Why we still do not have a good online RL algorithm in 2026?

Xuezhou Zhang

Boston University

April 1st, 2026

Why we still do not have a good online RL algorithm in 2026?

Xuezhou Zhang

Boston University

April 1st, 2026

Why we still do not have a good online RL algorithm in 2026?

Xuezhou Zhang

Boston University

April 1st, 2026

What is Online Reinforcement Learning (old-school, aka pre-2025)

“An intelligent agent learns to adapt in an environment in real-time.”

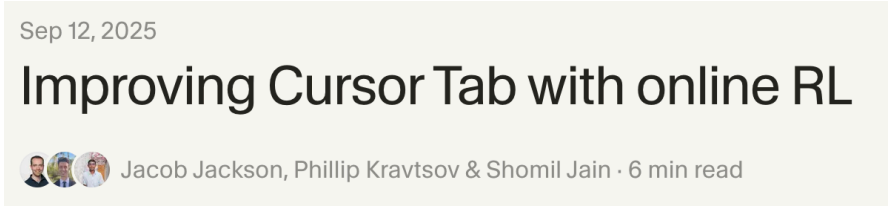
Online Reinforcement Learning

Not online RL:

- In-house Post-training of LLMs

Online RL:

- Online adaptation of LLMs to user feedbacks



Online Reinforcement Learning

Not online RL:

- In-house Post-training of LLMs
- Human-level game agent through self-play

Online RL:

- Online adaptation of LLMs to user feedbacks
- Learning to play poker inside a casino

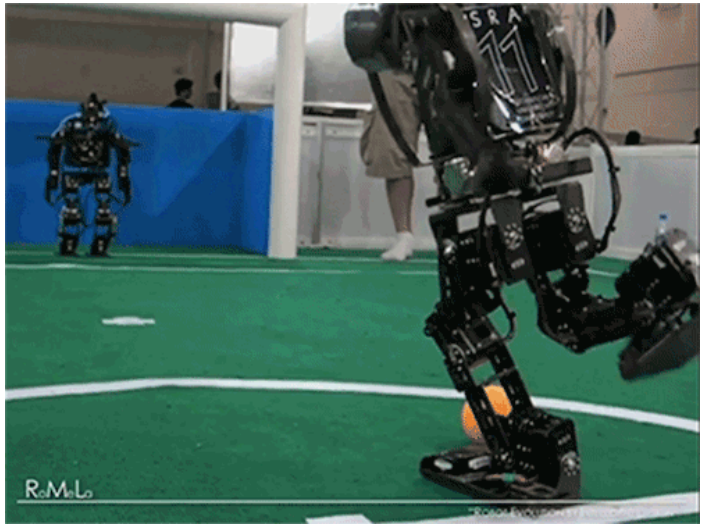
Online Reinforcement Learning

Not online RL:

- In-house Post-training of LLMs
- Human-level game agent through self-play
- Robot learning inside simulators

Online RL:

- Online adaptation of LLMs to user feedbacks
- Learning to play poker inside a casino
- Robot learning in real-time



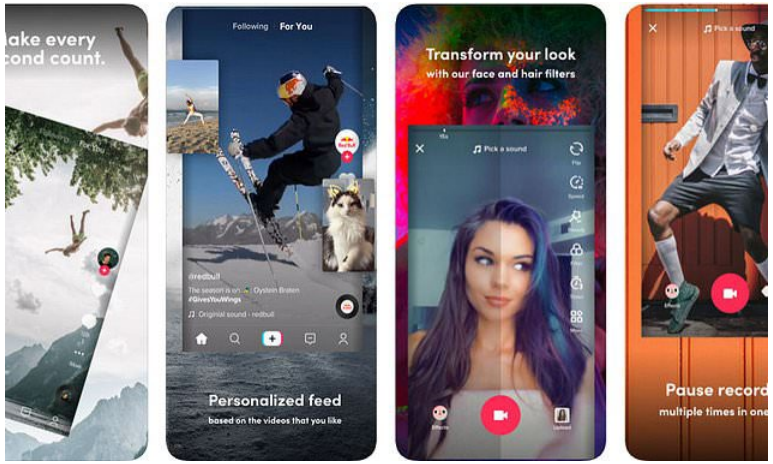
Online Reinforcement Learning

Not online RL:

