Neural Networks 188 (2025) 107448

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Self-Referencing Agents for Unsupervised Reinforcement Learning

Andrew Zhao [a,1], Erle Zhu [b,1], Rui Lu [a], Matthieu Lin [b], Yong-Jin Liu [b], Gao Huang [a,*]

[a] Department of Automation, BNRist, Tsinghua University, China
[b] Department of Computer Science, BNRist, Tsinghua University, China

ARTICLE INFO

Keywords:
Reinforcement learning
Unsupervised reinforcement learning
Pretraining
Finetuning

ABSTRACT

Current unsupervised reinforcement learning methods often overlook reward nonstationarity during pre-training and the forgetting of exploratory behavior during fine-tuning. Our study introduces Self-Reference (SR), a novel add-on module designed to address both issues. SR stabilizes intrinsic rewards through historical referencing in pre-training, mitigating nonstationarity. During fine-tuning, it preserves exploratory behaviors, retaining valuable skills. Our approach significantly boosts the performance and sample efficiency of existing URL model-free methods on the Unsupervised Reinforcement Learning Benchmark, improving IQM by up to 17% and reducing the Optimality Gap by 31%. This highlights the general applicability and compatibility of our add-on module with existing methods.

1. Introduction

Unsupervised Reinforcement Learning (URL) is an attempt at replicating the success of the pre-train-fine-tune framework used in computer vision (CV) and natural language processing (NLP) (Laskin et al., 2021). It involves two stages: pre-training (PT) and fine-tuning (FT). During PT, the agent explores and understands the environment through designed intrinsic rewards. In FT, the agent utilizes extrinsic rewards to tackle specific tasks, leveraging the pre-trained policy for efficient adaptation to the downstream task. The goal of URL is to gather extensive environment transition dynamics knowledge during PT, independent of task-specific rewards, and efficiently apply this knowledge during FT to tackle downstream tasks. This framework has been studied in various existing works (Burda, Edwards, Storkey, & Klimov, 2018; Eysenbach, Gupta, Ibarz, & Levine, 2018; Laskin et al., 2022; Liu & Abbeel, 2021a, 2021b; Liu, Chen, & Zhao, 2023; Pathak, Agrawal, Efros, & Darrell, 2017; Pathak, Gandhi, & Gupta, 2019; Yarats, Fergus, Lazaric, & Pinto, 2021; Zhao, Lin, Li, Liu, & Huang, 2022). Existing unsupervised reinforcement learning algorithms typically use a measure of surprise as a reward function to explore the environment during pre-training (Zhao et al., 2022). This reward function should decrease rewards for frequently visited states and increase rewards for less visited states, making it implicitly dependent on the agent's history. However, most popular URL algorithms do not model this dynamic change during PT, leading to a nonstationary Markov Decision Process (MDP) and potential instability or sub-optimality in learning (Choi, Yeung, & Zhang, 1999; Laskin et al., 2021). Furthermore, the naive pre-train-then-fine-tune paradigm can result in unlearning exploratory behaviors during fine-tuning, as the gradients from the fine-tuned task quickly overwrite the parameters of the pre-trained policy, with no incentives to retain the pre-trained behaviors during the naive finetuning process (Campos et al., 2021; Wolczyk et al., 2023).

Inspired by the success of retrieval-augmented reinforcement learning and building on the above observations, we introduce a novel add-on module called Self-Reference (SR) for unsupervised reinforcement learning, specifically designed to harness historical information more effectively during training. Our approach enhances both performance and efficiency through the pre-training and fine-tuning phases by continually presenting the agent with historical visited states at every decision-making epoch. SR augments the agent's state with information to generate summary statistics of the states visited during pre-training, allowing it to explicitly model the dynamic changes in the reward function. Additionally, SR also helps preserve the exploratory behaviors learned during pre-training by explicitly showing the agent past behaviors. Fig. 1 provides a schematic depiction of the Self-Reference method.

We evaluate SR on the standard Unsupervised Reinforcement Learning Benchmark (Laskin et al., 2021) and achieve state-of-the-art results for model-free methods by applying Self-Reference to RND (Burda et al., 2018). Our module increases the IQM of APS (Liu & Abbeel, 2021a) and ProtoRL (Yarats et al., 2021) by up to 17% while decreasing

* Corresponding author.
E-mail addresses: zqc21@mails.tsinghua.edu.cn (A. Zhao), zel20@mails.tsinghua.edu.cn (E. Zhu), r-lu21@mails.tsinghua.edu.cn (R. Lu), lyh21@mails.tsinghua.edu.cn (M. Lin), liuyongjin@tsinghua.edu.cn (Y.-J. Liu), gaohuang@tsinghua.edu.cn (G. Huang).
[1] Equal contribution.

https://doi.org/10.1016/j.neunet.2025.107448
Received 7 August 2024; Received in revised form 15 March 2025; Accepted 27 March 2025
Available online 5 April 2025

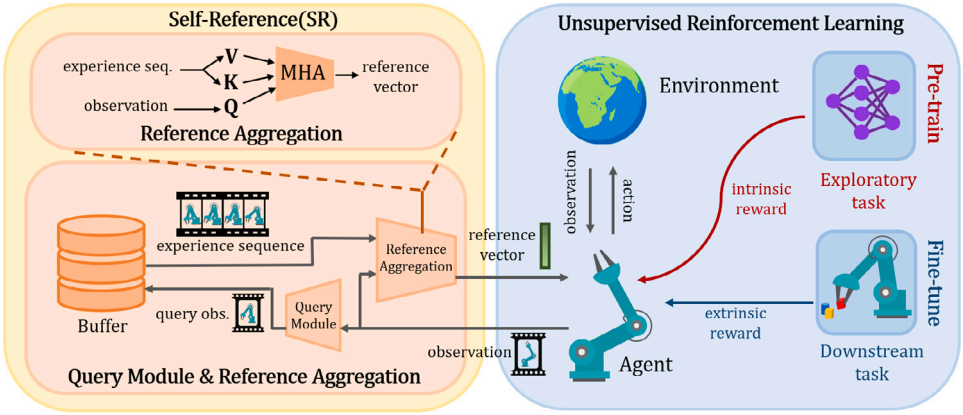

Fig. 1. Unsupervised reinforcement learning with self-reference (SR). The schematic illustration of our method. Given a state, Self-Reference forms a query to the buffer and receives top k neighbors and their subsequent D states. Then, SR aggregates the transitions into a reference vector that aids the agent in learning under the URL setting.

the Optimality Gap (OG) by an average of 11% and up to 31% for ProtoRL. Additionally, Self-Reference improves sample efficiency, allowing the agent to reach asymptotic performance with fewer pre-training steps. By addressing key challenges in URL, our method complements existing methods and has practical implications for deploying efficient, robust robotic systems in real-world scenarios.

2. Related works

2.1. Unsupervised reinforcement learning

Unsupervised reinforcement learning methods consist of pre-training and fine-tuning phases. During pre-training, the policy leverages intrinsic rewards (Bellemare et al., 2016; Zhao et al., 2025), such as maximum state entropy methods (Guo et al., 2021; Hazan, Kakade, Singh, & Soest, 2019; Jain, Lehnert, Rish, & Berseth, 2023; Kim, Shin, Abbeel, & Seo, 2023; Lee et al., 2019; Mutti, 2023; Mutti, Mancassola and Restelli, 2022; Mutti, Pratissoli, & Restelli, 2021; Mutti & Restelli, 2020; Mutti, Santi and Restelli, 2022; Nedergaard & Cook, 2022; Seo et al., 2021; Tiapkin et al., 2023; Yang & Spaan, 2023; Zamboni, Cirino, Restelli, & Mutti, 2024a, 2024b; Zhang, Cai, Huang, & Li, 2021; Zisselman, Lavie, Soudry, & Tamar, 2023), to learn exploratory policies and develop generally useful behaviors in the environment. During fine-tuning, the policy is optimized using extrinsic rewards, with the pre-trained policy serving as an initialization. This paradigm accelerates adaptation to new tasks with improved sample efficiency. Laskin et al. (2021) provide a comprehensive summary and detailed implementation of mainstream unsupervised reinforcement learning (URL) algorithms. For a thorough explanation of the baselines used in this paper, we refer readers to their work.

