Pattern Recognition 123 (2022) 108352

Contents lists available at ScienceDirect

Pattern Recognition

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

Unified curiosity-Driven learning with smoothed intrinsic reward estimation

Fuxian Huang, Weichao Li, Jiabao Cui, Yongjian Fu, Xi Li*

College of Computer Science, Zhejiang University, Hangzhou, China

ARTICLE INFO

Article history:
Received 29 September 2020
Revised 18 September 2021
Accepted 25 September 2021
Available online 26 September 2021

Keywords:
Reinforcement learning
Unified curiosity-driven exploration
Robust intrinsic reward
Task-relevant feature

ABSTRACT

In reinforcement learning (RL), the intrinsic reward estimation is necessary for policy learning when the extrinsic reward is sparse or absent. To this end, Unified Curiosity-driven Learning with Smoothed intrinsic reward Estimation (UCLSE) is proposed to address the sparse extrinsic reward problem from the perspective of completeness of intrinsic reward estimation. We further propose state distribution-aware weighting method and policy-aware weighting method to dynamically unify two mainstream intrinsic reward estimation methods. In this way, the agent can explore the environment more effectively and efficiently. Under this framework, we propose to employ an attention module to extract task-relevant features for a more precise estimation of intrinsic reward. Moreover, we propose to improve the robustness of policy learning by smoothing the intrinsic reward with a batch of transitions close to the current transition. Extensive experimental results on Atari games demonstrate that our method outperforms the state-of-the-art approaches in terms of both score and training efficiency.

1. Introduction

Reinforcement learning has been successfully applied to various applications such as games [1,2], computer vision [3–5], recommender systems [6], and anomaly detection [7]. Despite the remarkable achievements of RL, it is still a major challenge to explore the environment and learn the policy when the extrinsic reward is sparse or absent. Especially in real-world applications, the reward is often missing. Thus, how to effectively tackle the sparse reward problem is a key research point.

Several recent studies seek to solve this problem based on experience replay [8–10], reward shaping [11,12], task redesigning [13–15], curiosity-driven learning [16,17], *etc.* The curiosity-driven learning addresses the sparse reward problem by introducing an exploration bonus that reflects the novelty of transition[1] There are two typical works of this method: 1) state novelty estimation [17], 2) state-action novelty estimation [16]. For 1), the agent is encouraged to visit novel states where the novelty is measured by the information gain in the observation space [17,18] or state density estimation [19], *etc.* As shown in Fig. 1 (a), the state novelty estimation method used in [17] focuses on estimating the novelty of the state where the agent will land in after taking a certain action, thus the agent learns to visit the most novel state. For 2), the agent is encouraged to take action where the state-action path has not been well modeled [16,20,21]. Fig. 1 (b) illustrates the forward dynamic prediction method used for state-action novelty in [16]. It assesses the ability of the model to predict the next state, and thus, the agent improves its understanding of the environment by adopting the path that is difficult to predict.

Although these two methods can alleviate the sparse extrinsic reward problem, there are still four limitations: 1) *only utilizing partial information of the transition.* The state novelty estimation merely reflects the state access and only considering it could miss some critical paths. Meanwhile, the state-action novelty estimation merely reflects path visiting, *i.e.*, $s_t, a_t \rightarrow s_{t+1}$, and only considering it could result in inadequate exploration for some states. 2) *ignoring the learned policy when estimating the reward.* Previous reward estimation methods are policy independent, and thus they could not utilize the characteristic of learned policy to guide the agent to explore states that the policy is uncertain about which action to take. 3) *ignoring the state distribution when estimating the reward.* Since the states are usually unevenly distributed in the state space, and key states are usually rare and located in small clusters [22,23]. When calculating intrinsic rewards, previous methods independently consider one transition. We expect that the relationship among different states can be taken into account to incentivize the agent to explore the states diversely. 4) *high variance of estimated intrinsic reward.* The unbalanced visiting frequencies of

* Corresponding author.
E-mail address: xilizju@zju.edu.cn (X. Li).

