Final Report

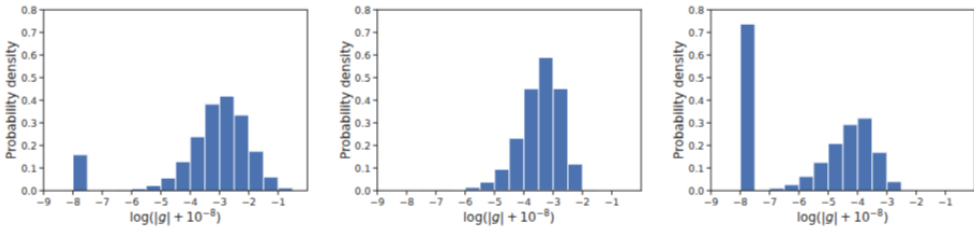
20255521 맹성주

Arseniy kan

Github link :

Problem setup

1. Non IID data and Non-stationary Targets



(a) At the beginning of training (b) In the middle of training (c) At the end of training

Traditional optimizers (e.g., Adam, RMSProp) assume that data is Independent and Identically Distributed (IID).

However, RL relies on locally correlated trajectories generated through agent environment interactions, violating the IID assumption.

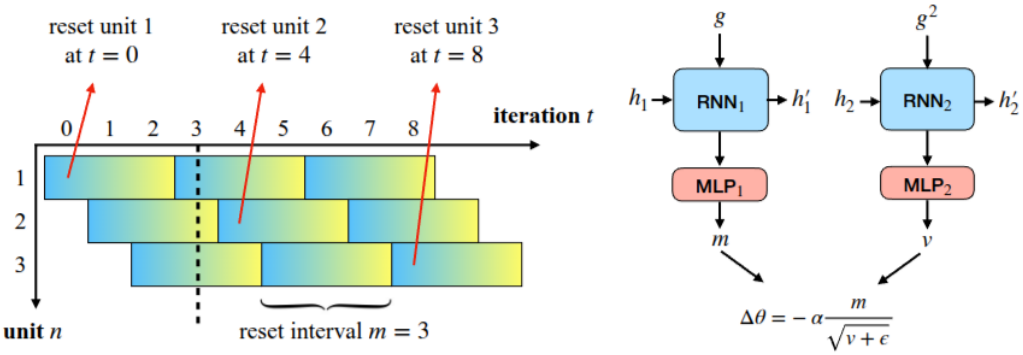
Furthermore, as the agent's policy evolves, the distribution of collected data shifts, leading to non-stationary targets.

This stochastic nature introduces high bias and variance into the gradients, complicating the optimization landscape.

2. Hyperparameter Sensitivity and the Meta Optimization Gap

Standard optimizers require an extensive hyperparameter. While Meta learned optimizers -which learn update rules directly from data- have been proposed as a solution, existing models like VeLO perform poorly in RL despite their success in supervised learning. Consequently, developing an effective meta optimization strategy that can adapt to the dynamic and unstable nature of RL remains an unsolved problem.

Methods



(a) Pipeline training (b) The optimizer structure of Optim4RL

Optim4RL employs two key strategies to tackle the non-stationary and non-IID data issues inherent in RL. Then apply this strategy to train the meta optimizer.

1. Pipeline training (optim4RL)

By initializing agents at different intervals (Reset intervals) within parallel environments, this strategy mixes gradients from early, middle, and late training stages.

This allows the optimizer to learn from a data distribution that more closely resembles an IID distribution.

2. Inductive bias (optim4RL)

To prevent the learning of erratic update rules, the search space is constrained to follow an Adam-like formula structure$\left(\Delta\theta = -\alpha\frac{m}{\sqrt{v+\epsilon}}\right)$. Momentum and velocity are generated via Dual RNN that learn temporal patterns of gradient changes.