2.2. Retrieval-augmented techniques in machine learning

Information retrieval is common in machine learning, helping to reduce memorization in parametric weights, as seen in Large Language Models (Guu, Lee, Tung, Pasupat, & Chang, 2020), Multi-Modal Modeling (Chen, Hu, Saharia, & Cohen, 2022), Decision-making (Zhao et al., 2024), and Time Series Forecasting (Jing et al., 2022). Retrieval-augmented systems improve neural network performance and generalization while enhancing training efficiency. Beyond reducing memorization, we apply retrieval to address nonstationarity and prevent the forgetting of exploratory behaviors during pre-training and fine-tuning.

Traditional imitation learning struggles with scalability due to high data supervision and weak generalization. Leveraging prior task data enhances robustness and efficiency (Nasiriany, Gao, Mandlekar, & Zhu, 2022). In reinforcement learning, retrieval techniques use a data buffer (e.g., replay buffer (Goyal et al., 2022), external datasets (Humphreys et al., 2022), or data from other agents (Nasiriany et al., 2022)). These systems select top-k nearest neighbors using multi-head attention or specific fusion networks (Humphreys et al., 2022). We introduce a query module that improves adaptability by using learned queries and leveraging the agent's own historical trajectories as queryable data.

2.3. Catastrophic forgetting in transferring policy

Fine-tuning a pre-trained policy can lead to catastrophic forgetting, where the agent forgets exploratory skills due to new learning signals (Campos et al., 2021; Wolczyk et al., 2023). Behavior Transfer (BT) addresses this by using a pre-trained policy for exploration, enhancing downstream policy performance (Campos et al., 2021). Our approach mitigates this issue by replaying snapshots of old behaviors at each decision epoch.

3. Preliminaries and notations

We follow the standard reinforcement learning assumption that the system can be described by a Markov Decision Process (MDP) (Sutton & Barto, 2018). An MDP is represented by a tuple $(\mathcal{S}, \mathcal{A}, p, r, \mathcal{S}_0, \gamma)$, which has states space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, reward function $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, $\mathcal{S}_0$ initial state distribution and discount factor γ. At each time step, an agent observes the current state $\mathbf{S} \in \mathcal{S}$, performs an action $\mathbf{A} \in \mathcal{A}$, and then observes a reward and next state $\mathbf{R} = r(\mathbf{s}, \mathbf{a}), \mathbf{S}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ from the environment. Generally, the reward r consists of an extrinsic component r^{ext}, provided by the environment, and an intrinsic component r^{int}, generated by the agent using URL algorithms. Thus, the final additive reward can be expressed as $r = r^{\text{ext}} + r^{\text{int}}$, where r^{int} can be zero if no intrinsic reward is present.

An RL algorithm aims to train an agent to interact with the environment and maximize the expected discounted return. Most popular RL algorithms use the value function V_π as the optimization objective. It represents the expected return when following the parameterized policy π_θ starting from a sampled initial state: $V_{\pi_\theta} = \mathbb{E}_{\pi_\theta, s\sim S_0}[G|\mathbf{S} = \mathbf{s}]$, where G is the discounted sum of rewards.

Popular deep RL algorithms excel in many decision-making tasks but struggle with generalization to other tasks. URL approaches train agents with self-supervised rewards to develop more generalizable policies for efficient adaptation to downstream tasks (Laskin et al., 2021). The Unsupervised Reinforcement Learning Benchmark (URLB) evaluates URL algorithms by splitting training into two phases: pre-training and fine-tuning. Agents are trained with intrinsic rewards for N_{PT} steps during pre-training and evaluated based on performance after fine-tuning with extrinsic rewards for N_{FT} steps.

Suppose we express the history of transitions obtained during training as $\Lambda_g = \{\mathbf{s}_0^{\pi_0}, \mathbf{a}_0^{\pi_0}, \ldots, \mathbf{s}_H^{\pi_0}, \mathbf{a}_H^{\pi_0}, \ldots, \mathbf{s}_0^{\pi_g}, \mathbf{a}_0^{\pi_g}, \ldots \mathbf{s}_H^{\pi_g}, \mathbf{a}_H^{\pi_g}\}$, where π_g indicates the changing training policy indexed by episode g and H is the

finite number of decision epochs in an episode. We can express the PT reward as $r(\mathbf{s}, \mathbf{a}; \Lambda_g)$, which implicitly depends on this history of transitions produced during pre-training. Current methods largely ignore the dependence on the history Λ and train agents with only states and actions. Even though these methods obtain satisfactory empirical results, we posit that URL methods could be further improved if we address this issue.

4. Self-reference

The unsupervised reinforcement learning (URL) paradigm encompasses two distinct phases: pre-training and fine-tuning. During the pre-training phase, the reward function's implicit dependence on a history of transitions results in a non-stationary Markov decision process, provided the agent does not explicitly account for the evolving reward structure. Based on this observation, we strategically retrieve relevant historical data to address the challenges posed by nonstationarity.

Furthermore, works like (Humphreys et al., 2022) have shown that referencing expert demonstrations benefits learning their behavior. In the URL setting, we observe that pre-trained agents with exploratory behaviors can better transfer to downstream tasks during the fine-tuning phase (Laskin et al., 2021). We posit that retrieving the exploratory trajectories from the pre-training phase can further enhance the model's transfer performance during fine-tuning.

Equipped with the intuition that explicitly showing the agent its past experiences as references could benefit it in the URL setting both during pre-training and fine-tuning, we devised a new module for unsupervised reinforcement learning algorithms called Self-Reference (SR). We designed this module as an add-on method that can boost any unsupervised reinforcement learning method's performance. The Self-Reference module, featuring a query system and an aggregation network, augments the agent's state by integrating essential information from past experiences. Readers can find a schematic figure of Self-Reference in Fig. 1. Next, we outline the details of the query module system and the aggregation network in detail.

4.1. The query module

In order to query informative trajectories from historical experiences, we need to develop a module that can retrieve appropriate values that could be useful for the unsupervised reinforcement learning agent. We argue that the query module should have the flexibility to query the following trajectories:

- *Neighbors.* Intuitively, neighbors can show where the agent has been in close proximity and infer where to explore more.
- *Neighbors in partial dimensions.* In some cases, we only need to select reference neighbors with particular features and explore different parts of the feature space.
- *Information on possible future visitations.* Since the agent needs to maximize the state-value function, it needs to be forward-seeking and have the option to look at other states that the agent might wish to visit.

To enhance the system's adaptability, we developed a module capable of determining the optimal queries. This module, denoted as $\pi_\phi^{\text{query}} : S \to S$, takes the current state as input and queries historical data to support the URL agent's decision-making process during training. For optimization, we employed the well-known on-policy algorithm PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017), aiming to maximize the task reward shared with actor model. This encompasses the intrinsic reward during the pre-training phase and the task reward throughout the fine-tuning stage. Our approach keeps the hyperparameters consistent with those specified in the CleanRL (Huang et al., 2022) framework. To further refine our system, based on the notion that querying for nearest neighbors can serve as a beneficial inductive bias (Goyal et al., 2022; Humphreys et al., 2022), we introduced an identity loss $\mathcal{L}_{\text{identity}} = \|\mathbb{E}_{\mathbf{q}\sim\pi_\phi^{\text{query}}(\mathbf{s})}[\mathbf{q}] - \mathbf{s}\|_2$ to the query actor, where $\mathbf{q} \sim \pi_\phi^{\text{query}} \in S$ represents the generated query. The total loss for the query actor is

$$\mathcal{L}_{\pi_\phi^{\text{query}}} = \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{identity}} \tag{1}$$

To further facilitate this inductive bias, we do not use the query module for half of the episodes during the PT phase and use the current state s_t as the query. We do not update the query module's parameters on the trajectories that simply use s_t as the query. The value function of the query module, V_ρ^{query} is trained using the standard PPO value loss.

