Evaluation of Meta-Learned Optimizers for Reinforcement Learning

Arseniy Kan
Electrical Engineering
UNIST
arseniykan@unist.ac.kr

Seongju Meang
Artificial Intelligence
UNIST
meangsungjoo@unist.ac.kr

Abstract

Optimization in Deep Reinforcement Learning (RL) presents unique challenges compared to Supervised Learning, primarily due to non-stationarity, plasticity loss, and the lack of independent and identically distributed (IID) data. Recently, meta-learned optimizers have emerged as a promising direction to address these specific difficulties by learning update rules directly from data. This project conducts a comparative analysis between two distinct approaches in meta-learned optimization: **Optim4RL** (Learning to Optimize for Reinforcement Learning), which utilizes pipeline training and a dual-RNN architecture to handle non-IID gradients, and **OPEN** (Learned Optimization for Plasticity, Exploration and Non-stationarity), which targets specific RL difficulties via learned stochasticity and input feature engineering. We evaluate these methods against traditional hand-crafted optimizers, specifically Adam and RMSProp, across a range of discrete and continuous control tasks. Our experiments utilize Atari-like environments in `Gymnax` (Breakout, Asterix, Space Invaders, Freeway) for discrete control and `Brax` environments (Ant, Humanoid) for continuous control. We report on the training stability, final return, and generalization capabilities of these learned optimizers to assess their viability as general-purpose solutions for RL. Our code is available at `https://github.com/arseniykan/meta-optim-eval`

1 Introduction

Reinforcement Learning (RL) has achieved remarkable success in solving complex sequential decision-making problems, ranging from game playing to continuous control [3]. However, the optimization process in RL is notoriously unstable and sample-inefficient compared to Supervised Learning (SL). This difficulty stems from unique challenges inherent to the RL paradigm, including non-stationary data distributions, high variance in gradient estimates, and the "deadly triad" of function approximation, bootstrapping, and off-policy learning [3; 1].

Traditionally, RL practitioners rely on hand-crafted optimizers such as Adam or RMSProp [4; 5]. While effective, these general-purpose optimizers are designed under the assumption of independent and identically distributed (IID) data and stationary objectives—assumptions that are frequently violated in RL [1]. Consequently, they often struggle to address specific RL pathologies like plasticity loss and premature convergence to local optima [2].

Meta-learning, or "learning to learn," offers a data-driven alternative: replacing hand-designed update rules with parameterized optimizers trained to maximize agent performance. While learned optimizers like VeLO have shown promise in SL, they often fail to generalize to RL tasks [6; 1].

This project presents a comparative analysis of two state-of-the-art meta-learned optimizers specifically designed to overcome these limitations:

Preprint. Under review.

1. **Optim4RL** (Learning to Optimize for Reinforcement Learning) [1]: This method addresses the non-IID nature of RL gradients. It introduces a hierarchical RNN architecture and a "pipeline training" curriculum to stabilize the meta-learning process.
2. **OPEN** (Learned Optimization for Plasticity, Exploration, and Non-stationarity) [2]: This approach explicitly targets the difficulties of plasticity loss and exploration. It utilizes a learned update rule that conditions on agent statistics (e.g., neuronal dormancy) and injects learned stochasticity into parameter updates.

We evaluate these methods against traditional baselines (Adam, RMSProp) across a diverse set of environments, including discrete control tasks in Gymnax (Breakout, Asterix, Space Invaders, Freeway) [7] and continuous control tasks in Brax (Ant, Humanoid) [8]. By rigorously benchmarking these approaches, we aim to verify whether meta-learned optimizers can serve as reliable, general-purpose solutions for the distinct challenges of reinforcement learning.

2 Background

2.1 Reinforcement Learning and Optimization Challenges

Formally, we consider an agent interacting with an environment modeled as a Markov Decision Process (MDP). The goal is to learn a policy π_θ that maximizes the expected cumulative reward. In practice, this optimization landscape is fraught with difficulties not present in SL:

- **Non-IID Data and Non-Stationarity:** Unlike the fixed datasets in SL, the data distribution in RL is generated by the agent's current policy. As the policy evolves, the input distribution shifts, leading to non-stationary learning dynamics that can confuse standard momentum-based optimizers [1].
- **Plasticity Loss:** Recent studies have highlighted the phenomenon of "dormant neurons," where network capacity degrades over time, preventing the agent from adapting to new experiences later in training [2].
- **Exploration vs. Exploitation:** To maximize returns, an optimizer must facilitate exploration of the parameter space to avoid getting stuck in poor local optima, a requirement that standard gradient descent does not explicitly address [2].

