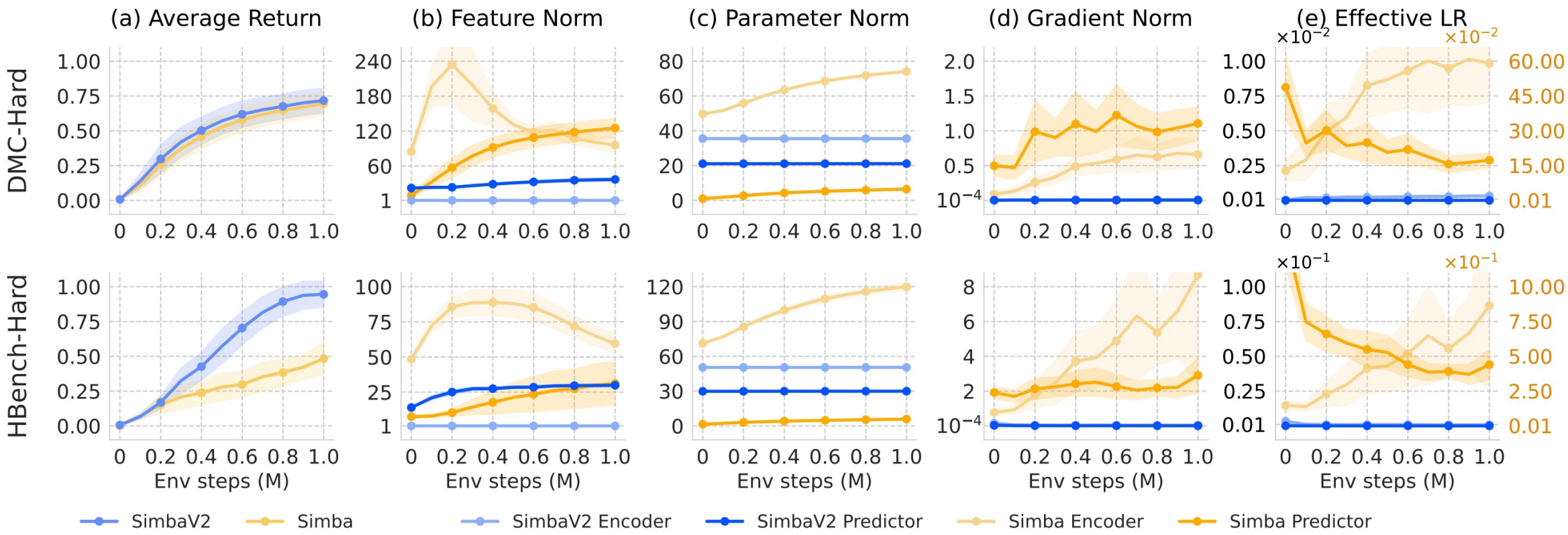
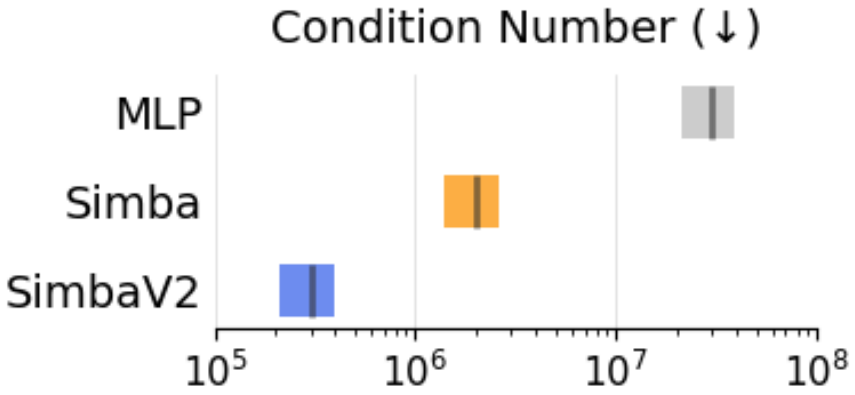


Figure: Training curves of key metrics for Simba and SimbaV2 with the SAC algorithm on DMC-Hard and Hbench-Hard tasks. In SimbaV2, feature, parameter, and gradient norms and ELR remain highly stable throughout training.