After obtaining the query, we use a vectorstore to find the top k nearest neighbors (keys) of that query and retrieve all subsequent states within D timesteps. We hypothesize that expanding the retrieved states to trajectories provides more information like rolling out in planning algorithms (See for confirmation). From this retrieval process, we obtain a set E defined as

$$E = \{(\mathbf{s}_{t_1} : \mathbf{s}_{t_1+D-1}), \ldots, (\mathbf{s}_{t_k} : \mathbf{s}_{t_k+D-1})\}, \tag{2}$$

which contains k ordered lists, each containing D subsequent states of respective k key states. The subscript under t indicates the kth neighbor for the query. With this query module trained, we obtained a viable way to query the historical states for additional knowledge under appropriate situations. We will leave retrieving the action and reward from historical trajectories as future work.

Since Self-Reference agents need to retrieve historical states, we must have a component that maintains historical information. Fortunately, off-policy reinforcement learning algorithms often already possess a large experience replay buffer $\mathcal{B}$ in order to train the RL agent; therefore, we can query from this buffer without additional memory overhead. As the agent needs to query historical information exhaustively at every step, we propose using a subset of the experience replay buffer as retrievable historical experiences for SR agents: $\mathcal{B}^{\text{reference}} \subset \mathcal{B}$. In all of our experiments, the number of retrievable states is set to the $|\mathcal{B}^{\text{reference}}| = 1e5$ latest transitions unless stated otherwise ($|\mathcal{B}| = 1e6$ for comparison). We keep the context window size $|\mathcal{B}^{\text{reference}}|$ capped at the agent's last episode's experiences, avoiding the possibility of the agent querying a state from the current episode.

We utilized `Faiss` (Johnson, Douze, & Jégou, 2019) as our retrieval library which is a scalable method to efficiently employ GPUs when performing similarity searches. We maintained the key-query space identical to the state space because it already contains all the information needed for decision-making. Moreover, cosine similarity was used as a metric when performing a k-NN search, as we found that it slightly outperformed the norm l-2 in our experiments.

4.2. Aggregating retrieved experiences

In our model, historical states retrieved from the replay buffer are integrated into a unified feature vector, which we name as reference vector. This vector serves as input for the critic and actor networks. We adopt a multiheaded cross-attention mechanism to dynamically combine these experiences, leveraging the current state hidden feature as the query. It is important to note that this query is distinct from the one utilized to query the replay buffer. The features extracted from the experiences E (referenced in Eq. (2)) are used as keys and values in this attention framework.

Additionally, we enrich the key and value features with learnable time step embeddings. These embeddings signify the temporal sequence of events, suggesting that each transition spanning D steps carries specific temporal information. This aspect is crucial as it allows the model to update these embeddings through the RL learning objective, enhancing the model's temporal awareness.

The computation of the reference vector, which serves as an enriched historical context for the agent during training, is detailed

below. Here, U represents both the embedding dimension of the Multi-HeadAttention (MHA) (Vaswani et al., 2017) and the dimension of the reference vector. Query of MHA, where s_t is the current state:

$$Q = \text{QEncoder}(s_t) \tag{3}$$

Keys and Values from experiences, where $\{s_{t_j:t_{j+D-1}}\}_{j=1}^{k}$ are elements of E:

$$K = \text{KEncoder}(s_{t_1} : s_{t_1+D-1}, \dots, s_{t_k} : s_{t_k+D-1}) \tag{4}$$

$$V = \text{VEncoder}(s_{t_1} : s_{t_1+D-1}, \dots, s_{t_k} : s_{t_k+D-1}) \tag{5}$$

Time Embedding for transitions of D steps:

$$T = \text{TimeEmbed}(t_1 : t_1 + D - 1, \dots, t_k : t_k + D - 1) \tag{6}$$

Reference Vector, integrating all components:

$$u_t = \text{MultiHeadAttention}(Q, K + T, V + T) \tag{7}$$

Finally, the reference vector is concatenated with the actor network's state feature and the critic network's state and action feature. With the added computation of the SR module, we noticed a mean training Frames-Per-Second decrease from 17 to 15, which is relatively marginal due to efficient implementations. The entire algorithm is formalized in Algorithm 1.

Algorithm 1 Unsupervised Reinforcement Learning with Self-Reference

Input: Randomly initialized actor π_θ, critic Q_ϕ, and encoder f_ξ networks, replay buffer $\mathcal{B}$.
Input: Intrinsic r^{int} and extrinsic r^{ext} reward functions, discount factor γ.
Input: Environment (env), H maximum number of steps in episode, M downstream tasks T_k, $k \in [1, ..., M]$.
Input: pre-train N_{PT} and fine-tune N_{FT} steps.
Input: Randomly initialized query actor π_ϕ^{query}, critic V_ρ^{query}, reference buffer $\mathcal{B}^{\text{reference}}$, `GetTrajectories`, numbers of neighbors to retrieve K and `NearestNeighborSearch`.
for $t = 1$ **to** N_{PT} **do**
  $\mathbf{q}_t \sim \pi_\phi^{\text{query}}(\mathbf{o_t})$
  $\mathbf{I}_t \leftarrow \texttt{NearestNeighborSearch}(\mathbf{q}_t, \mathcal{B}^{\text{reference}}, K)$
  $\mathcal{T}_t \leftarrow \texttt{GetTrajectories}(\mathbf{I}_t, \mathcal{B}^{\text{reference}})$
  $a_t \leftarrow \pi_\theta(f_\xi(\mathbf{o}_t), \mathcal{T}_t) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$
  $\mathbf{o}_{t+1} \sim P(\cdot|\mathbf{o}_t, \mathbf{a}_t)$
  $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$
  if $t \mod H = 0$ **then**
    $\mathcal{B}^{\text{reference}} \leftarrow \mathcal{B}^{\text{reference}} \cup \{\mathbf{o}_{t-H}, ..., \mathbf{o}_{t+1}\}$
  end if
  Update π_θ, Q_ϕ, and f_ξ using minibatches from $\mathcal{B}$ and intrinsic reward r^{int} using DDPG;
  Update π_ϕ^{query}, V_ρ^{query} using minibatches from $\mathcal{B}$ according to Eqs. (1).
end for
for $T_k \in [T_1, ..., T_m]$ **do**
  initialize $\theta \leftarrow \theta_{\text{PT}}, \phi \leftarrow \phi_{\text{PT}}, \xi \leftarrow \xi_{\text{PT}}$, reset $\mathcal{B}$.
  initialize $\psi \leftarrow \psi_{\text{PT}}, \rho \leftarrow \rho_{\text{PT}}$, $\mathcal{B}^{\text{reference}} \leftarrow \mathcal{B}^{\text{reference}}_{\text{PT}}$.
  for $t = 1$ **to** N_{FT} **do**
    $\mathbf{q}_t \sim \pi_\phi^{\text{query}}(\mathbf{o_t})$
    $\mathbf{I}_t \leftarrow \texttt{NearestNeighborSearch}(\mathbf{q}_t, \mathcal{B}^{\text{reference}}, K)$
    $\mathcal{T}_t \leftarrow \texttt{GetTrajectories}(\mathbf{I}_t, \mathcal{B}^{\text{reference}})$
    $a_t \leftarrow \pi_\theta(f_\xi(\mathbf{o}_t), \mathcal{T}_t) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$
    $\mathbf{o}_{t+1} \sim P(\cdot|\mathbf{o}_t, \mathbf{a}_t)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{o}_t, \mathbf{a}_t, r_t^{\text{ext}}, \mathbf{o}_{t+1})$
    if $t \mod H = 0$ **then**
      $\mathcal{B}^{\text{reference}} \leftarrow \mathcal{B}^{\text{reference}} \cup \{o_{t-H}, ..., \mathbf{o}_{t+1}\}$
    end if
    Update π_θ, Q_ϕ, and f_ξ using minibatches from $\mathcal{B}$ using DDPG;
    Update π_ϕ^{query}, V_ρ^{query} using minibatches from $\mathcal{B}$ by according to Eqs. (1).
  end for
  Evaluate performance of RL agent on task T_k
end for

5. Experiments

In this section, we present the benchmarks, evaluation metrics, and main results, along with the results from pre-training.