[1] A transition is the atomic unit of interaction in RL, and it usually contains (s_t, a_t, r_t, s_{t+1}). It only has three items: (s_t, a_t, s_{t+1}) in this paper, because the reward is considered to be sparse or absent.

https://doi.org/10.1016/j.patcog.2021.108352
0031-3203/

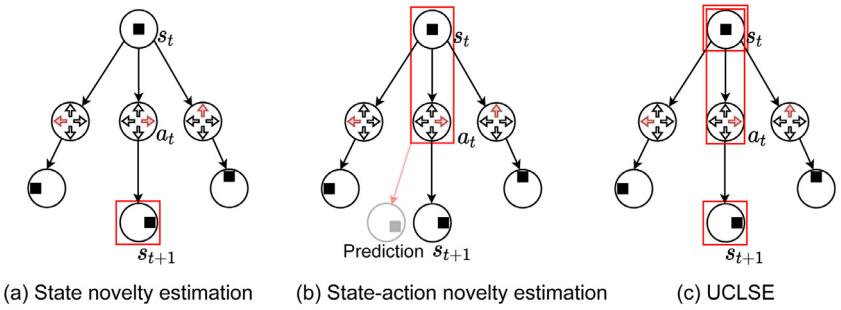

Fig. 1. The illustration of different methods used to estimate the intrinsic reward. (a) is the state novelty estimation focusing on predicting s_{t+1}. (b) is the state-action novelty focusing on predicting the path $s_t, a_t \rightarrow s_{t+1}$. (c) illustrates the main idea of our method, which unifies these two methods. The red box denotes the part used to compute the intrinsic reward. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

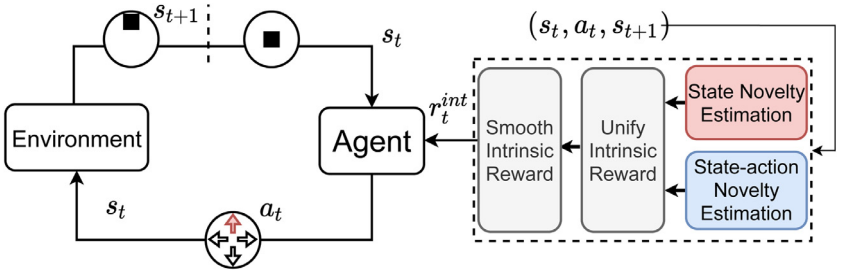

Fig. 2. The overview of our framework. The intrinsic reward r_t^{int} is obtained by unifying the results of state novelty estimation and state-action novelty estimation. The state novelty estimation calculates the intrinsic rewards of state s_t and s_{t+1}. The state-action novelty estimation estimates the intrinsic reward by investigating the uncertainty of the path $s_t, a_t \rightarrow s_{t+1}$. These intrinsic rewards are further smoothed to promote the stable learning of RL.

different states could lead to inconsistent intrinsic reward. Therefore, the learning would be unstable if the variance of estimated intrinsic reward is high.

Motivated by the above observations, we propose a novel distribution-aware and policy-aware unified curiosity-driven learning framework shown in Fig. 2. The framework presents a complete intrinsic reward estimating mechanism by unifying state novelty and state-action novelty. Specifically, the former reflects the exploration degree of state, *i.e.*, s_t and s_{t+1}, while the latter reflects the exploration degree of path, *i.e.*, $s_t, a_t \rightarrow s_{t+1}$. State distribution-aware weighting method (DAW) and policy-aware weighting method (PAW) are proposed to adaptively adjust the weights of above estimated rewards. By considering the distribution of the whole state space, DAW aims to encourage the agent to explore the state space diversely, especially for the states of small clusters. PAW utilizes the uncertainty of policy to incentivize the agent to explore uncertain states more frequently.

Furthermore, to reduce the high variance of estimated intrinsic reward, we utilize the neighbor information when estimating the intrinsic reward. In this way, we can obtain a smoothed intrinsic reward to make the policy learning procedure more stable. We complement our framework with an attention module, and it can shift the focus to task-relevant features. By integrating the above-proposed components, our method can generate more accurate and reliable intrinsic rewards to benefit RL policy learning.

The main contributions of this work are given as follows:

1) We propose a novel distribution-aware and policy-aware unified curiosity-driven learning framework to unify state novelty and state-action novelty. DAW enables the agent to explore states diversely, and PAW encourages the agent to explore the states that the policy is uncertain about which action to take. The proposed approach improves the exploration ability of RL with a complete intrinsic reward.

2) We propose an intrinsic reward smoothing method to alleviate the fluctuation of intrinsic rewards between similar transitions for a more efficient policy learning.