SimbaV2 | Hyperspherical Normalization for Scalable Deep RL

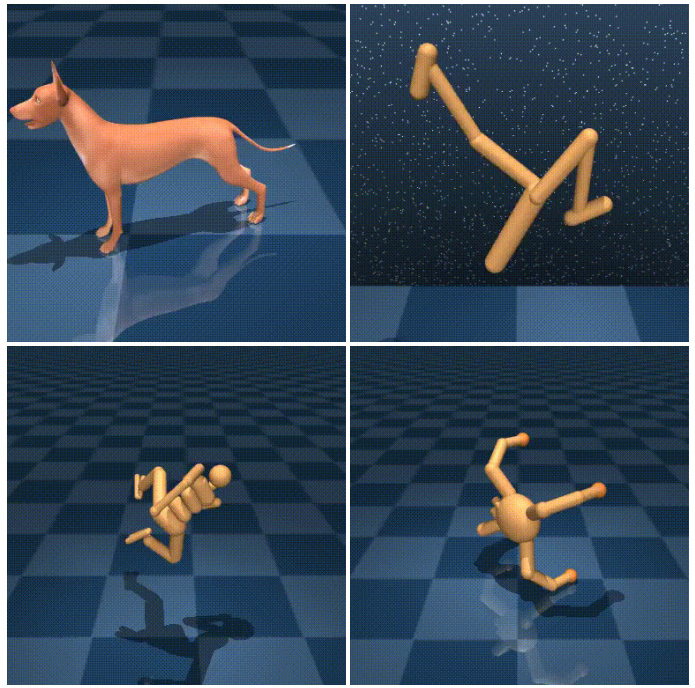
Plasticity Analysis



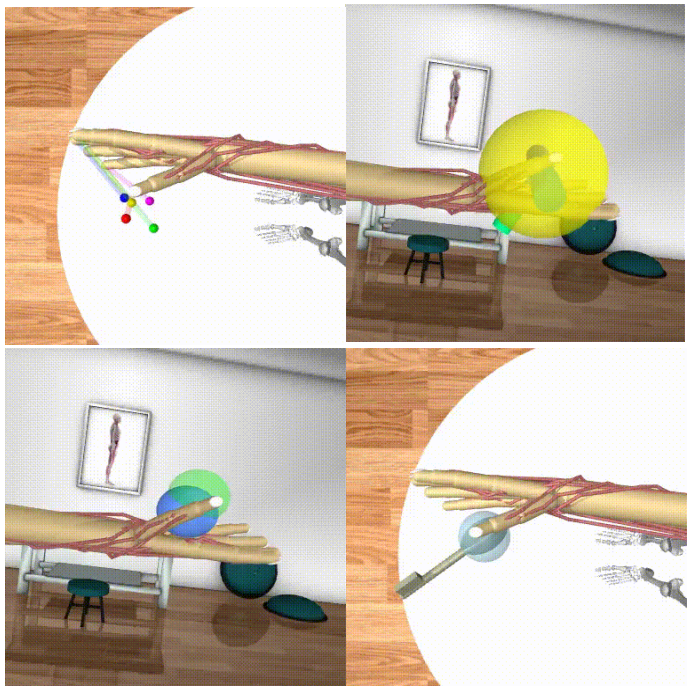
SimbaV2 induces much smoother loss landscape.