- In-house Post-training of LLMs
- Human-level game agent through self-play
- Robot learning inside simulators
- …

Online RL:

- Online adaptation of LLMs to user feedbacks
- Learning to play poker inside a casino
- Robot learning in real-time
- Content Recommendation Systems



Online Reinforcement Learning

Not online RL:

- In-house Post-training of LLMs
- Human-level game agent through self-play
- Robot learning inside simulators
- …

Online RL:

- Online adaptation of LLMs to user feedbacks
- Learning to play poker inside a casino
- Robot learning in real-time
- Content Recommendation Systems
- Quantitative Trading

Key Difference: “the Stake of Learning”

Not online RL:

- Learning cost nothing but time and electricity.
- Mainly care about the final performance of the output policy π.
- Essentially stochastic optimization.

Online RL:

- Agent’s actions have real-world consequences.
- Cares about the efficiency of learning as much as the final policy.
  - Regret
  - Sample Complexity

Key Difference: “the Stake of Learning”

Not online RL:

- Learning cost nothing but time and electricity.
- Mainly care about the final performance of the output policy π.
- Essentially stochastic optimization.

➢ What 90% of the success of RL are on.

➢ Why??

Online RL:

- Agent’s actions have real-world consequences.
- Cares about the efficiency of learning as much as the final policy.
  - Regret
  - Sample Complexity

➢ What we hope to achieve as the ultimate goal.

The “Reward-Hacking” Hypothesis

- We, as a community, got reward-hacked.
  - ➢We wish to achieve **objective A** in practice
  - ➢but have been optimizing for **reward B** in development.

- Let’s look at a concrete example: hyperparameter tuning (where the real talk begins…)

Hyperparameter Optimization (HPO) in RL

- How we tune hyper-parameters in RL papers:
  - **M** set of candidate hyper-parameter configurations.
  - Run the algorithm with each config for **T** steps.
  - Report the performance of the best config.

- How we tune hyper-parameters in quantitative trading?
  - Unclear…

Let's get a bit more formal…

- Two metrics commonly used to measure "learning efficiency" in online RL:
  1. Regret:

$$\mathbf{Reg(T)} = \sum_{t=1}^{T} (V^* - V^{\pi_t})$$

  2. Sample Complexity (SC):

$$T_\epsilon = \min t, \mathbf{s.t.}\ V^* - V^{\pi_t} \leq \epsilon$$

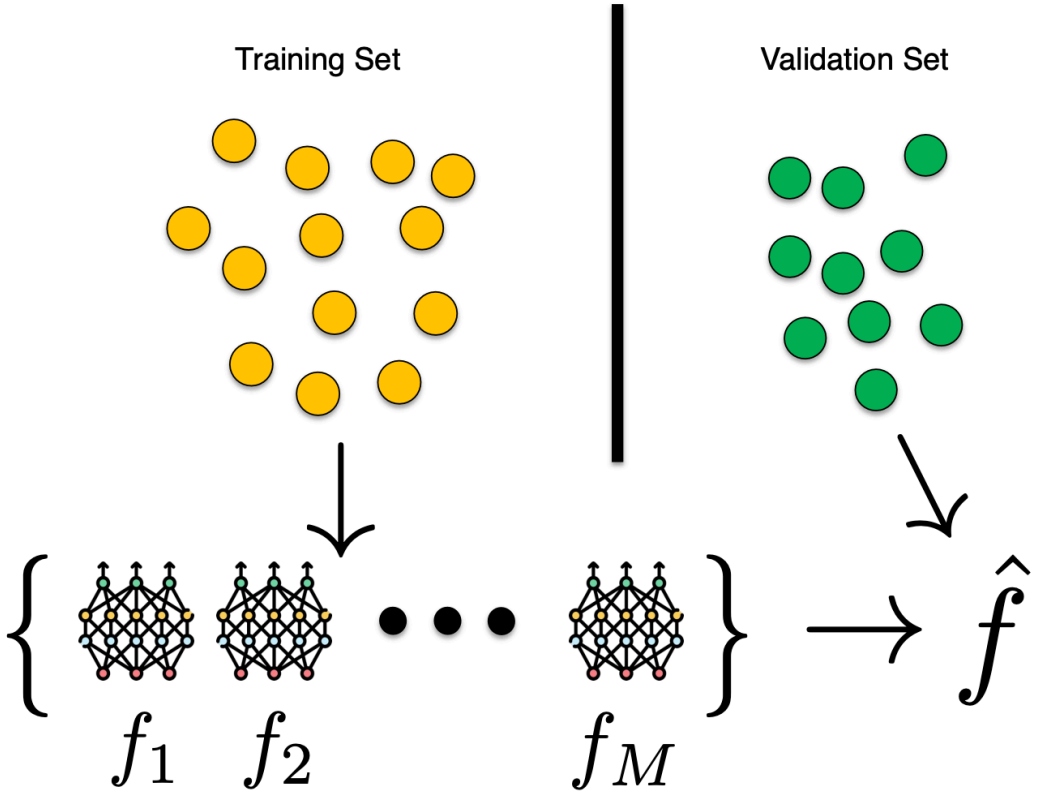
Hyperparameter Optimization (HPO) in RL

- Naïve HPO:
  - **M** set of candidate hyper-parameter configurations.
  - Run the algorithm with each config for **T** steps.
  - Report the performance of the best config.

- Regret of Naïve HPO: $\mathbf{Reg_M(T)} = \sum_{m=1}^{M} \mathrm{Reg}_m\left(\frac{T}{M}\right)$
- SC of Naïve HPO: $T_{\epsilon,M} = M \times \min_m T_{\epsilon,m}$

- How bad is it?

HPO in Supervised Learning



The Exponential Gap

- SC of HPO in SL: $T_{\epsilon,M} = \log M \times \min_m T_{\epsilon,m}$
- SC of HPO in RL: $T_{\epsilon,M} = M \times \min_m T_{\epsilon,m}$

How we evaluate algorithms currently

- **M = exp(# of h.p.)** if doing grid search.
- The consequence?
- Algorithms with a large number of hyperparameters are favored.

[Credit to Adkins et al., 2024.]

Hyperparameter Optimization (HPO) in RL

- Regret of Naïve HPO: $\mathbf{Reg_M(T)} = \sum_{m=1}^{M} \mathrm{Reg}_m\left(\frac{T}{M}\right)$
- SC of Naïve HPO: $T_{\epsilon,M} = M \times \min_m T_{\epsilon,m}$

- Can you do better? Spoiler: Yes and No.

- This is studied in the line of work called **online model selection**, of which Aldo Pacchiano is THE expert.

Hyperparameter Optimization (HPO) in RL

- **Theorem [SC Lower-bound]:** For any black-box HPO algorithm in online RL

$$T_{\epsilon,M} \geq \frac{M}{2} \times \min_{m} T_{\epsilon,m}$$

- Black-box HPO: a meta algorithm that assign each episode to a base learner. No data sharing among base learners.

- Proof TLDR: Early termination do not help.
- Naïve HPO [$T_{\epsilon,M} = M \times \min_{m} T_{\epsilon,m}$] is the best you can do.

Hyperparameter Optimization (HPO) in RL

- **Corollary [Regret Lower-bound]:** For any black-box HPO algorithm in online RL

$$\mathbf{Reg_M(T)} \geq \frac{\boldsymbol{M}}{\mathbf{2}} \times \min_{\boldsymbol{m}} \mathrm{Reg}_{\boldsymbol{m}}\left(\frac{\boldsymbol{T}}{\boldsymbol{M}}\right)$$

- Proof TLDR: Regret -> SC reduction.

- If $\min_{\boldsymbol{m}} \mathrm{Reg}_{\boldsymbol{m}}(\boldsymbol{T}) = \boldsymbol{\Theta}(\sqrt{\boldsymbol{T}})$, then $\mathbf{Reg_M(T)} \geq \sqrt{\boldsymbol{M}} \times \min_{\boldsymbol{m}} \mathrm{Reg}_{\boldsymbol{m}}(\boldsymbol{T})$.