3) Extensive experiments on Atari games demonstrate the effectiveness of our approach in the case of sparse extrinsic reward or no extrinsic reward.

2. Background

In this section, we introduce some basic concepts about RL and two closely related methods within curiosity-driven learning.

2.1. Reinforcement learning

RL is usually formulated as Markov decision processes, or MDPs, and the finite MDPs can be defined as:

$$\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >, \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, R is the reward function, and $\mathcal{P}$ is the transition function.

The agent takes action a_t in state s_t according to policy π, and then it receives reward r_t and transfers to next state s_{t+1}. We obtain a series of transitions from the interaction with the environment, which are formatted as $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^{\infty}$.

The state-action value function $q_\pi(s, a)$ of a policy π is usually defined as:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s, a_t = a \right], \tag{2}$$

where γ is the discount parameter, and $\mathbb{E}_\pi$ denotes that the reward is expected of policy π. The value function $v_\pi(s)$ is the expectation of $q_\pi(s, a)$.

A greedy policy can be obtained from the state-action value function $q_\pi(s, a)$:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} q_\pi(s, a). \tag{3}$$

We define the optimal state-action value function q^* as:

$$q_\pi^*(s, a) = \max_\pi q_\pi(s, a). \tag{4}$$

The relationship between the optimal state value function v^* and the optimal state-action value function q^* can be formulated as:

$$v^*(s) = \max_{a \in \mathcal{A}} q^*(s, a). \tag{5}$$

Finally, the optimal policy π^* can be defined as:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} q^*(s, a). \tag{6}$$

2.2. Curiosity-driven learning

Curiosity-driven learning incentivizes the agent to explore unfamiliar areas by constructing the intrinsic reward. We introduce two specific methods corresponding to this method in Fig. 1 (a) and (b), *i.e.*, RND [17] and ICM [16].

RND proposes to quantify the novelty of states based on the distance between the output of a fixed, randomly initialized neural network and its prediction given by another trained neural network. The intrinsic reward estimated for next state s_{t+1} at time t is termed as r_t^N, and the computing process is formulated as:

$$g(s_{t+1}; \theta_N) = \|h(s_{t+1}; \theta_N) - \phi(s_{t+1})\|_2^2. \tag{7}$$

ICM estimates the intrinsic reward by predicting the next state based on the current state and action pair, and the reward is proportional to the prediction loss. r_t^F is the intrinsic reward for path: $s_t, a_t \rightarrow s_{t+1}$, and its computing process is formulated as:

$$f(s_t, a_t, s_{t+1}; \theta_F) = \|m(\phi(s_t), a_t; \theta_F) - \phi(s_{t+1})\|_2^2. \tag{8}$$

3. Related work

In RL, the sparse extrinsic reward problem has been studied extensively over the years. Generally, this problem can be tackled in many ways: sampling learnable transitions from experience replay [8–10]; imitating the demonstrations of expert [13]; transferring the learned policy to unseen tasks [24,25]; shaping the reward function to promote the agent to learn the policy [11,12,14,15], *etc.*

Experience replay has been exploited by many exploration algorithms to enhance the learning ability of agents. Prioritized experience replay (PER) [10] is proposed to learn the policy more efficiently by replaying important transitions more frequently. In goal-conditioned RL, hindsight experience replay (HER) [8] enables an agent to learn from failed trajectories by replacing the goal with a pseudo goal sampled from the replay buffer. Imitation learning (IL) [13,26,27] focuses on learning the policy from the experience replay given by an expert in a supervised way, and it avoids the sparse reward problem.

Some works alleviate the sparse reward problem via redesigning the learning mechanism. Hierarchical reinforcement learning [28–30] enhance the performance of learning difficult tasks by decomposing them to easier tasks. Reward shaping [12,31,32] is proposed to reshape the sparse reward function by inducing expert knowledge. Inverse reinforcement learning [14,15] can rebuild a reward function only depending on collected expert trajectories. Curriculum learning [33] aims to improve the learning ability by training on tasks from easy to hard. Following this idea, some works [34–36] show the sparse reward problem can be solved partially by incorporating knowledge from previous easy tasks. Some works [24,37] show that when the reward is sparse in one scenario, transferring the trained policy from another can reduce the difficulty in learning.