SimbaV2 | Hyperspherical Normalization for Scalable Deep RL

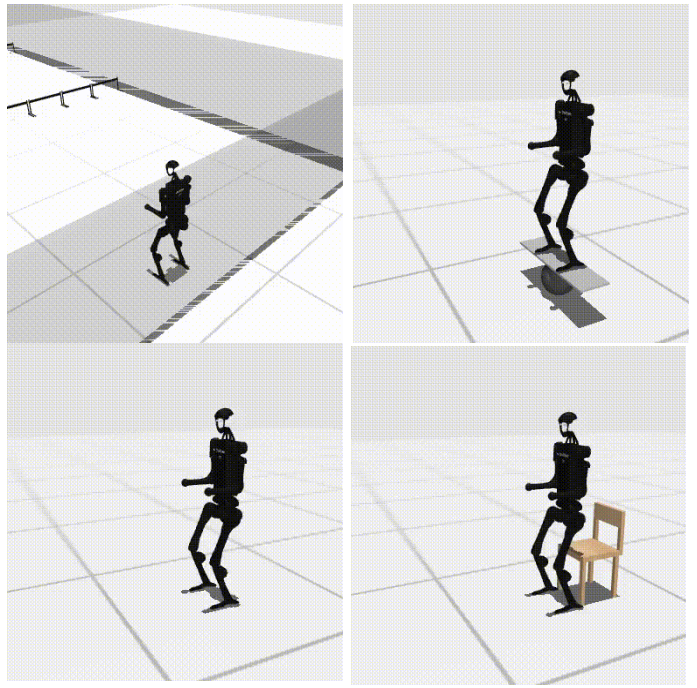
Benchmark



DMC [1]



MyoSuite [2]



Humanoid Bench [3]

[1] DeepMind Control Suite, Tassa et al, arXiv 2018.

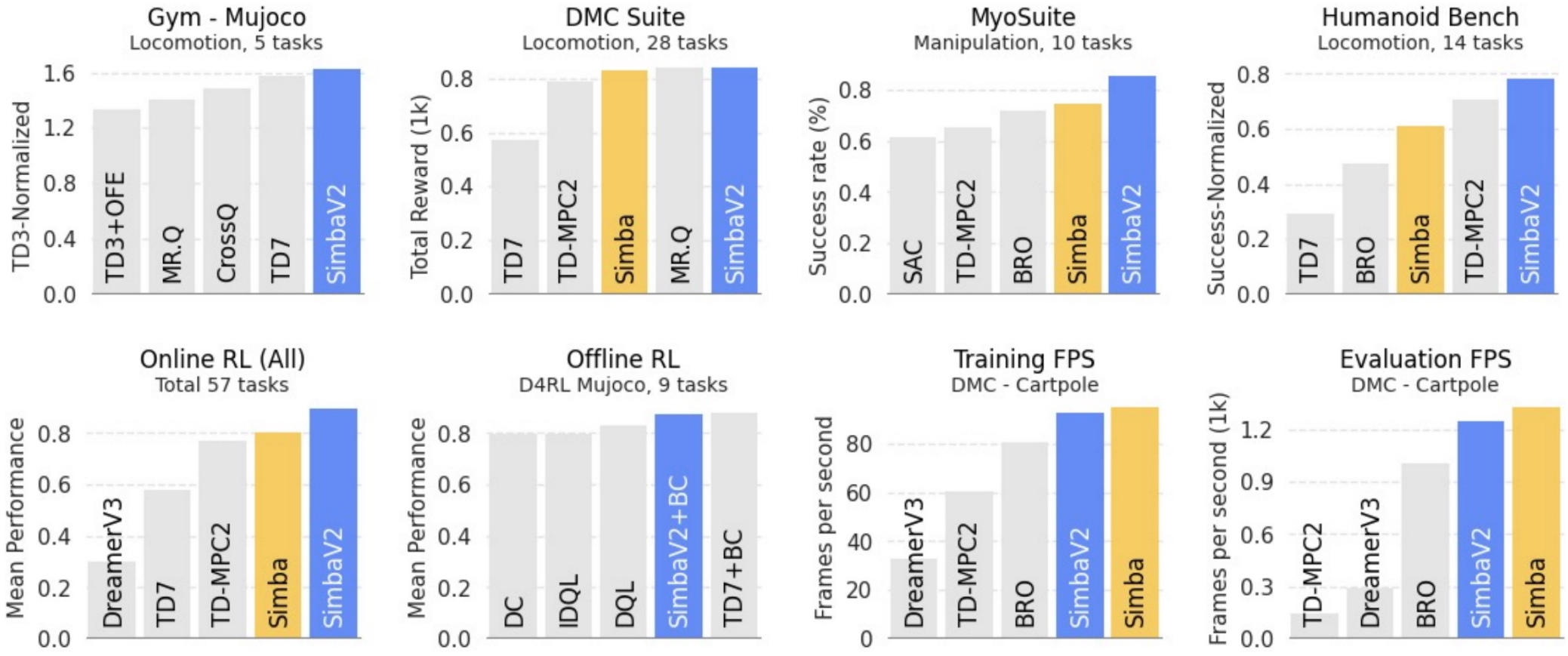
[2] MyoSuite -- A contact-rich simulation suite for musculoskeletal motor control, Caggiano et al, ICML 2022.

