MACE-RL: Meta-Adaptive Curiosity-Driven Exploration with Episodic Memory in RL Environments

Ayush Singh, Shubh Saxena

Indian Institute of Technology Roorkee
Roorkee, Uttarakhand, India – 247667
ayush_s@mt.iitr.ac.in, shubh_s1@es.iitr.ac.in

Abstract

Effective exploration remains a fundamental challenge in Reinforcement Learning (RL), particularly in complex environments where rewards are sparse or misleading. Although curiosity-driven methods offer intrinsic motivation for exploration, they often rely on fixed heuristics that fail to adapt to the evolving requirements of a learning task. This raises a critical question: how can an agent learn how to explore? To address this, we introduce MACE-RL: Meta-Adaptive Curiosity-driven Exploration with Episodic Memory. MACE-RL is a novel framework that learns to dynamically regulate its own exploration strategy. It integrates a curiosity module, informed by episodic memory of past experiences, with a meta-learning network that monitors the agent's performance and adaptively regulates the influence of the intrinsic curiosity bonus. This enables the agent to achieve a dynamic balance between exploration and exploitation, becoming more exploratory when progress stalls and more focused when learning advances. We evaluate MACE-RL on a suite of control benchmarks, including Acrobot-v1, CartPole-v1, and HopperBulletEnv-v0. The results demonstrate that learning an adaptive exploration policy significantly outperforms both a strong PPO baseline and an ablated version with a fixed curiosity bonus, confirming that meta-adapting the exploration drive is crucial for effective learning.

Code — https://github.com/ayushsi42/mace_rl

Introduction

While intrinsic motivation is a powerful tool for exploration in reinforcement learning, current methods often rely on static heuristics to balance curiosity with exploitation. Approaches like the Intrinsic Curiosity Module (ICM) (Pathak et al. 2017), pseudo-counts (Bellemare et al. 2016), and episodic memory systems (Pritzel et al. 2017) have improved sample efficiency, but their inflexibility can be suboptimal as an agent's strategy must dynamically adapt. This raises a critical question: instead of using a fixed exploration strategy, *can an agent learn how to explore?*

To address this, we propose **MACE-RL (Meta-Adaptive Curiosity-driven Exploration with Episodic Memory)**, a framework that learns to dynamically regulate its own exploration. MACE-RL integrates a curiosity module, informed by an episodic memory of past experiences, with a meta-learning network that observes agent performance. By monitoring the agent's learning dynamics, the meta-network adaptively modulates the influence of the intrinsic curiosity bonus. This enables the agent to autonomously balance exploration and exploitation, increasing exploratory behavior when progress stagnates and shifting to exploitation once rewarding behaviors emerge. Our primary contribution is a reinforcement learning architecture that successfully meta-learns the exploration-exploitation trade-off, leading to a more sample-efficient and adaptive agent.

Methodology

The MACE-RL framework is constructed upon a PPO (Schulman et al. 2017) agent, a widely-used policy gradient method. The standard PPO agent is augmented with 3 specialized modules that collectively enable the agent to learn its own exploration strategy.

The first component is an **Episodic Memory** module, implemented as a Differentiable Neural Dictionary. This module maintains a finite-capacity buffer of past state-value pairs, serving as a long-term memory of the agent's experiences. By querying this memory, the agent can access historical context, which is crucial for distinguishing states that are truly novel from those that are merely revisited after a long interval. The memory employs a Least Recently Used (LRU) replacement policy to ensure that the stored experiences remain relevant to the agent's recent history.

The second component is a **Curiosity Module**, which generates an intrinsic reward signal, $R_{\text{curiosity}}$, to guide exploration. This intrinsic reward is a function of both the novelty of the current state and its relevance to past experiences retrieved from the Episodic Memory. State novelty is estimated using visitation counts for discrete state spaces and a k-NN density estimation for continuous spaces. The memory relevance component modulates this novelty score, allowing the agent to temper its curiosity for states that, while novel, are similar to previously encountered unproductive states. This mechanism produces a more nuanced exploration signal than novelty alone.