OPEN is explicitly designed to address fundamental RL hurdles and adopt a different approach compared to Optim4RL

1. Explicit inputs & control (OPEN)

Optim4RL relies on gradients, OPEN utilizes explicit input features representing RL states, such as plasticity loss and non-stationarity. Furthermore, it employs stochastic updates (Mean+Noise) with learned noise to actively manage exploration.

2. Training stability (OPEN)

Instead of meta-gradients(bilevel optimization), OPEN uses Evolution strategies to enhance training stability and maximize parallelization.

Experimential design

1. Policy collapse Experiment (Optim4RL)

While Adam is widely used in RL tasks and has demonstrated the ability to complete training without collapse in Optim4RL paper. However, Since Adame default hyperparameters, specifically the asymmetric momentum coefficients($\beta_1 = 0.9, \beta_2 = 0.999$), These settings can create excessive updates that lead to policy collapse, a phenomenon where a well learned policy drastically degrades in performance during later stages of training and fails to recover.

In constrast to Adam,which relies on fixed hyperparameters, optim4RL utilizes dual RNN to learn temporal gradient patterns and dynamically generate momentum and velocity. Theoretically, this adaptive mechanism should allow the optimizer to adjust its update rules according to the non-stationary nature of the RL environment, thereby mitigating the risk of collapse.

Therefore, this experiment aims to go beyond the empirical observation that Adam can work, to systematically verify whether Optim4RL possesses a structural advantage over Adam in preventing Policy Collapse. Furthermore, we investigate the impact of mitigation strategies, such as L2 regularization, on the stability of Optim4RL should collapse phenomena occur.

To induce policy collapse, We replaced the stationary task to non-stationary task, introducing continuous variations in dynamics such as simulated actuator wear and changing ground friction during training. And The training duration was extended by a factor of 10(1e8 to 1e9). This long horizon training is designed to induce the policy collapse. Lastly, we performed an additional experiment to investigate whether an Optim4RL optimizer trained on stationary tasks can generalize to non-stationary tasks. We performed experiments using the following setup.

1-1. PPO + default Adam with non-stationary task (Ant task)

1-2. PPO + tunned Adam with non-stationary task (Ant task)

1-3. PPO + optim4RL optimizer with non-stationary task (Ant task)

1-4. PPO + optim4RL optimizer with L2 regularization + non-stationary task (Ant task)

1-5 PPO + learned optimizer (stationary tasks) to non-stationary tasks

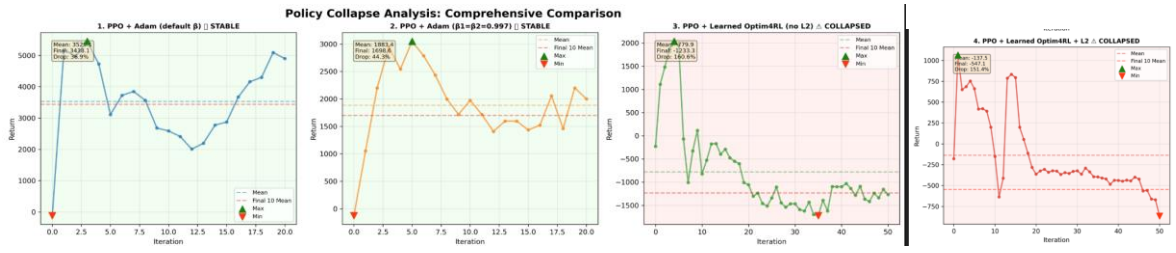
2. Comparison of the OPEN Optimizer and the Standard Optimizer in Discrete and Continuous Environments

The architecture of OPEN employs a feature-engineering approach, conditioning the optimizer on agent statistics such as neuronal dormancy and training progress. Furthermore, it actively manages exploration by injecting learned stochasticity into parameter updates to prevent getting stuck in local optima. For the meta-training phase, it utilizes Evolution Strategies (ES) via JAX instead of standard gradient-based methods, ensuring stability and high parallelizability.