2.2 Meta-Learned Optimization

Meta-learned optimization formulates the search for an update rule as a bilevel optimization problem. The "inner loop" trains an agent θ using the learned optimizer ϕ, while the "outer loop" updates ϕ to maximize the agent's final performance $J(\theta)$. The update rule is typically parameterized as a recurrent neural network (RNN) that takes gradients and other statistics as input and outputs parameter updates:

$$\theta_{t+1} = \theta_t + U_\phi(g_t, h_t) \tag{1}$$

where g_t is the gradient and h_t is the internal state of the optimizer.

Figure 1: Conceptual comparison between a standard hand-crafted optimizer (left) and a meta-learned optimizer (right), which maintains an internal state to adapt update rules dynamically.

2.3 Survey of Recent Methods

Early work in learned optimization demonstrated that recurrent networks could outperform SGD on simple supervised tasks. More recently, large-scale efforts like VeLO [6] employed massive compute resources to train general-purpose optimizers. However, these models often perform poorly in RL due to the high variance and bias of RL gradients [1]. Parallel efforts such as Discovered Policy Optimisation (LPO) [9] have attempted to discover reinforcement learning algorithms directly, but optimizing the update rule itself remains a distinct challenge.

To bridge this gap, specific architectures have been proposed:

- **Optim4RL** focuses on the stability of the meta-training process. It identifies that the distribution of gradients in RL is highly non-IID. To mitigate this, it employs *Pipeline Training*, where the optimizer is trained on gradients sampled from diverse stages of the agent's training simultaneously [1].
- **OPEN** takes a feature-engineering approach. It conditions the optimizer on additional inputs such as "dormancy" (a measure of dead neurons) and training progress. Furthermore, it introduces a novel exploration mechanism where the optimizer outputs learnable noise, effectively performing parameter-space exploration [2].

2.4 Theoretical Insights and Limitations

Theoretical analysis suggests that meta-learning in RL suffers from a "vicious spiral" of bilevel optimization [1]. A poor initial optimizer leads to a poor policy, which generates low-quality, noisy data. This noisy data, in turn, provides high-variance meta-gradients that hinder the improvement of the optimizer.

Additionally, *Plasticity Loss* has been identified as a critical theoretical bottleneck. As training progresses, non-stationarity causes neural networks to lose their ability to learn (plasticity). While standard optimizers like Adam have no mechanism to counteract this, recent insights suggest that explicitly tracking and targeting dormant neurons can sustain plasticity throughout training [2].

Despite these advances, limitations remain. Meta-learned optimizers often struggle with generalization to architectures or environments significantly different from their training distribution (out-of-distribution generalization). Furthermore, the computational cost of meta-training (often requiring thousands of GPU hours) remains a barrier to widespread adoption [6].

3 Experimental Setup

To rigorously evaluate the performance and generalization capabilities of meta-learned optimizers, we designed a comprehensive experimental framework that reproduces key findings from prior literature while extending the analysis to multi-task settings and hyperparameter sensitivity.

3.1 Evaluation Environments

Our experiments span both discrete and continuous control domains to test optimizer versatility:

- **Continuous Control (Brax):** We utilize the *Ant* environment, a high-dimensional locomotion task known for its complex dynamics and sensitivity to hyperparameter choices.
- **Discrete Control (Gymnax/MinAtar):** We employ four Atari-like environments—*Breakout*, *Asterix*, *Space Invaders*, and *Freeway*. These environments serve as a benchmark for multi-task learning performance.

3.2 Optimizers Compared

We conduct a comparative analysis of four distinct optimization algorithms:

1. **Adam (Baseline):** A standard adaptive moment estimation optimizer, widely used as a robust baseline in RL.
2. **RMSProp:** A gradient normalization method that often performs well in non-stationary RL settings.
3. **OPEN:** A meta-learned optimizer designed to explicitly target plasticity loss and exploration via learned stochasticity and feature engineering.
4. **Optim4RL:** A meta-learned optimizer utilizing a dual-RNN architecture and pipeline training to handle non-IID gradients.

3.3 Experimental Protocol

Our evaluation focuses on three primary objectives:

- **Reproduction of Baselines:** We reproduce experiments from the original Optim4RL and OPEN papers in the Ant environment to verify reported performance gains.
- **Multi-Task Generalization:** We evaluate the "Multi-OPEN" claim by testing the optimizer's ability to generalize across the four discrete Atari-like games and the continuous Ant environment simultaneously, comparing it against tuned Adam and RMSProp baselines.
- **Hyperparameter Sensitivity Analysis:** We investigate the claim of being "hyperparameter-free" by specifically analyzing the dependency of Optim4RL on its meta-hyperparameters during training and deployment.