Part 1 Summary

- Regret of Naïve HPO: $\mathbf{Reg_M(T)} = \sum_{m=1}^{M} \mathrm{Reg}_m\left(\frac{T}{M}\right)$
- SC of Naïve HPO: $T_{\epsilon,M} = M \times \min_m T_{\epsilon,m}$

What actually gets reported in empirical papers…

- Lower bounds:
- $\mathbf{Reg_M(T)} \geq \sqrt{M} \times \min_m \mathrm{Reg}_m(T)$
- $T_{\epsilon,M} \geq \frac{M}{2} \times \min_m T_{\epsilon,m}$

Algorithm Evaluation done properly?

- Effective SC: $\boldsymbol{M} \times \min_{\boldsymbol{m}} \boldsymbol{T}_{\epsilon,\, \boldsymbol{m}}$

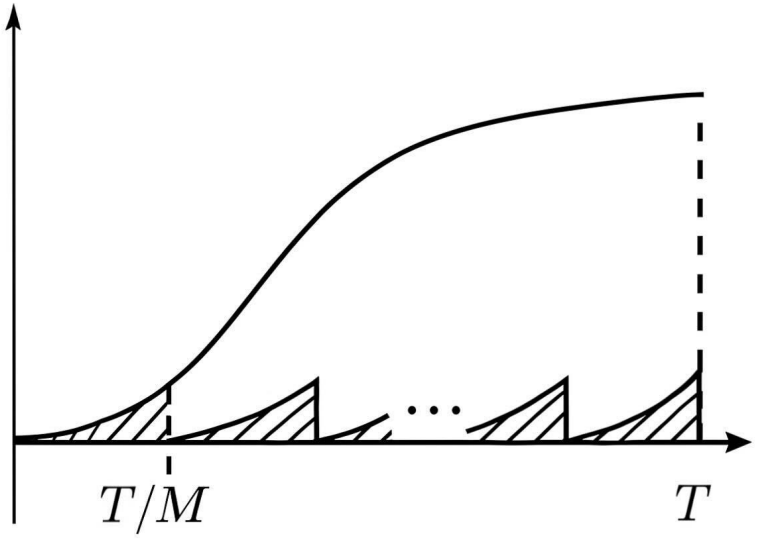
- Straight-forward, since upper and lower bound match.

Algorithm Evaluation done properly?

- Effective Regret: $\boldsymbol{M} \times \min_{\boldsymbol{m}} \mathrm{Reg}_{\boldsymbol{m}}(\boldsymbol{T}/\boldsymbol{M})$

"Optimistic": in the sense that this is the best that one can achieve with an optimal HPO algorithm.

- Effective AUC: $\boldsymbol{M} \times \max_{\boldsymbol{m}} \mathrm{AUC}_{\boldsymbol{m}}(\boldsymbol{T}/\boldsymbol{M})$