5.1. Benchmark and evaluation

We apply our method to existing unsupervised reinforcement learning algorithms and evaluate these new methods on tasks from the URL Benchmark (URLB) (Laskin et al., 2021), a standard evaluation suite in URL research.

Evaluation. For our main results, we follow the *URLB's* training procedure by pre-training the agent for two million steps in each domain with intrinsic rewards and fine-tuning the pre-trained agent for one hundred thousand steps with downstream task rewards. All baseline experiments were conducted for eight seeds *(0–7)* per downstream task for each algorithm using the code from the URL Benchmark. Furthermore, *SR* applied to vanilla URL algorithms are also evaluated for eight seeds per task. A total of $1920 = 2$ (use SR or not) $\times 10$ algorithms $\times 12$ tasks $\times 8$ seeds experiments were conducted for the main results. All methods use the DDPG (Lillicrap et al., 2016) agent as their backbone. DDPG is an off-policy, model-free reinforcement learning algorithm designed for continuous action spaces. It leverages an actor–critic architecture, where the actor π_θ maps states to deterministic actions, and the critic Q_ϕ estimates the action-value function using the Bellman equation:

$$Q_\phi(s_t, a_t) = r_t + \gamma Q_{\phi\prime}(s_{t+1}, \pi_{\theta\prime}(s_{t+1}))$$

where $\phi\prime$ and $\theta\prime$ are target network parameters that stabilize training. The policy is updated via the deterministic policy gradient.

We use the scores of a DDPG agent trained from scratch for two million steps as the expert scores and normalize the original task scores. To make the main results more convincing, we use statistical-based metrics *interquartile mean (IQM)* and *Optimality Gap (OG)* of normalized scores as our main evaluation criteria, with mean values and median values as references respectively. *IQM* is more unbiased than the median, while *Optimality Gap* is the distance between a method's score and expert scores. We present all results with error bars using standard error.

5.2. Main results

The main results in URLB are presented in Fig. 3; we separate the URL algorithms into three categories and present their before and after performance using SR. All URL algorithms have obtained a 5% improvement in *IQM* and an 11% reduction in *OG* on average. To our surprise, APS+SR achieves a 17% higher *IQM* than vanilla APS, and ProtoRL+SR obtains a 31% lower *OG* than vanilla ProtoRL. We found that after using SR, data-based methods achieved an 8.5% higher *IQM* and a 19% lower *OG* than before on average, representing the best progress among the three categories. We did notice that DIAYN and SMM dropped in performance after adding Self-Reference. We hypothesize that since DIAYN suffers from a lack of exploration (Strouse, Baumli, Warde-Farley, Mnih, & Hansen, 2021), and SR tends to accelerate convergence due to alleviating the nonstationarity problem, in the case of DIAYN, SR could impede exploration even more, resulting in worse performance. While we observe a compelling distributional improvement across tasks within any given domain and for most algorithms, certain algorithm-domain-task combinations remain comparable to the baselines. Lastly, since our method includes a newly trained module that is updated alongside the main policy, stability is often a concern in systems with many moving parts and modules trained end-to-end (Glasmachers, 2017). Fortunately, we have closely examined stability by comparing SR with its vanilla counterpart and observed minimal differences in gradient norms, as well as the variance of intrinsic and extrinsic rewards across seeds. This suggests that the additional training modules introduced by SR do not have a significant negative impact on stability during training.

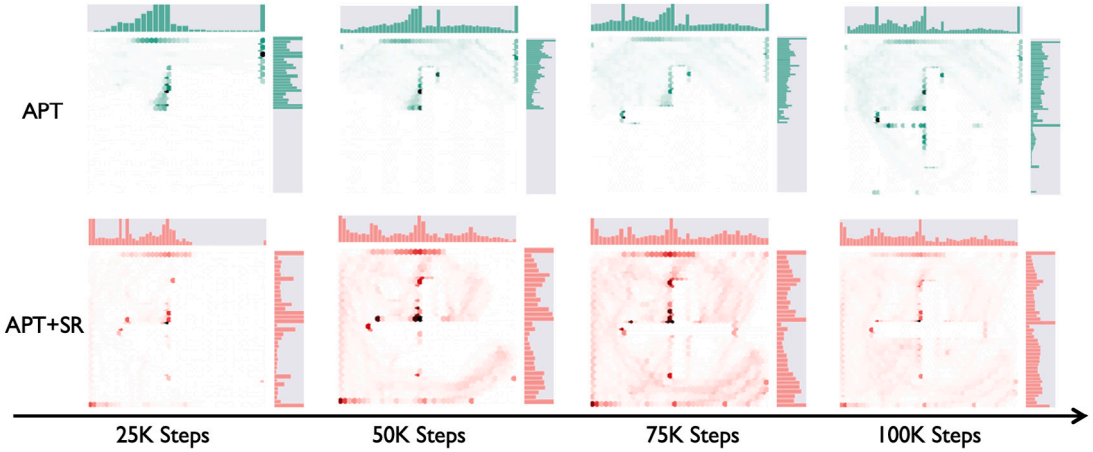

Fig. 2. Visitation jointplot. A plot of visitations as density points as PT steps increase, with margin histograms labeled with normalized probability. APT+SR (red) was able to cover the Y-axis in 25k steps and the whole X–Y plane in only 50k steps, while Vanilla APT (green) lags behind and only starts to cover the X–Y plane at 100k PT steps. Note that the empty cross-shape in the middle is formed due to an impenetrable wall that the agent cannot pass through.

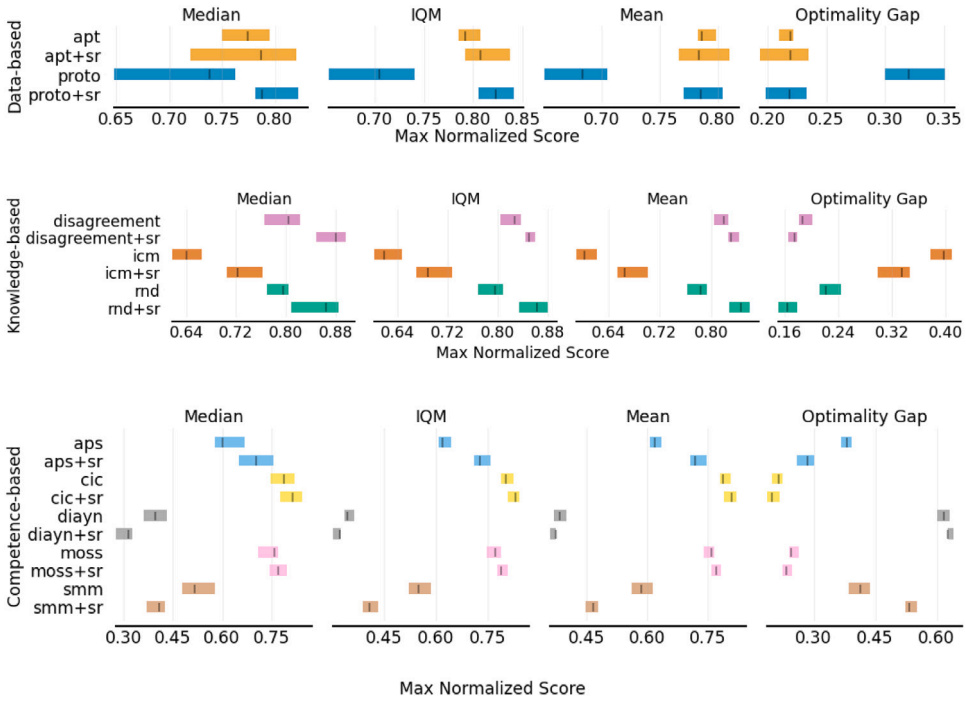

Fig. 3. Main results of self-reference on unsupervised reinforcement learning benchmark. We showcase our main results, reported using RLiable with statistically robust metrics. Our main results emphasize on the IQM and Optimality Gap metrics and provide Median and Mean for reference purposes.