[3] HumanoidBench: Simulated Humanoid Benchmark for Whole-Body Locomotion and Manipulation, Sferazza et al, RSS 2024.

SimbaV2 | Hyperspherical Normalization for Scalable Deep RL

Benchmark Results

- Simply using SimbaV2 with the SAC algorithm achieves SOTA performance on various benchmarks.



SimbaV2 | Hyperspherical Normalization for Scalable Deep RL

Scaling Results

- SimbaV2 allows parameters and compute scaling without periodic reinitialization.

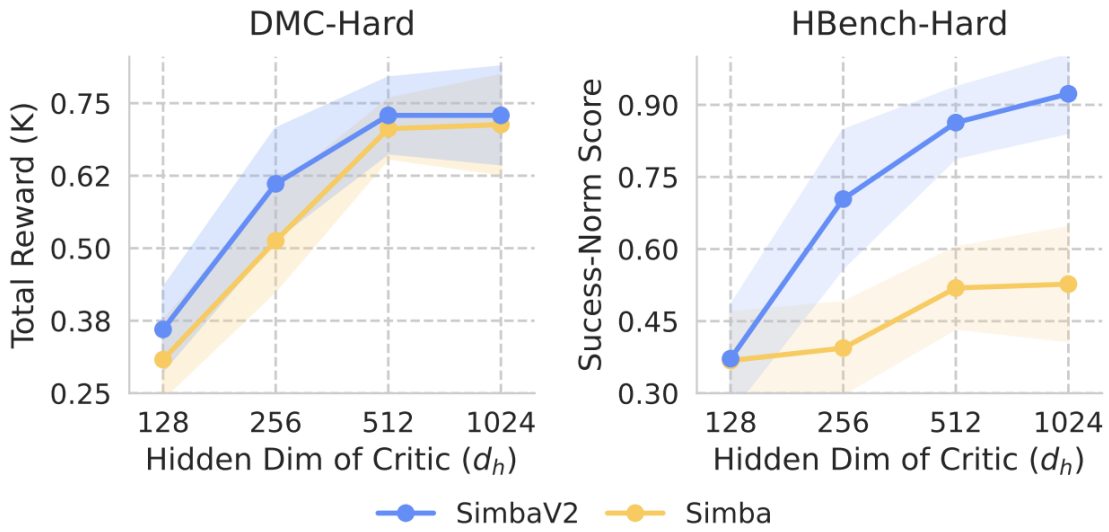


Figure 5. **Width Scaling.** We scale the number of model parameters by increasing the width of the critic network. On DMC-Hard, both Simba and SimbaV2 benefit from increased model size. On HBench-Hard, however, Simba plateaus at larger model sizes, whereas SimbaV2 continues to improve.