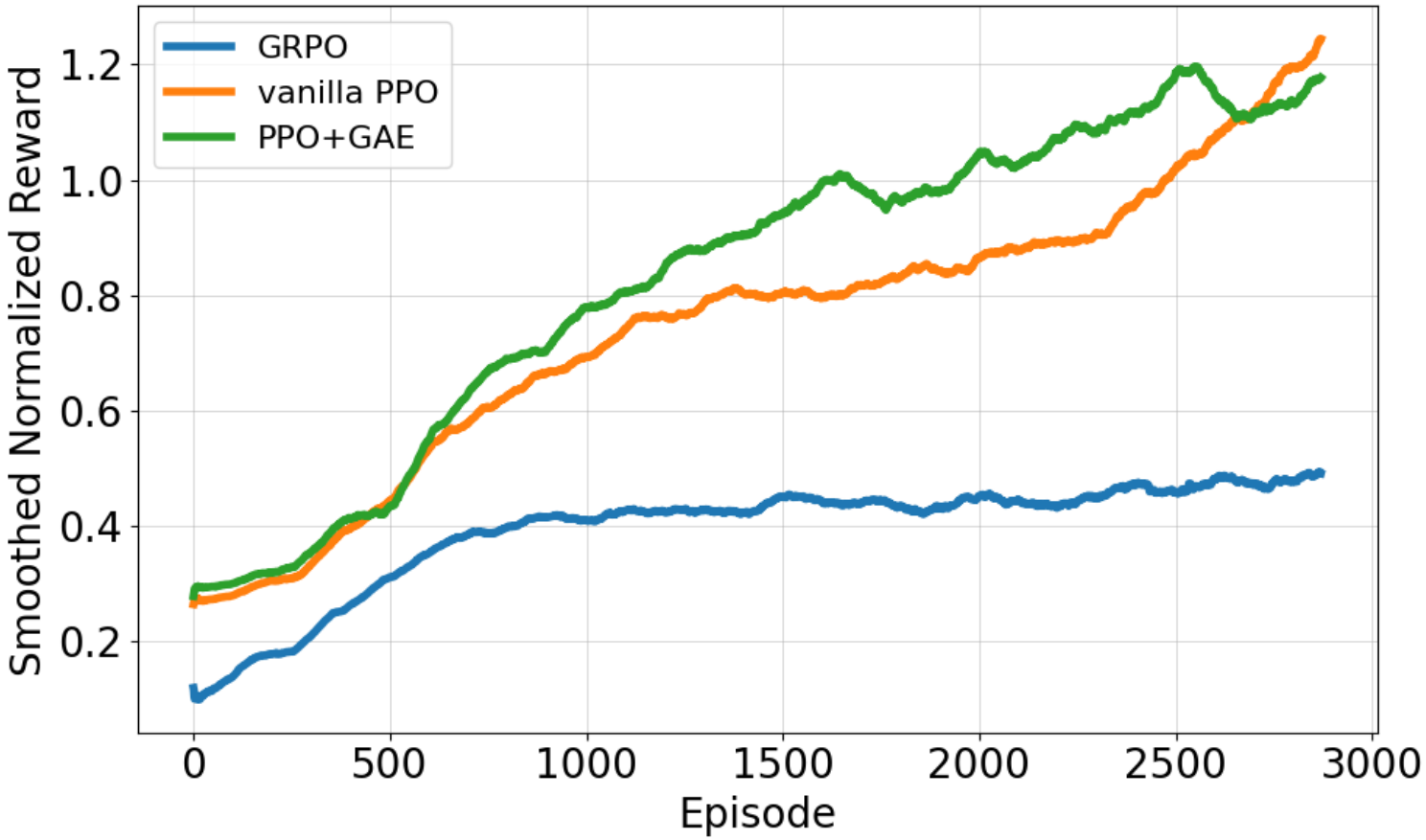


Use Case I: Algorithm Comparison

Use Case I: Algorithm Comparison

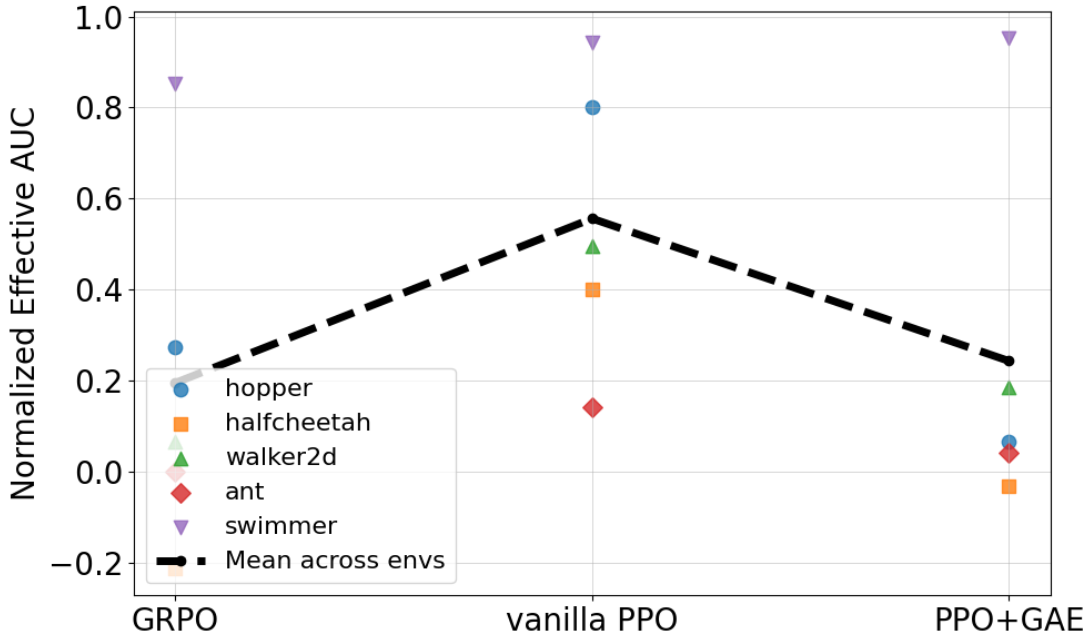
- Hyperparameters:
  - GRPO (n=2): actor LR, entropy coefficient
  - Vanilla PPO (n=3): actor LR, entropy coefficient, critic LR
  - PPO+VAE (n=4): actor LR, entropy coefficient, critic LR, GAE λ
- M = 3^n