5.3. Pre-train phase results

To better understand how our method behaves differently in the pre-training phase alone, we conducted experiments in the point mass maze environment to demonstrate that obtaining information from the history of transitions helps agents learn more efficiently. The point mass maze environment lets the agent control a ball in a 2D plane with a cross-shaped wall in the center where the ball cannot go through. We choose the APT method in this experiment since the intrinsic reward is the entropy of the state visitation, which is greatest when the state visitation is uniformly distributed and can be more intuitively visualized. We train APT and APT+SR for 100k steps and plot the visitation joint-plot in Fig. 2. The plot shows that the APT agent struggles to get out of the left top quadrant pre-25k steps while APT+SR already covers the Y-axis evenly. Then, at 50k steps, the APT+SR agent covers the X–Y plane while the APT agent struggles to visit the lower half of the plane. Finally, at 100k steps, APT+SR achieves a relatively uniform coverage while vanilla APT only started exploring the lower half of the plane. Overall, this result demonstrates that the APT+SR agent covers the X–Y plane much faster than vanilla APT, demonstrating the Self-Reference module's efficacy in learning from the nonstationary task reward of changing state-space coverage.

6. Ablation and empirical analysis

PT-FT sample efficiency. First, we demonstrate how our method can boost sample efficiency under the PT/FT framework. Since RND+SR achieved SOTA performance on URLB, we will use RND as the intrinsic reward and Quadruped as the test domain for the remainder of this Section 6 with three seeds to account for variability. For this experiment, we constrain the pre-training steps RND and RND+SR to fewer steps and show the FT performance in Fig. 4. The plot at

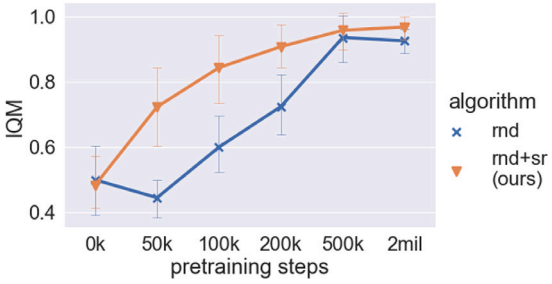

Fig. 4. Pre-training steps vs. performance. To illustrate the efficiency of our method, we pre-train agents with fewer steps and plot the FT performances in the Quadruped domain. We observe that RND+SR quickly rises in performance while vanilla RND struggles to perform high in fewer pre-training steps scenarios.

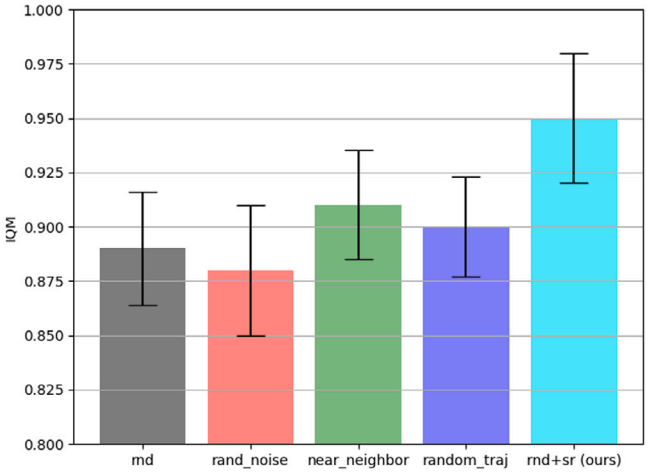

Fig. 5. Ablation on reference vector. We compare SR's query module to different hand-crafted ways of constructing the reference vector. Noticeably, if we only retrieve neighbors based on the current state or randomly, it performs worse than with our query module on the downstream task.

0k steps shows DDPG vs. DDPG with Self-Reference, essentially relegating to a supervised RL scenario. It is evident that training from scratch (pre-train steps = 0k) with the Self-Reference module yields no significant gains. This suggests that the incorporation of a retrieval mechanism alone does not substantially aid in the supervised reinforcement learning scenario. However, the development of a robust exploratory policy from PT, coupled with the prevention of forgetting, is essential for achieving performance improvements. Furthermore, in the 50k and 100k pre-train steps scenarios, vanilla RND increased its performance slowly (even decreased in performance with 50k PT steps). At the same time, RND with Self-Reference quickly achieved a high score across Quadruped tasks with a normalized IQM of 0.83 in just 100k pre-training steps. We believe the increase in sample efficiency with the Self-Reference module stems from its ability to mitigate the non-stationary reward problem, which enables the model to learn exploration rewards more effectively, thereby improving the agent's ability to explore the state space more efficiently.

Efficacy and analysis of query module. We additionally demonstrate how different types of reference vectors affect agent performance and the effectiveness of our query module. Besides our method of using a query actor to select references, we devised three *hand-crafted* approaches for selecting references: (1) combining features from the nearest neighbors of the current state s_t and their subsequent D states, (2) combining features from uniformly sampled states and their subsequent D states, and (3) setting the reference vector $\sim \mathcal{N}(0, 1)$. We present the aggregated results for the Quadruped domain in Fig. 5.

Table 1
Comparison with (Campos et al., 2021) and freezing SR module. We compare with the baseline method (Campos et al., 2021) (BT) and also investigate the disentangled efficacy of SR during pre-training and fine-tuning.

Method	Flip	Run	Stand	Walk
RND	698 ± 23	369 ± 19	946 ± 13	$\mathbf{854 \pm 29}$
RND+BT	563 ± 16	452 ± 8	907 ± 47	821 ± 11
RND+Freeze SR PT	672 ± 22	453 ± 22	946 ± 4	828 ± 2
RND+Freeze FT SR	700 ± 19	442 ± 20	948 ± 4	824 ± 21
RND+SR	$\mathbf{717 \pm 29}$	$\mathbf{476 \pm 23}$	$\mathbf{955 \pm 10}$	832 ± 40

Utilizing the nearest neighbor of the current state or retrieving random trajectories proves somewhat helpful, illustrating how explicitly providing the agent with the history of transitions is beneficial. Moreover, concatenating random noise to the agent features, essentially denoising does not significantly impact performance. Overall, SR achieves the best result, emphasizing the query module's flexibility, which can either mimic the nearest neighbor queries by learning an identity function, attempt to match random sampling by outputting high entropy queries or query the replay buffer in any other manner beneficial for enhancing the agent's performance.

To further investigate the query module's functionality, we visualize the query module outputs during PT and FT phases in point mass maze environment. Current observations (i.e., current state), query observations (i.e., outputs of query module), and neighbor trajectories (sampled from the buffer) are plotted in blue, red, and green, respectively. To better visualize the agent's behaviors, we increased the color gradient of the state/query as a linear function of time. In the PT phase, since the agent needs to explore the space, we plot the density of all retrievable states as the intensity of the blue background color. We find that during pre-training, the query module tends to learn to guide the agent's future occupancy and lets it avoid local highly dense areas created by nearest neighbors in the buffer. In Fig. 6(a), the agent wishes to go to less dense areas to collect more rewards, e.g., the right bottom. The query module creates a "boundary" and guides the main agent to avoid densely populated areas, e.g., the right corner near the cross-shaped wall. For the FT case, we observe that the query module tends to create a "target" where the agent tends to somewhat go towards the queried states. Fig. 6(b) shows the reach bottom left task where the query module learns to create a "destination of point(s)".