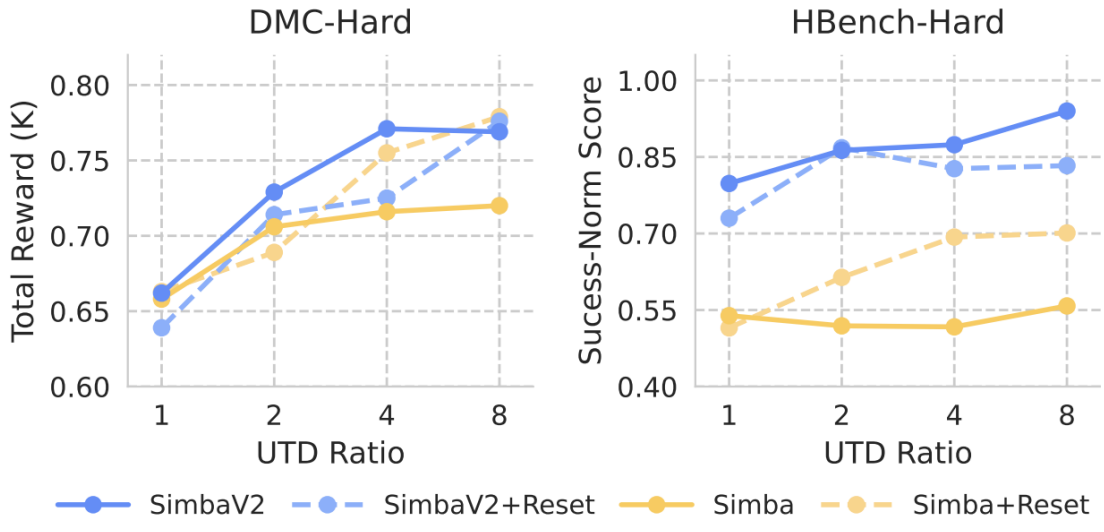


Figure 7. **Compute Scaling.**. We scale compute by increasing the UTD ratio. We compare Simba and SimbaV2, both with and without periodic reset. Simba saturates at lower ratios without reset, but improves with reset. In contrast, SimbaV2 scales smoothly even without reset, where using reset slightly degrades its performance.

FlashSAC: Scaling Off-Policy RL for Speed

Hojoon Lee*, Donghu Kim*, Youngdo Lee*, Takuma Seno, Ajay Subramanian,
Younggyo Seo, Jaegul Choo, Pieter Abbeel

In Progress

KAIST AI Graduate School of AI · Berkeley UNIVERSITY OF CALIFORNIA · NYU · TURING · KRAFTON · amazon

FlashSAC | Scaling Off-Policy RL for Speed

Towards Better Generalization and Compute-Efficiency

- SimbaV2 enables scaling parameters and compute in RL.
- Can we further achieve compute-efficiency through scalable RL?

FlashSAC | Scaling Off-Policy RL for Speed

Towards Better Generalization and Compute-Efficiency

- SimbaV2 enables scaling parameters and compute in RL.
- Can we further achieve compute-efficiency through scalable RL?

Lessons from Supervised Learning

- Under a fixed compute budget C, larger models, larger batch size, small updates lead to faster convergence.

FlashSAC | Scaling Off-Policy RL for Speed

Towards Better Generalization and Compute-Efficiency

- SimbaV2 enables scaling parameters and compute in RL.
- Can we further achieve compute-efficiency through scalable RL?

Lessons from Supervised Learning

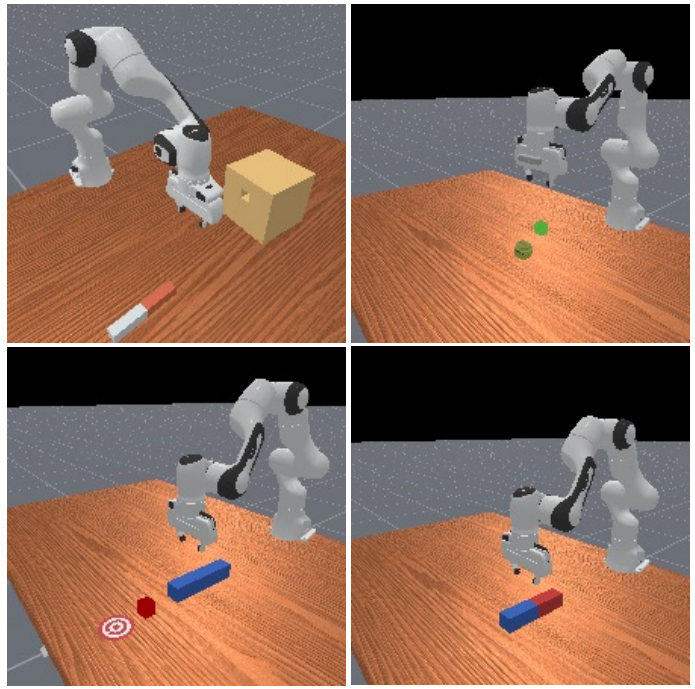
- Under a fixed compute budget C, larger models, larger batch size, small updates lead to faster convergence.