Most closely related to our work are curiosity-driven methods [16,17,19,38,39] that construct the exploration bonus to encourage the agent to explore novel states or actions. The well-known UCB algorithm of [40] introduces $\sqrt{\frac{2\log(t)}{n(a_t)}}$ as a preference for less frequently chosen action, where $n(a_t)$ is the number of times action a_t was previously chosen. Novelty search [41–49] is a general framework which can directly optimize the parameters with evolutionary algorithm by encouraging the agent to behave differently. The behavior is the basis for novelty estimation, and it can be quantified using the location [50], action sequence [51], *etc.* Model Based Interval Estimation-Exploration Bonus (MBIE-EB) of [25] adds a bonus reward of the form $\frac{1}{n(s,a)}$ to encourage exploring less-visited pairs. Count-based method [39] proposes to use the pseudo-count of state via static hashing as a reward bonus, and they show that it can be applied in high-dimensional state space. Bellman *et al.* [19] combine the count-based exploration and intrinsic motivation obtained with prediction gain to generalize the agent's uncertainty in its environment. In [16,17], DNN-based method is used to predict the state either in a forward dynamic way or in a random network distillation way, and the prediction error is treated as a bonus to incentivize the agent to explore novel states. Nikolay *et al.* [52] propose to compute the reward bonus via estimating whether the state can be reached within a constant step.

The aforementioned methods estimate the exploration bonus either from state novelty or state-action novelty, and they only utilize partial information from the transition. In comparison, our approach proposes a comprehensive intrinsic reward by adaptively unifying the novelties estimated from the state and the state-action path. Moreover, we propose a smoothing module to stabilizes the learning with robust intrinsic reward. Additionally, we utilize the attention module to attain semantic representation, which enables the policy more data-efficient.

4. Approach

4.1. Problem formulation

In this work, we aim to build a framework to solve the sparse extrinsic reward problem in RL. Assuming an agent takes action a_t when it observes s_t, and the environment returns next state s_{t+1} and extrinsic reward r_t^{ext}. Since the extrinsic reward r_t^{ext} may be sparse or absent, our goal is to compute an intrinsic reward r_t^{int} by estimating the uncertainty in the triple (s_t, a_t, s_{t+1}). Then the value function is formulated as Eq. 9. As the training process goes, the intrinsic reward will decrease, and the value function $v_\pi(s)$ would be Eq. 10 when the intrinsic reward is so small to be ignored. $v_\pi(s)$ would not be affected by the intrinsic reward at last.

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k (r_{t+k+1}^{ext} + r_{t+k+1}^{int}) | s_t = s \right]. \tag{9}$$

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}^{ext} | s_t = s \right]. \tag{10}$$

We estimate the intrinsic reward by considering the state novelty and the state-action novelty at the same time. In addition, we unify these two rewards with two novel adaptive novelty scheduling methods, i.e., distribution-aware weighting method and policy-aware weighting method. To alleviate the variance of estimated rewards, we propose to utilize the neighbor information to smooth the estimated rewards. Besides, the attention module is adopted to obtain task-relevant features.

We introduce the unified curiosity-driven learning in Section 4.2, the smoothing intrinsic reward estimation in Section 4.3, the attention module in Section 4.4, and an implementation example in the PPO algorithm in Section 4.5. To improve the readability of our paper, we provide the nomenclature in Table 1.

Table 1
Notation.

Notations	Definition
η_t	Adaptive weight obtained with DAW
βt	Adaptive weight obtained with PAW
γ	Discount coefficient of reward
r_t^{ext}	Extrinsic reward
r_t^{int}	Intrinsic reward
r_t^C	Intrinsic reward estimated for s_t
r_t^N	Intrinsic reward estimated for s_{t+1}
r_t^F	Intrinsic reward estimated for $s_t, a_t \rightarrow s_{t+1}$
$\phi(s_t)$	Feature for state s_t
$\overline{r_t^C}$	Smoothed intrinsic reward for r_t^C
$\overline{r_t^N}$	Smoothed intrinsic reward for r_t^N
$\overline{r_t^F}$	Smoothed intrinsic reward for r_t^F
$d(s_i, s_t)$	Distance between state s_i and state s_t
$B_\delta(s_t)$	δ-neighborhood of state s_t
w_j	Similarity between $s_{t,j}$ and s_t
$\mathbf{H}$	Feature map after convolution layer
$\mathbf{M}$	Attentional mask
$\mathbf{H}'$	Attentional feature for $\mathbf{H}$