Freezing SR during PT or FT. We conducted an additional experiment to evaluate the effectiveness of the SR module during both the pre-training and fine-tuning phases. Specifically, we set the reference vector to all zeros during pre-training and fine-tuning, respectively. The results, presented in Table 1, clearly demonstrate that the SR module enhances performance in each phase individually. Furthermore, employing the SR module in both pre-training and fine-tuning yields superior results, as it addresses distinct challenges inherent to each stage of the training process.

Scaling hyperparameters ablation. In our methodology, we have identified three critical hyperparameters for SR: the number of nearest neighbors (k), the length of neighbor sequences (D), and the context window size for retrievable states. These are set at 10, 5, and 100k, respectively. To understand how these hyperparameters affect the agent's performance, we conducted experiments using RND+SR with varying values. The experimental results, as illustrated in Fig. 7, reveal that an increase in these hyperparameters leads to higher scores for the agent, alongside a greater acquisition of reference information. This suggests that scaling up is beneficial for our method. However, it is important to note that increasing the context window size, the number of nearest neighbors, and the length of neighbor sequences all require additional computational resources. Thus, an optimal balance between computational demands and performance efficiency was established for all three pivotal hyperparameters of SR.

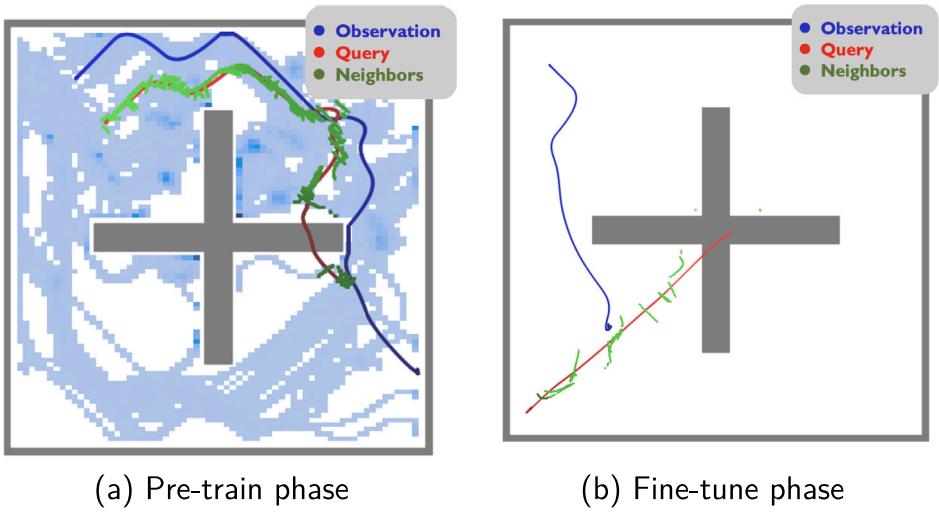

Fig. 6. Visualization of Query Module in Point Mass Env. (a) query module's output during PT: the blue background represents visited states, and the query output helps the agent to explore. (b) query module's output during fine-tuning: query outputs guide the agent to the target position.

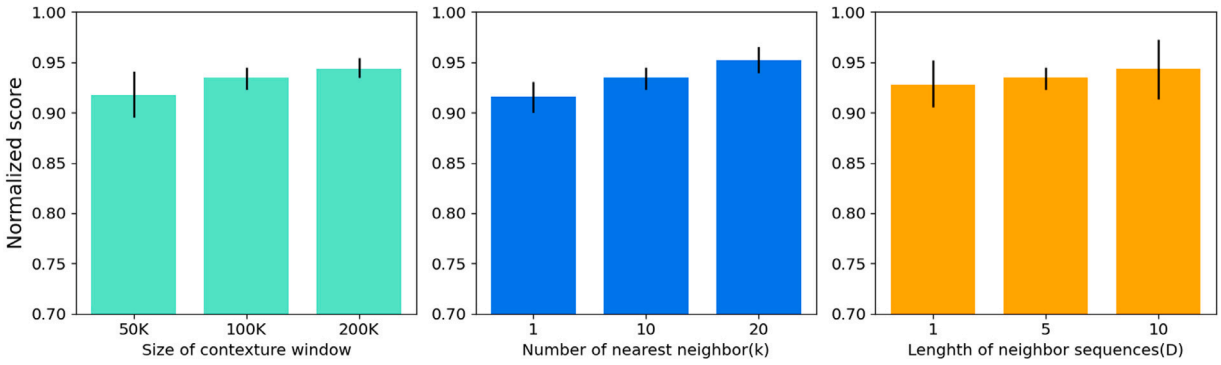

Fig. 7. Experiments of representative hyperparameters of Self-Reference. We use RND+SR and quadruped to evaluate the relationship between the agent's performance and three representative hyperparameters of SR: The number of nearest neighbors k, the length of neighbor sequences D, and the context window of retrievable states. We noticed a general trend of increasing reference capacity also increases performance.

Table 2
Policy change during fine-tuning. We compare the policy change during fine-tuning at 10k training steps to the policy at the end of pre-training in Quadruped.

Task	RND (KL/Norm. Intr.)	RND + SR (KL/Norm. Intr.)
Walk	18.9/0.46	16.9/0.59
Jump	17.5/0.55	17.1/0.66
Run	18.9/0.59	15.9/0.64
Stand	18.3/0.74	18.3/0.68

Alleviation of unlearning of PT policy. Works such as Campos et al. (2021) and Wolczyk et al. (2023) argue that naively fine-tuning an agent can exhibit varying degrees of catastrophic forgetting. Since the PT agent behaviors are often purely exploratory, quickly forgetting these exploratory properties might decrease the efficiency during FT in finding the optimal policy. As we explicitly show the agent its old behavior, we hypothesize that this explicit behavior could alleviate the unlearning of helpful exploratory behaviors. To determine if this is the case, we measure the extent of change from the PT behavior to the FT behavior by computing the distance of policy output distributions at every step. Specifically, inspired by KL-control (Stengel, 1986), we use the KL divergence from the policy during FT compared to its frozen PT policy as a proxy for the similarity between policies. Furthermore, we also use the drop in intrinsic reward of the FT policy as a proxy for how much the FT policy forgot. Since all FT policies experience the most significant drop in performance of the intrinsic reward at the 10k step, we report the normalized intrinsic return at this step with the average PT intrinsic return as the normalizer. Table 2 showcases the average KL divergence over fine-tuning and the normalized intrinsic return at 10k steps. We observe that RND + SR found "closer solutions" to the strong exploratory PT policy, confirming our intuition that explicitly showing old behaviors to the agent resulted in less unlearning of the PT policy.

We also conducted a comparison with the baseline method proposed by Campos et al. (2021), which addresses the issue of forgetting in unsupervised reinforcement learning. Following their approach, we employed a zeta distribution to sample the flight steps as described in their paper. The results, presented in Table 1, demonstrate that, at least within the domains we investigated, their method did not achieve superior performance compared to SR.

7. Conclusion and future work

In this paper, we present the Self-Reference (SR) method, an enhancement that significantly improves the effectiveness and efficiency of existing unsupervised reinforcement learning algorithms. Our method successfully mitigates the challenges of nonstationarity in the pre-training phase and prevents the unlearning of beneficial behaviors during fine-tuning. By explicitly integrating historical behaviors into current decision-making processes, SR not only preserves essential exploratory actions but also streamlines the learning process, as evidenced by our state-of-the-art results on the Unsupervised Reinforcement Learning Benchmark. These improvements were quantified

as a 5% average increase in IQM and an 11% average reduction in the Optimality Gap, highlighting the method's capability to enhance performance robustly.