The core of our contribution lies in the third component, the **Meta-Adaptation Network**. The total reward provided to the agent is a weighted sum of the extrinsic and intrin-

Algorithm	Acrobot-v1	CartPole-v1	HopperBulletEnv-v0
PPO (Baseline)	186 / 617	140 / 660	865 / 1841
MACE-RL (Fixed β, $\beta = 0.4$, Exponential Scheduler)	201 / 595	120 / **620**	782 / 1801
MACE-RL (Full, Meta Network)	**249 / 586**	**104** / 634	**495 / 1799**

Table 1: Metrics reported are Sample Efficiency and Convergence Steps across three environments Acrobot-v1, CartPole-v1, HopperBulletEnv-v0. Lower is better for both metrics. Best results are in **bold**.

sic rewards, calculated as $R_{\text{total}} = R_{\text{extrinsic}} + \beta \times R_{\text{curiosity}}$. The meta-adaptation network, implemented as an LSTM, dynamically adjusts the parameter β. It processes sequences of exploration outcomes, such as reward patterns and curiosity effectiveness, as input. Based on this temporal analysis of agent performance, the LSTM outputs an adjusted value for β. This process constitutes a meta-learning loop where the agent learns an optimal, non-stationary schedule for its curiosity bonus, effectively learning when to prioritize exploration over exploitation. More on implementation details of each component in the supplementary.

Experimentation and Results

We evaluate the efficacy of MACE-RL across a suite of challenging continuous control environments. Its performance is benchmarked against a standard PPO implementation and an ablated version of our framework ("With fixed β") that utilizes a fixed, hand-tuned exponential decay schedule for the curiosity bonus. This experimental design allows us to isolate and quantify the benefits of meta-learning the exploration strategy. Further details on the experimental setup, hyperparameter configurations, and implementation specifics are provided in the supplementary material.

Our quantitative results, summarized in Table 1, highlight the advantages of the meta-adaptive approach. In both the *Acrobot-v1* and *HopperBulletEnv-v0* environments, the full MACE-RL agent with a meta-network achieves the best performance, demonstrating superior sample efficiency and faster convergence compared to both the PPO baseline and the fixed-schedule agent. Notably, in *HopperBulletEnv-v0*, the meta-adaptive agent is significantly more sample-efficient (495 episodes to reach threshold vs. 865 for PPO). The results in *CartPole-v1* present a more nuanced scenario: while our full MACE-RL agent is the most sample-efficient, learning faster than all other methods, the fixed-beta agent ultimately converges to a slightly higher final reward.

These findings are further supported by the learning curves shown in Figure 1. In *Acrobot-v1* and *HopperBulletEnv-v0*, the MACE-RL agent (Meta Network) not only learns faster but also reaches a higher level of average reward than the other methods. The learning curve for *CartPole-v1* visually confirms the trade-off observed in the quantitative results, where the meta-network agent exhibits the steepest initial learning curve, while the fixed-beta agent shows a steadier climb to its final performance level. Collectively, these results suggest that learning a dynamic exploration strategy is highly effective, leading to significant gains in learning speed and, superior final policy performance.

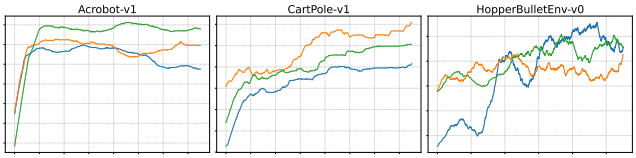

Figure 1: Learning curves for the PPO in blue, MACE-RL with a fixed β=0.4 in orange, and the full MACE-RL with a meta-network in green, evaluated on the Acrobot-v1, CartPole-v1, and HopperBulletEnv-v0 environments. The plots show the moving average of the extrinsic reward over the course of training episodes.

Future Scope

Future research direction could involve analyzing the emergent β schedules to distill fundamental principles of optimal exploration. These insights could guide the development of more advanced meta-learners capable of adapting not just a single curiosity parameter, but other richer facets of the exploration process.

Conclusion

In summary, MACE-RL uses a meta-learning network to adapt its exploration strategy by combining curiosity and episodic memory. This improves intrinsic motivation, sample efficiency, and overall performance, showing that meta-learning the exploration–exploitation trade-off is key for more autonomous RL agents.

Acknowledgement

The authors would like to thank E2E Networks for sponsoring the compute credits used in this work.

References

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, 1471–1479.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, 2778–2787. PMLR.

Pritzel, A.; Uria, B.; Srinivasan, S.; Puigdomenech Badia, A.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural episodic control. In *International Conference on Machine Learning*, 2827–2836. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Experimentation Details

All experiments were conducted using a unified framework to ensure reproducibility and fair comparison. The base reinforcement learning algorithm for all agents is Proximal Policy Optimization (PPO). The hyperparameters for the PPO agent were kept consistent across all experiments and are detailed in Table 2. All experiments were run with a consistent random seed to ensure reproducibility.