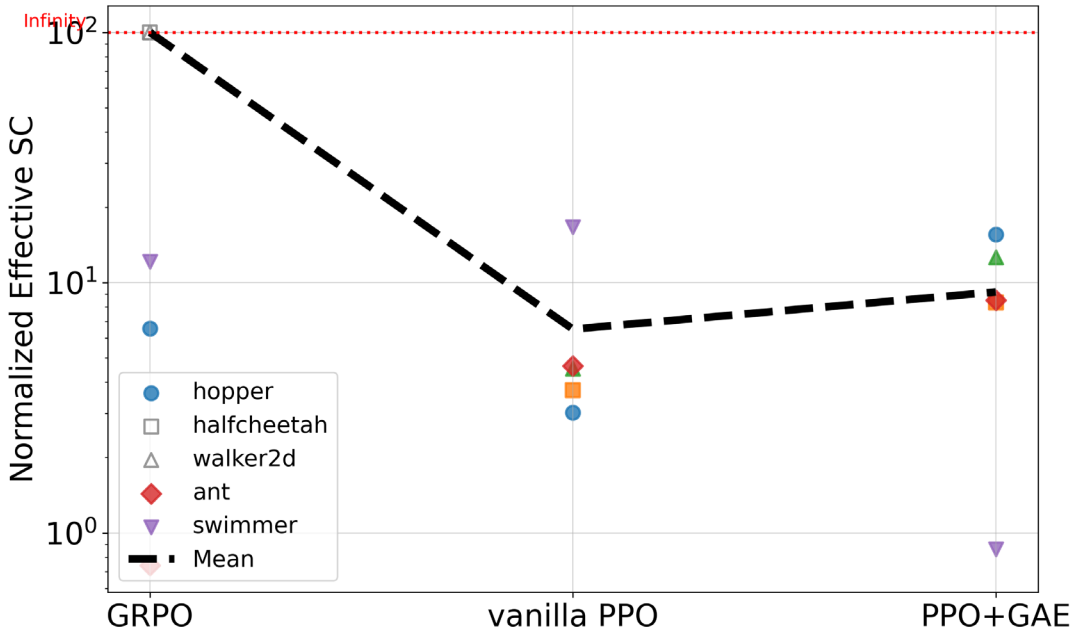
Plot in a typical RL paper



Take-home message: PPO+GAE > Vanilla PPO > GRPO

HPO-aware Evaluation





Under both metric: Vanilla PPO > PPO+GAE > GRPO

Use Case II: Which h.p. should I tune?