Limitations and future work. In our work, we only explored retrieving references using a single query, therefore a natural way to enhance our method is to provide more than one query, which allows the agent to access multi-modal distributed information. Additionally, we used the state space as the key and query space for querying. This query space works well when we have the actual state of the world, but it could be inefficient if the environment is observational, e.g., images. Querying in a compact and meaningful space should significantly extend our method under these circumstances. Moreover, the non-stationarity of intrinsic rewards remains an ongoing challenge. SR is the first to address this issue in unsupervised RL scenarios while also being designed as an adaptive add-on for any intrinsic reward, which is an extremely challenging task. While SR makes significant progress, fully resolving this issue remains an exciting direction for future research and improvement. Furthermore, we only evaluated SR with model-free URL methods, and applying SR to model-based methods could be valuable for future work. Additionally, since SR requires a retrieval and aggregation step during PT-FT stages, there is an additional computation cost during training. Works can explore retrieving not at every step to reduce training computation. Lastly, another way to extend our approach is to augment the buffer of references with expert demonstrations or any other information that could be useful in helping the agent learn about the environment and task better. Despite some progress in the field, inherent problems in the URL paradigm still need to be addressed. We believe that coherent solutions are more effective in continuing to advance this area of research.

CRediT authorship contribution statement

Andrew Zhao: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Erle Zhu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Rui Lu:** Resources, Project administration, Investigation, Formal analysis. **Matthieu Lin:** Methodology, Investigation, Conceptualization. **Yong-Jin Liu:** Writing – review & editing, Supervision. **Gao Huang:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of Generative AI and AI-assisted technologies in the writing process

Statement: During the preparation of this work the author(s) used chatgpt (https://chatgpt.com/) in order to fix sentence level grammar and spelling mistakes. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gao Huang reports financial support, administrative support, and article publishing charges were provided by National Key R&D Program of China. Gao Huang reports financial support was provided by Natural Science Foundation of China. Andrew Zhao reports a relationship with BIG AI that includes: employment. Andrew Zhao reports a relationship with Microsoft Research that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by the National Key R&D Program of China under Grant 2024YFB4708200 and the National Natural Science Foundation of China under Grants U24B20173.

Data availability

Data will be made available on request.

References

Bellemare, Marc, Srinivasan, Sriram, Ostrovski, Georg, Schaul, Tom, Saxton, David, & Munos, Remi (2016). Unifying count-based exploration and intrinsic motivation. *NeurIPS*.

Burda, Yuri, Edwards, Harrison, Storkey, Amos, & Klimov, Oleg (2018). Exploration by random network distillation. In *ICLR*.

Campos, Víctor, Sprechmann, Pablo, Hansen, Steven Stenberg, Barreto, Andre, Kapturowski, Steven, Vitvitskyi, Alex, et al. (2021). Beyond fine-tuning: Transferring behavior in reinforcement learning. In *ICML 2021 workshop on unsupervised reinforcement learning*.

Chen, Wenhu, Hu, Hexiang, Saharia, Chitwan, & Cohen, William W (2022). Re-imagen: Retrieval-augmented text-to-image generator. arXiv preprint arXiv:2209.14491.

Choi, Samuel, Yeung, Dit-Yan, & Zhang, Nevin (1999). An environment model for nonstationary reinforcement learning. *NeurIPS*.

Eysenbach, Benjamin, Gupta, Abhishek, Ibarz, Julian, & Levine, Sergey (2018). Diversity is all you need: Learning skills without a reward function. In *ICLR*.

Glasmachers, Tobias (2017). Limits of end-to-end learning. In *Asian conference on machine learning* (pp. 17–32). PMLR.

Goyal, Anirudh, Friesen, Abram, Banino, Andrea, Weber, Theophane, Ke, Nan Rosemary, Badia, Adria Puigdomenech, et al. (2022). Retrieval-augmented reinforcement learning. In *ICML*. PMLR.

Guo, Zhaohan Daniel, Azar, Mohammad Gheshlaghi, Saade, Alaa, Thakoor, Shantanu, Piot, Bilal, Pires, Bernardo Ávila, et al. (2021). Geometric entropic exploration. CoRR, abs/2101.02055. URL https://arxiv.org/abs/2101.02055.

Guu, Kelvin, Lee, Kenton, Tung, Zora, Pasupat, Panupong, & Chang, Mingwei (2020). Retrieval augmented language model pre-training. In *ICML*. PMLR.

Hazan, Elad, Kakade, Sham M., Singh, Karan, & Soest, Abby Van (2019). Provably efficient maximum entropy exploration. In Kamalika Chaudhuri, & Ruslan Salakhutdinov (Eds.), *Proceedings of machine learning research: vol. 97, Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 June 2019, long beach, california, USA* (pp. 2681–2691). PMLR, URL http://proceedings.mlr.press/v97/hazan19a.html.

Huang, Shengyi, Dossa, Rousslan Fernand Julien, Ye, Chang, Braga, Jeff, Chakraborty, Dipam, Mehta, Kinal, et al. (2022). CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *JMLR*.

Humphreys, Peter Conway, Guez, Arthur, Tieleman, Olivier, Sifre, Laurent, Weber, Theophane, & Lillicrap, Timothy P (2022). Large-scale retrieval for reinforcement learning. In *NeurIPS*.

Jain, Arnav Kumar, Lehnert, Lucas, Rish, Irina, & Berseth, Glen (2023). Maximum state entropy exploration using predecessor and successor representations. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, & Sergey Levine (Eds.), *Advances in neural information processing systems 36: annual conference on neural information processing systems 2023, neurIPS 2023, new orleans, la, USA, December 10 - 16, 2023*. URL http://papers.nips.cc/paper_files/paper/2023/hash/9c7900fac04a701cbed83256b76dbaa3-Abstract-Conference.html.

Jing, Baoyu, Zhang, Si, Zhu, Yada, Peng, Bin, Guan, Kaiyu, Margenot, Andrew, et al. (2022). Retrieval based time series forecasting. arXiv preprint arXiv:2209.13525.

Johnson, Jeff, Douze, Matthijs, & Jégou, Hervé (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

Kim, Dongyoung, Shin, Jinwoo, Abbeel, Pieter, & Seo, Younggyo (2023). Accelerating reinforcement learning with value-conditional state entropy exploration. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, & Sergey Levine (Eds.), *Advances in neural information processing systems 36: annual conference on neural information processing systems 2023, neurIPS 2023, new orleans, la, USA, December 10 - 16, 2023*. URL http://papers.nips.cc/paper_files/paper/2023/hash/6530db249c161fe9254db2667453952c-Abstract-Conference.html.

Laskin, Michael, Liu, Hao, Peng, Xue Bin, Yarats, Denis, Rajeswaran, Aravind, & Abbeel, Pieter (2022). CIC: Contrastive intrinsic control for unsupervised skill discovery. In *NeurIPS*.

Laskin, Michael, Yarats, Denis, Liu, Hao, Lee, Kimin, Zhan, Albert, Lu, Kevin, et al. (2021). URLB: Unsupervised reinforcement learning benchmark. In *NeurIPS datasets and benchmarks track (round 2)*.

Lee, Lisa, Eysenbach, Benjamin, Parisotto, Emilio, Xing, Eric P., Levine, Sergey, & Salakhutdinov, Ruslan (2019). Efficient exploration via state marginal matching. CoRR, abs/1906.05274. URL http://arxiv.org/abs/1906.05274.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, et al. (2016). Continuous control with deep reinforcement learning.. In *ICLR*.

Liu, Hao, & Abbeel, Pieter (2021a). Aps: Active pretraining with successor features. In *ICML*. PMLR.

Liu, Hao, & Abbeel, Pieter (2021b). Behavior from the void: Unsupervised active pre-training. In *NeurIPS*.