SAC → FlashSAC

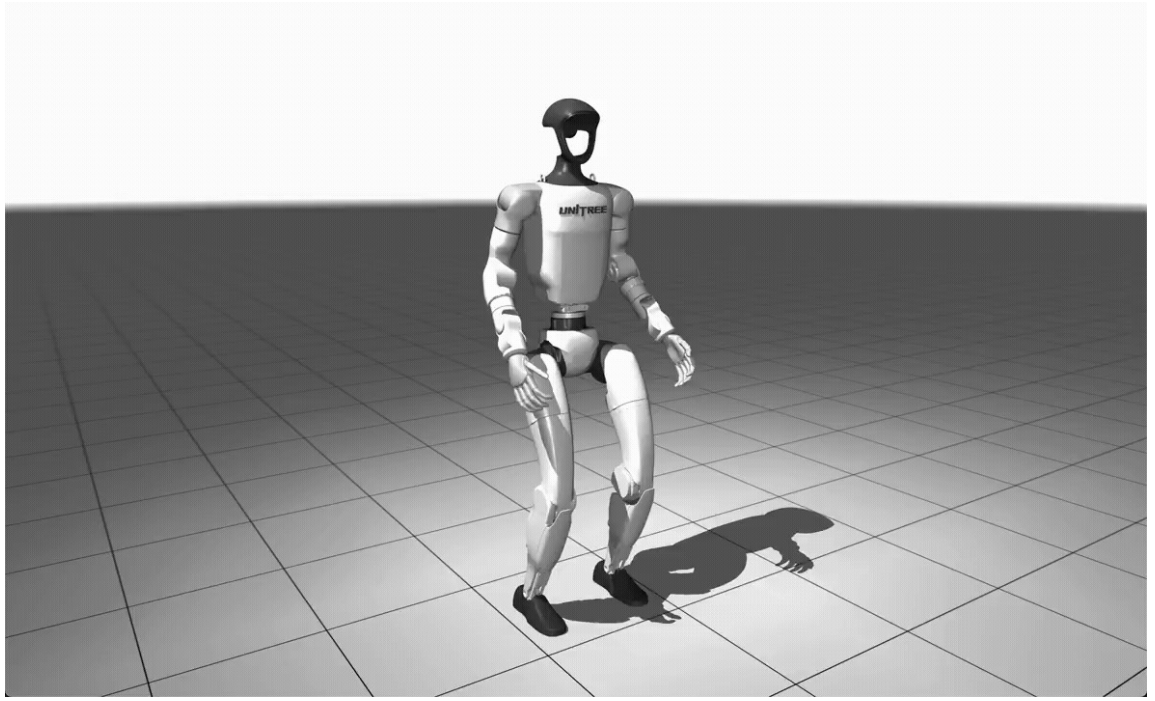
- Architecture: MLP → SimbaV2.
- Parameters: 0.5M → 5M.
- Learning Rate: 1e-4 → 3e-4.
- Batch-Size: 256 → 8192.
- Updates-to-Data ratio: 1.0 → 0.001.

FlashSAC | Scaling Off-Policy RL for Speed

Benchmark (w/ Highly-Parallelizable Simulators)



ManiSkill [1]



MujocoPlayground Humanoid Locomotion [2]

[1] ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI, Tao et al, RSS 2025.
[2] Mujoco Playground, Zakka et al, RSS 2025.