Table 2: PPO Hyperparameters

Hyperparameter	Value
Actor Learning Rate	0.0003
Critic Learning Rate	0.001
Discount Factor (γ)	0.99
PPO Epochs (K)	4
PPO Clip Parameter (ϵ)	0.2
Optimizer	Adam

The agents were evaluated on a suite of environments from the OpenAI Gym library, chosen to provide a diverse set of challenges:

- **Acrobot-v1**: A classic control problem with a 6-dimensional state space and a discrete action space of 3 actions. The goal is to swing a two-linked pendulum above a certain height.
- **CartPole-v1**: Another classic control task with a 4-dimensional state space and a discrete action space of 2 actions. The objective is to balance a pole on a cart.
- **HopperBulletEnv-v0**: A complex continuous control task from the PyBullet library. It features a 15-dimensional state space and a 3-dimensional continuous action space, requiring the agent to learn a hopping gait.

We compare three main algorithmic configurations:

1. **PPO (Baseline)**: The standard PPO algorithm without any MACE-RL components. This serves as our performance baseline.
2. **MACE-RL (Fixed Beta with Exponential Decay)**: This version of our agent uses a fixed schedule for the curiosity bonus weight, β. The scheduler follows an exponential decay pattern, starting from an initial β value and decreasing over time. We test initial β values of 0.2 and 0.6.
3. **MACE-RL (Full Meta Network)**: This is the complete MACE-RL agent, which employs a meta-learning network to dynamically adapt the β parameter. The meta-adaptation network, an LSTM, observes sequences of the agent's experiences and outputs an adjusted β value for each episode, allowing the agent to learn its own exploration strategy.

Technical Implementation of MACE-RL Components

The MACE-RL framework is composed of three main modules that augment the base PPO agent. The following sections detail their technical implementation.

Episodic Memory

The Episodic Memory module functions as a Differentiable Neural Dictionary, designed to store and retrieve past experiences in a key-value format. The keys are state embeddings, and the values are tensors containing information about the outcome of an action, such as the subsequent state and the reward received.

The memory has a fixed capacity and employs a Least Recently Used (LRU) replacement policy. When the memory is full, the least recently accessed item is overwritten with the new experience. The module maintains a usage tensor to track the recency of each memory slot. When a memory is queried, its usage count is incremented, ensuring that frequently accessed memories are retained. The query mechanism retrieves the top-k most similar memories to a given query state, based on the Euclidean distance between the query state and the keys stored in memory.

Curiosity Module

The Curiosity Module is responsible for generating the intrinsic reward signal. This module integrates signals from the Episodic Memory to produce a context-aware curiosity bonus. The core of this module is the calculation of a novelty score, which is then modulated by a memory relevance score.

For continuous state spaces, the novelty is estimated using an ensemble of k-Nearest Neighbors (k-NN) models, combined with a neural density estimator. This hybrid approach provides a robust measure of novelty that is less susceptible to noise. For discrete state spaces, a count-based approach is used, where the novelty of a state is inversely proportional to its visitation count. A temporal decay factor is applied to the visitation counts to give more weight to recent experiences. The memory relevance score is calculated by querying the Episodic Memory for states similar to the current state. The final curiosity bonus is a product of the novelty and memory relevance scores, encouraging the agent to explore states that are not only novel but also distinct from past unproductive experiences.

Meta-Adaptation Network

The Meta-Adaptation Network is the central component of MACE-RL and learns to dynamically adjust the curiosity weight, β, which balances the influence of the intrinsic and extrinsic rewards. The network architecture is based on a Long Short-Term Memory (LSTM) model, which is well-suited for processing sequential data.

The input to the Meta-Adaptation Network is a sequence of the agent's experiences from the preceding episode, including states, actions, and rewards. The LSTM processes this sequence to capture temporal patterns in the agent's performance. The output of the network is a single value that

determines the β for the next episode. This allows the agent to learn a dynamic schedule for its curiosity, increasing it when performance stagnates and decreasing it when it is successfully exploiting its knowledge. The network is trained with its own optimizer, separate from the main PPO agent, forming a meta-learning loop where the agent learns how to best guide its own exploration.

Detail of Evaluation Metrics

To provide a comprehensive and fair comparison of the different agents, we use two primary evaluation metrics: **Sample Efficiency** and **Convergence Steps**. These metrics are designed to capture not only the final performance of an agent but also the speed and efficiency with which it learns.

- **Sample Efficiency**: We define sample efficiency as the number of episodes required for an agent to reach a certain performance threshold. This threshold is defined as 75% of the final average performance of the agent in the last 10% of training episodes. A lower value for sample efficiency indicates that the agent learns more quickly and requires fewer interactions with the environment to achieve a competent policy.
- **Convergence Steps**: This metric measures the number of episodes it takes for the agent's performance to stabilize, indicating that it has converged to a stable policy. We determine this by finding the point in the training process where the slope of the moving average of the reward curve becomes minimal, suggesting that further training yields diminishing returns. A lower number of convergence steps implies that the agent reaches its final performance level faster.