Liu, Xin, Chen, Yaran, & Zhao, Dongbin (2023). ComSD: Balancing behavioral quality and diversity in unsupervised skill discovery. arXiv preprint arXiv:2309.17203.

Mutti, Mirco (2023). *Unsupervised reinforcement learning via state entropy maximization* (Ph.D. thesis), Italy: University of Bologna, URL http://amsdottorato.unibo.it/10588/.

Mutti, Mirco, Mancassola, Mattia, & Restelli, Marcello (2022). Unsupervised reinforcement learning in multiple environments. In *Thirty-sixth AAAI conference on artificial intelligence, AAAI 2022, thirty-fourth conference on innovative applications of artificial intelligence, IAAI 2022, the twelveth symposium on educational advances in artificial intelligence, EAAI 2022 virtual event, February 22 - March 1, 2022* (pp. 7850–7858). AAAI Press, http://dx.doi.org/10.1609/AAAI.V36I7.20754.

Mutti, Mirco, Pratissoli, Lorenzo, & Restelli, Marcello (2021). Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *Thirty-fifth AAAI conference on artificial intelligence, AAAI 2021, thirty-third conference on innovative applications of artificial intelligence, IAAI 2021, the eleventh symposium on educational advances in artificial intelligence, EAAI 2021, virtual event, February 2-9, 2021* (pp. 9028–9036). AAAI Press, http://dx.doi.org/10.1609/AAAI.V35I10.17091.

Mutti, Mirco, & Restelli, Marcello (2020). An intrinsically-motivated approach for learning highly exploring and fast mixing policies. In *The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence, EAAI 2020, new york, NY, USA, February 7-12, 2020* (pp. 5232–5239). AAAI Press, http://dx.doi.org/10.1609/AAAI.V34I04.5968.

Mutti, Mirco, Santi, Riccardo De, & Restelli, Marcello (2022). The importance of non-Markovianity in maximum state entropy exploration. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, & Sivan Sabato (Eds.), *Proceedings of machine learning research*: *162*, *International conference on machine learning, ICML 2022, 17-23 July 2022, baltimore, maryland, USA* (pp. 16223–16239). PMLR, URL https://proceedings.mlr.press/v162/mutti22a.html.

Nasiriany, Soroush, Gao, Tian, Mandlekar, Ajay, & Zhu, Yuke (2022). Learning and retrieval from prior data for skill-based imitation learning. In *CoRL*.

Nedergaard, Alexander, & Cook, Matthew (2022). K-means maximum entropy exploration. http://dx.doi.org/10.48550/ARXIV.2205.15623, CoRR, abs/2205.15623. URL https://doi.org/10.48550/arXiv.2205.15623.

Pathak, Deepak, Agrawal, Pulkit, Efros, Alexei A, & Darrell, Trevor (2017). Curiosity-driven exploration by self-supervised prediction. In *ICML*. PMLR.

Pathak, Deepak, Gandhi, Dhiraj, & Gupta, Abhinav (2019). Self-supervised exploration via disagreement. In *ICML*. PMLR.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, & Klimov, Oleg (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Seo, Younggyo, Chen, Lili, Shin, Jinwoo, Lee, Honglak, Abbeel, Pieter, & Lee, Kimin (2021). State entropy maximization with random encoders for efficient exploration. In Marina Meila, & Tong Zhang (Eds.), *Proceedings of machine learning research*: *139*, *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 July 2021, virtual event* (pp. 9443–9454). PMLR, URL http://proceedings.mlr.press/v139/seo21a.html.

Stengel, Robert F. (1986). *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc..

Strouse, DJ, Baumli, Kate, Warde-Farley, David, Mnih, Vlad, & Hansen, Steven (2021). Learning more skills through optimistic exploration. arXiv preprint arXiv:2107.14226.

Sutton, Richard S., & Barto, Andrew G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Tiapkin, Daniil, Belomestny, Denis, Calandriello, Daniele, Moulines, Eric, Munos, Rémi, Naumov, Alexey, et al. (2023). Fast rates for maximum entropy exploration. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, Jonathan Scarlett (Eds.), *Proceedings of machine learning research*: *202*, *International conference on machine learning, ICML 2023, 23-29 July 2023, honolulu, hawaii, USA* (pp. 34161–34221). PMLR, URL https://proceedings.mlr.press/v202/tiapkin23a.html.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, et al. (2017). Attention is all you need. *NeurIPS*.

Wolczyk, Maciej, Cupiał, Bartłomiej, Zając, Michał, Pascanu, Razvan, Kuciński, Łukasz, & Miłoś, Piotr (2023). On the role of forgetting in fine-tuning reinforcement learning models. In *Workshop on reincarnating reinforcement learning at ICLR 2023*.

Yang, Qisong, & Spaan, Matthijs T. J. (2023). CEM: constrained entropy maximization for task-agnostic safe exploration. In Brian Williams, Yiling Chen, & Jennifer Neville (Eds.), *Thirty-seventh AAAI conference on artificial intelligence, AAAI 2023, thirty-fifth conference on innovative applications of artificial intelligence, IAAI 2023, thirteenth symposium on educational advances in artificial intelligence, EAAI 2023, washington, DC, USA, February 7-14, 2023* (pp. 10798–10806). AAAI Press, http://dx.doi.org/10.1609/AAAI.V37I9.26281.

Yarats, Denis, Fergus, Rob, Lazaric, Alessandro, & Pinto, Lerrel (2021). Reinforcement learning with prototypical representations. In *ICML*. PMLR.

Zamboni, Riccardo, Cirino, Duilio, Restelli, Marcello, & Mutti, Mirco (2024a). How to explore with belief: State entropy maximization in POMDPs. In *Forty-first international conference on machine learning, ICML 2024, vienna, Austria, July 21-27, 2024*. OpenReview.net, URL https://openreview.net/forum?id=LbcNAIgNnB.

Zamboni, Riccardo, Cirino, Duilio, Restelli, Marcello, & Mutti, Mirco (2024b). The limits of pure exploration in POMDPs: When the observation entropy is enough. *RLJ*, *2*, 676–692.

Zhang, Chuheng, Cai, Yuanying, Huang, Longbo, & Li, Jian (2021). Exploration by maximizing renyi entropy for reward-free RL framework. In *Thirty-fifth AAAI conference on artificial intelligence, AAAI 2021, thirty-third conference on innovative applications of artificial intelligence, IAAI 2021, the eleventh symposium on educational advances in artificial intelligence, EAAI 2021, virtual event, February 2-9, 2021* (pp. 10859–10867). AAAI Press, http://dx.doi.org/10.1609/AAAI.V35I12.17297.

Zhao, Andrew, Huang, Daniel, Xu, Quentin, Lin, Matthieu Gaetan, Liu, Y., & Huang, Gao (2024). Expel: LLM agents are experiential learners. In *AAAI conference on artificial intelligence*.

Zhao, Andrew, Lin, Matthieu Gaetan, Li, Yangguang, Liu, Yong-Jin, & Huang, Gao (2022). A mixture of surprises for unsupervised reinforcement learning. In *NeurIPS*.

Zhao, Andrew, Xu, Quentin, Lin, Matthieu Gaetan, Wang, Shenzhi, Liu, Yong-Jin, Zheng, Zilong, et al. (2025). Diver-CT: Diversity-enhanced red teaming large language model assistants with relaxing constraints. In *AAAI conference on artificial intelligence*.

Zisselman, Ev, Lavie, Itai, Soudry, Daniel, & Tamar, Aviv (2023). Explore to generalize in zero-shot RL. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, & Sergey Levine (Eds.), *Advances in neural information processing systems 36: annual conference on neural information processing systems 2023, neurIPS 2023, new orleans, la, USA, December 10 - 16, 2023*. URL http://papers.nips.cc/paper_files/paper/2023/hash/c793577b644268259b1416464a6cdb8c-Abstract-Conference.html.