FlashSAC | Scaling Off-Policy RL for Speed

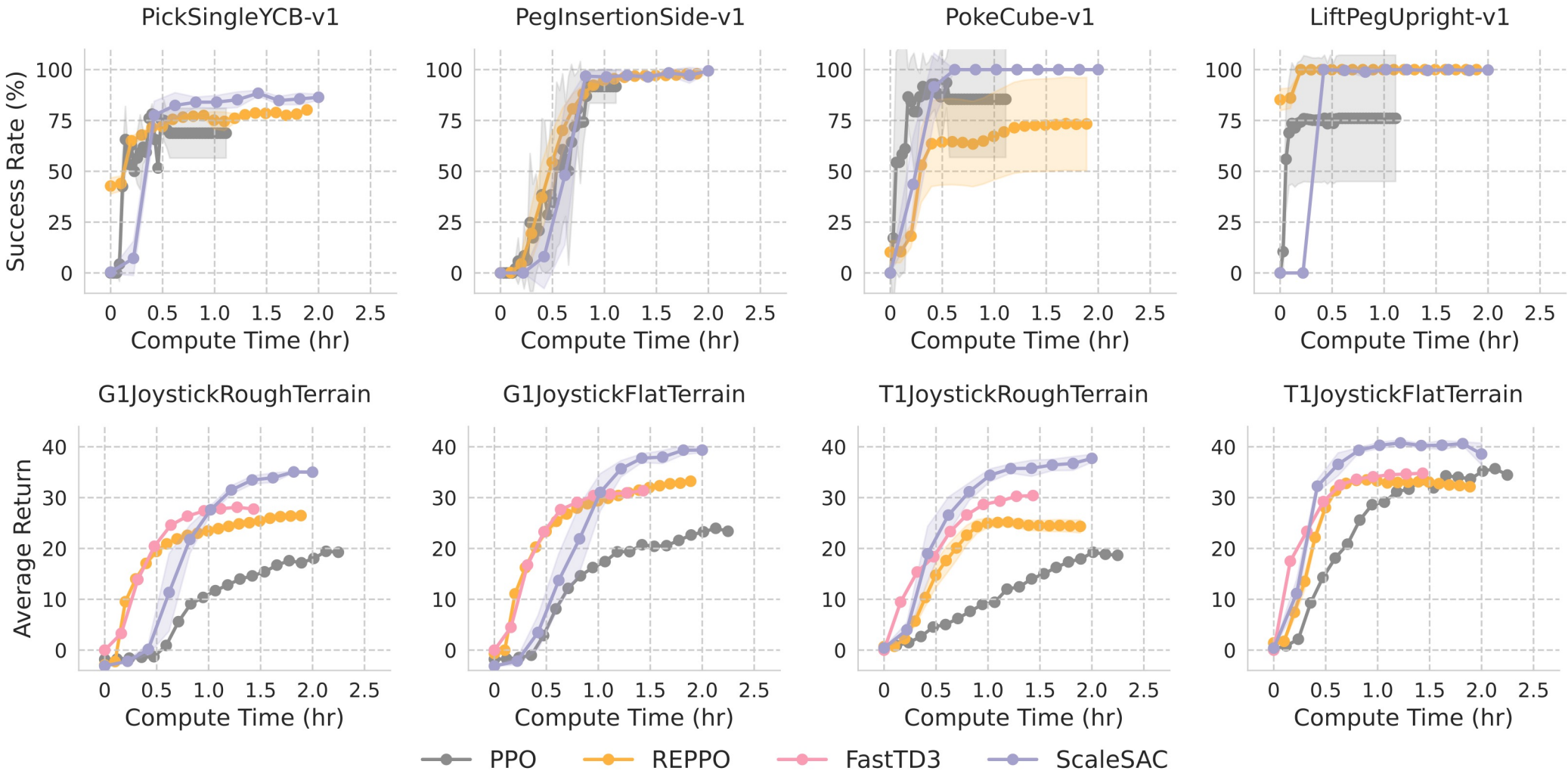
Baselines

- **PPO (RSL-RL)**: Defacto standard in Isaac Lab.
- **REPPO**: PPO with an added entropy bonus.
- **FastTD3**: SOTA compute-efficient off-policy algorithm.

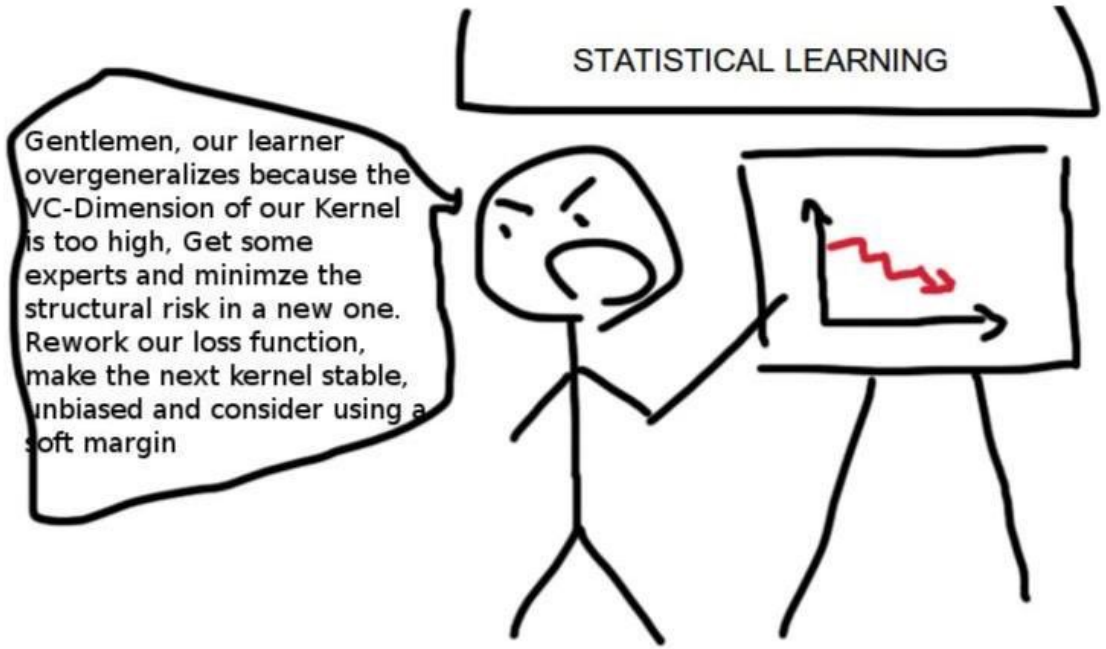
[1] RSL-RL: A Learning Library for Robotics Research, Schwarke et al, arXiv 2025.
[2] REPPO: Relative Entropy Pathwise Policy Optimization, Voelcker et al, arXiv 2025.
[3] FastTD3: Simple, Fast, and Capable Reinforcement Learning for Humanoid Control, arXiv 2025.