4 Results

5 Discussion

5.1 Architectural Efficacy: OPEN vs. Optim4RL

Our comparative analysis indicates that OPEN represents a significant architectural advancement over Optim4RL. While Optim4RL relies on a rigid, Adam-like update structure using implicit RNN representations to handle non-IID gradients [1], OPEN employs a more flexible approach. By explicitly conditioning on RL-specific features—such as neuronal dormancy and training progress—and managing exploration via learned stochasticity, OPEN consistently outperforms Optim4RL in multi-task settings [2]. This validates the hypothesis that targeting specific RL pathologies (plasticity loss, non-stationarity) via input feature engineering is more effective than relying solely on curriculum-based stabilizations like pipeline training.

5.2 The Generalization Gap: Discrete vs. Continuous Control

We observed a distinct performance disparity across domains. In discrete control tasks (MinAtar), meta-learned optimizers demonstrated clear superiority, with OPEN significantly outperforming baselines in environments like *Asterix* and *Breakout*. However, this dominance did not transfer to continuous control. In the *Ant* environment, OPEN merely achieved parity with Adam and failed to generalize when tested in an out-of-distribution capacity. This suggests that while learned update rules can internalize the dynamics of value-based discrete RL effectively, the high variance and sensitivity of continuous control policy gradients remain a barrier to robust generalization.

5.3 The "Meta-Tuning" Paradox

A central promise of learned optimization is the automation of hyperparameter tuning. However, our reproduction of Optim4RL reveals a "tuning shift" rather than a reduction. While the optimizer eliminates the need to tune the inner-loop learning rate, it introduces a dependency on meta-hyperparameters. Our sensitivity analysis showed that performance in the *Ant* environment fluctuated drastically based on the choice of meta-learning rate and reset intervals. Consequently, the burden of manual tuning is not removed but merely shifted one level up to the meta-training phase, contradicting the claim of a truly "hyperparameter-free" solution.

5.4 Policy Collapse and Regularization

Our experiments on the *Ant* task highlighted the susceptibility of standard optimization schedules to policy collapse. We found that standard Adam, with its asymmetric momentum ($\beta_1 = 0.9, \beta_2 = 0.999$), often leads to excessive updates that destabilize the policy. In contrast, applying L2 regularization to the learned Optim4RL optimizer significantly mitigated this collapse, suggesting that learned optimizers still require traditional regularization techniques to maintain stability in non-stationary environments.

6 Conclusion

This project conducted a comprehensive evaluation of meta-learned optimizers for Reinforcement Learning, comparing Optim4RL and OPEN against industry-standard baselines. Our results lead to three primary conclusions:

1. **Domain Specificity:** Meta-learned optimizers are currently "specialists" rather than "generalists." They excel in discrete, sparse-reward environments where exploration is difficult, but they struggle to consistently outperform tuned Adam in continuous control tasks.
2. **Barriers to Adoption:** The computational overhead of meta-training, combined with the "tuning shift" paradox where meta-hyperparameters require extensive validation, currently hinders widespread practical adoption.
3. **Design Philosophy:** The success of OPEN over Optim4RL highlights that explicit architectural interventions targeting specific RL difficulties (e.g., plasticity) are more promising than purely data-driven, implicit representations.

While meta-learned optimization offers a promising frontier for addressing the non-stationary nature of RL, current methods do not yet provide a robust, "plug-and-play" replacement for hand-crafted optimizers like Adam across all domains. Future work must address the generalization gap in continuous control and reduce the sensitivity to meta-training configurations.

References

References

[1] Q. Lan, A. R. Mahmood, S. Yan, and Z. Xu. Learning to Optimize for Reinforcement Learning. *Reinforcement Learning Conference (RLC)*, 2024. Also arXiv:2302.01470.

[2] A. D. Goldie, C. Lu, S. Whiteson, M. T. Jackson, and J. N. Foerster. Can Learned Optimization Make Reinforcement Learning Less Difficult? *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[4] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[5] T. Tieleman and G. Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[6] L. Metz, J. Harrison, et al. VeLO: Training Versatile Learned Optimizers by Scaling Up. *arXiv preprint arXiv:2211.09760*, 2022.

[7] R. T. Lange. gymnax: A JAX-based reinforcement learning environment library. 2022. http://github.com/RobertTLange/gymnax

[8] C. D. Freeman, E. Frey, et al. Brax: A differentiable physics engine for large scale rigid body simulation. 2021. http://github.com/google/brax

[9] C. Lu, J. Kuba, et al. Discovered Policy Optimisation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.