- **Evaluation Environments:** Experiments utilize Atari-like environments in Gymnax (Breakout, Asterix, Space Invaders, Freeway) for discrete control and Brax (Ant, Humanoid) for continuous control.

- **Baselines:** OPEN is compared against hand-crafted optimizers (Adam, RMSProp) and other meta-learned approaches like Optim4RL and VeLO.

Results and Analysis



1. **PPO + Adam optimizer (graph1,2)**

   As observed in Graphs 1 and 2, we initially hypothesized that the Adam optimizer would trigger a policy collapse in non-stationary tasks. However, the experimental results show that a permanent collapse did not occur. Although performance in the non-stationary environment was lower than in stationary tasks, the agent demonstrated a clear ability to recover from performance dips.

   The reason Adam avoided a total collapse is likely due to the clipping mechanism of the PPO algorithm. While Adam's stale momentum might suggest an erroneously large update step (the theoretical cause of collapse), PPO's trust-region constraints act as a "hard brake," preventing the policy from entering an unrecoverable state. Thus, Adam's theoretical instability is effectively masked and mitigated by the algorithmic safeguards of PPO.

2. **Meta learned optimizer(optim4RL - graph3,4)**

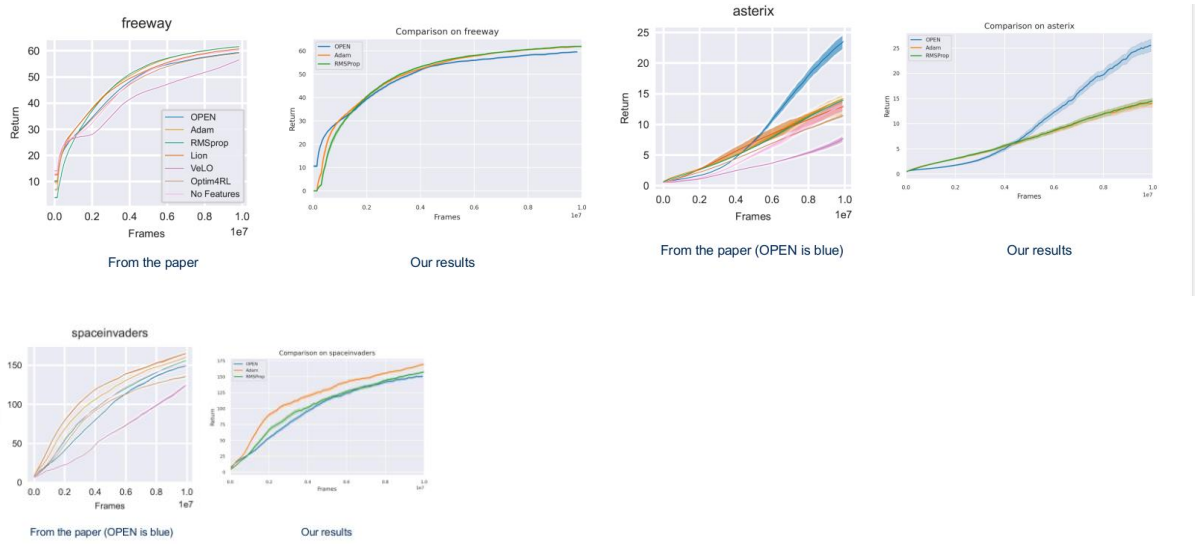
   In Graphs 3 and 4, contrary to our expectation that the Optim4RL optimizer would be more robust against policy collapse, the results showed poor performance and severe collapse. While the addition of L2 regularization in Graph 4 showed temporary recovery intervals—suggesting a slight mitigation effect—it ultimately failed to overcome the collapse. The collapse in direct meta-training occurs due to Meta-optimization Divergence. Since the environment is non-stationary, the "target" for the meta-optimizer is constantly moving. This creates extremely noisy meta-gradients, making it nearly impossible for the optimizer to learn a stable update rule. The meta-learner confuses environmental shifts with poor update steps, leading to a divergent update policy that eventually destroys the agent's neural weights. L2 regularization only limits the magnitude of these weights but cannot correct the fundamentally erroneous direction of the updates.