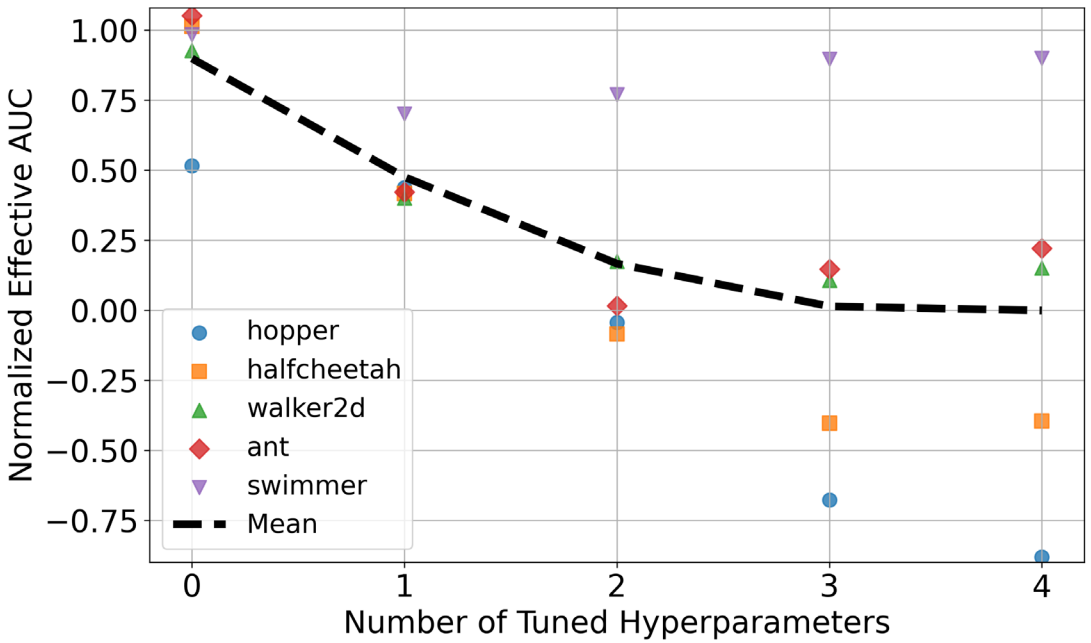
- PPO+GAE (4): actor LR, entropy coefficient, critic LR, GAE λ
- Which hyperparameters should I fix and which ones should I tune per task?
- 2^4 total cases

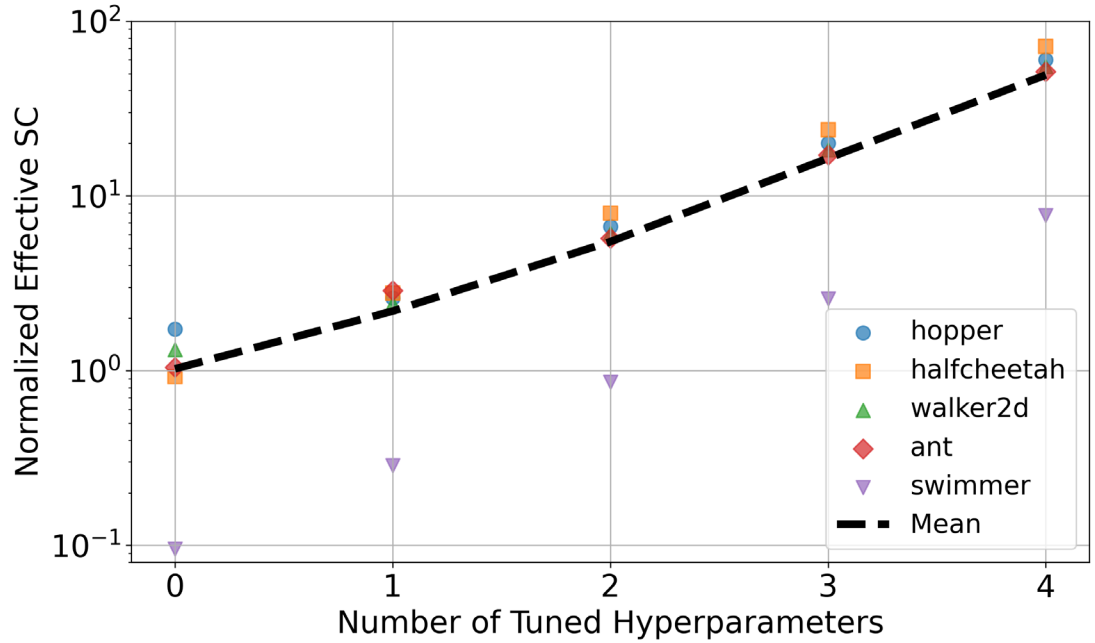
Use Case II: Which h.p. should I tune?

- For each case, find the best value for the fixed hyperparameters on auxiliary tasks.

- Tune the rest of the parameters online and report the effective AUC/SC.

Use Study II: Which h.p. should I tune?





Part 2 Summary

- Effective AUC/SC allow us to

    1. Evaluate and compare algorithms more fairly.

    2. Quantitatively answer HPO-related design questions.

Can we stop this trend?

These might help…

- Start evaluate algorithms with HPO-aware metrics.
- Design algorithms with fewer sensitive hyperparameters.
- Design scalable and robust online HPO algorithms.