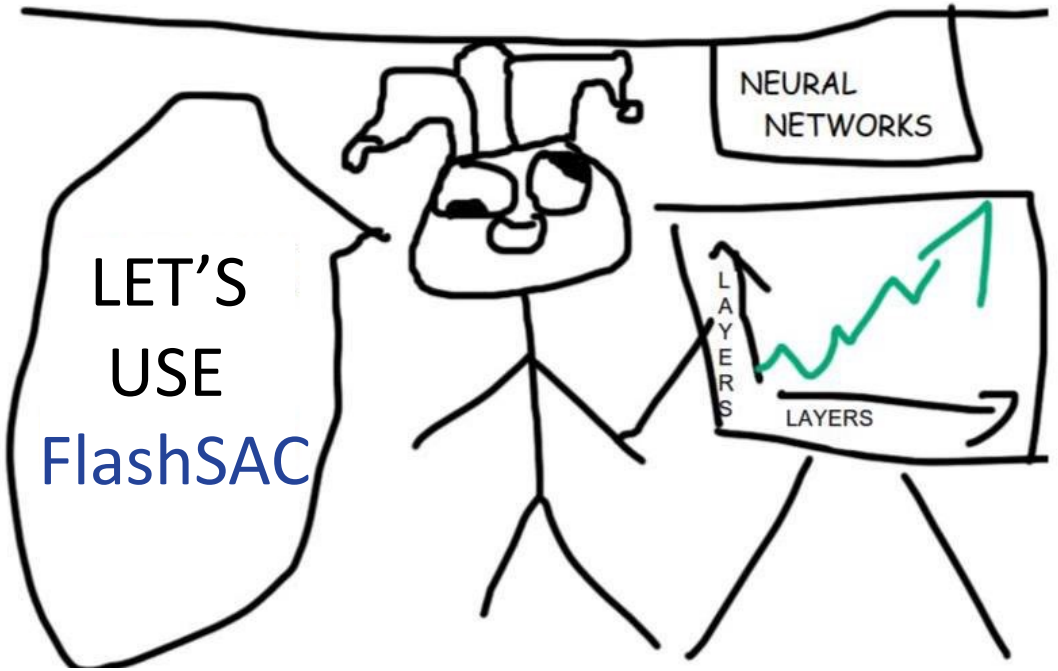
FlashSAC | Scaling Off-Policy RL for Speed

Benchmark Results



Summary


STATISTICAL LEARNING
Gentlemen, our learner overgeneralizes because the VC-Dimension of our Kernel is too high, Get some experts and minimze the structural risk in a new one. Rework our loss function, make the next kernel stable, unbiased and consider using a soft margin


LET'S
USE
FlashSAC
NEURAL
NETWORKS
LAYERS
LAYERS

Open Questions

Extension to Different Models & Algorithms

- Does SimbaV2's architectural philosophy transfer to :
  - Transformer-based LLMs?
  - Flow-based models in VLAs?
- Do the benefits persist in different RL algorithms? (likely yes…)
  - PPO?
  - TD-MPC2? (Model-based algorithms are known to suffer less from plasticity loss).

Sincere gratitude to all my Collaborators