3. **Transfer learning(Graph 5)**

   The results of applying a stationary-task-trained Optim4RL to a non-stationary task

show that it maintains high performance initially but eventually suffers from a catastrophic policy collapse. The initial high-performance stems from the optimizer successfully learning the fundamental physics and dynamics of the robot (Ant) in a stable environment. However, the eventual collapse happens because the optimizer is overfitted to the stationary regime. In a stationary task, the optimizer learns that the loss of landscape is fixed. When moved to a non-stationary task, it lacks the "flexibility" or "meta-adaptability" to recognize that the landscape has shifted. It continues to apply "aggressive stationary rules" to a "changing environment," which eventually pushes the policy parameters off a cliff once the distribution shift becomes too large.

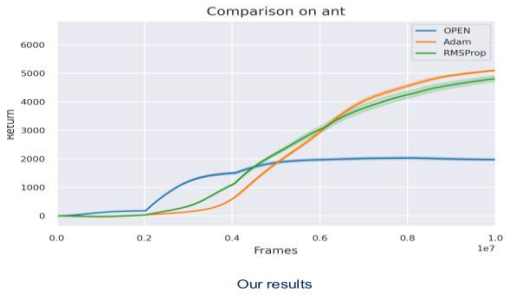
4. Multitask performance of Tested OPEN pretrained on 3 Atari-like Environment (OPEN)



- **Discrete Control:** OPEN significantly outperformed all baselines in environments, validating its design choice to target RL-specific hurdles.

- **Multi-Task Capabilities:** OPEN was the only learned optimizer to achieve an average score higher than the Adam baseline across the multi-task suite, demonstrating its potential as a Foundation Optimizer.

5. **Continuous control (Brax Ant task)**



Our results

The Continuous Control Gap: In the Brax Ant environment, OPEN achieved lower performance compared to Adam and RMSProp. and failed to generalize tasks.

Limitations

- **Hardware Constraints (OOM Issues):** A primary limitation was the Out-of-Memory (OOM) error encountered when meta-training the Optim4RL optimizer from scratch. Due to limited GPU memory, we had to simplify the training process by reducing the batch size and update frequency. These constraints likely hindered the meta-optimizer's ability to capture complex gradient patterns, contributing to the poor performance seen in Graphs 3 and 4.

- **Computational Cost:** The meta-training process is extremely resource-intensive, often requiring thousands of GPU hours. This high computational overhead remains a major barrier to widespread adoption in the industry.

- **The Vicious Spiral of Bilevel Optimization:** Meta-learning in RL often falls into a "vicious spiral." A poor initial optimizer results in an ineffective policy, which

generates low-quality and noisy data. This noisy data then produces high-variance meta-gradients, preventing the optimizer from improving.

- **Task Diversity:** The meta-learning was performed primarily on the Ant task. The lack of diverse environments during the meta-training phase may have limited the optimizer's ability to generalize the sudden environmental changes characteristic of non-stationary tasks.

Takeaways

- **Theory versus Practice in Adam:** As seen in Graph 1, the existence of performance drops followed by recovery suggests that the theory—Adam *can* cause policy collapse—is fundamentally correct. However, in practice, gradient clipping and trust-region methods (like those in PPO) act as a critical safety layer that prevents these theoretical risks from becoming catastrophic failures.
- **The Necessity of Curriculum Meta-Learning:** The contrast between the failure of direct training (Graphs 3,4) and the initial success of transfer (Graph 5) indicates that a curriculum approach is essential. One should first meta-train an optimizer in a stable environment to master physics before attempting to adapt it to dynamic or non-stationary environments.

- **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, 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. 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.

- **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 up- date 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.

- **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.