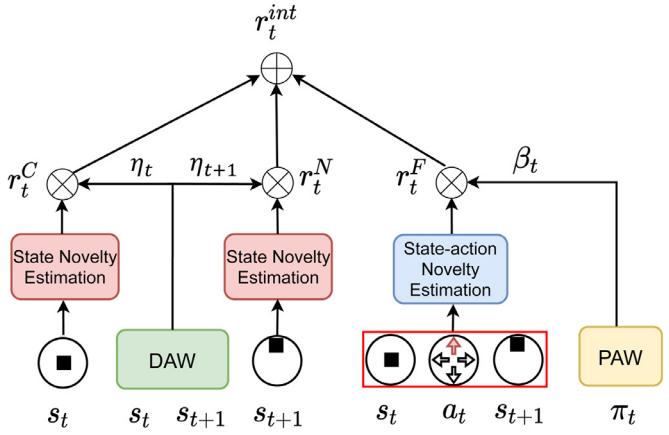

Fig. 3. The architecture demonstrates the dynamically unified estimation of the intrinsic reward for transition (s_t, a_t, s_{t+1}). The intrinsic rewards r_t^C, r_t^N and r_t^F are firstly obtained with state novelty estimation method and state-action novelty estimation method respectively. After that, DAW and PAW calculates their adaptive weights. Finally, the unified intrinsic reward r_t^{int} is obtained by dynamically unifying r_t^C, r_t^F and r_t^N as shown in Eq. 11.

4.2. Unified curiosity-Driven learning

Curiosity-driven learning methods usually utilize the intrinsic reward obtained in a self-supervised way to guide the exploration in the environment. As shown in Eqs. (7) and (8), RND and ICM only extract partial information from the transition, and these incomplete rewards may prevent the agent from exploring the environment effectively. To tackle this problem, we propose to calculate the intrinsic reward by considering three aspects of novelty in the transition (s_t, a_t, s_{t+1}). As shown in Fig. 3, the unified intrinsic reward contains the novelty of states s_t, s_{t+1}, *i.e.*, $r_t^C = g(s_{t+1}; \theta_C)$, $r_t^N = g(s_t; \theta_N)$, and the novelty of path from s_t to s_{t+1}, *i.e.*, $r_t^F = f(s_t, a_t, s_{t+1}; \theta_F)$. θ_C, θ_N and θ_F are the parameters of these models respectively. The state novelty estimation merely contains the state access information, and only considering it could miss some critical paths. Meanwhile, the state-action novelty estimation merely contains the path visiting information, and only considering it could result in inadequate exploration for some states. Thus, they are naturally complementary and mutually reinforced. They altogether provide a more effective bonus to encourage the agent to explore the environment.

To effectively unify these estimated rewards, we further propose state distribution-aware weighting method (DAW) and policy-aware weighting method (PAW) to establish a dynamic reward weight scheduling mechanism, which can adaptively unify state novelty and state-action novelty. Specifically, as shown in Eq. 11, η_t and η_{t+1} are the adaptive weights for state novelty estimation obtained by DAW. To encourage the agent to explore states diversely, we increase η_t and η_{t+1} for rare states. β_t is the adaptive weight for state-action novelty estimation obtained by PAW. We encourage the agent to explore uncertain states by increasing their β_t.

Hence, the unified intrinsic reward can be formulated as:

$$r_t^{int} = \eta_t r_t^C + \eta_{t+1} r_t^N + \beta_t r_t^F. \tag{11}$$

To compute η_t, we first constantly collect the visited state s_t, and then these states are divided into K clusters. Since the number of states is very large, it is computationally difficult to do so directly. Thus, we adopt SimHash [53,54] to reduce the dimensionality of states. Let state s_t be in cluster i and it has n_i states, then the corresponding η_t can be formulated as Eq. 12. Suppose the cluster i is small, *i.e.*, $n_i < \frac{1}{K}\sum_{i=1}^{K} n_i$, and then η_t will be bigger than 1. In this case, state novelty of s will be increased for encouraging the agent to explore this rare state and *vice versa*.

$$\eta_t = 1 - \frac{n_i - \frac{1}{K}\sum_{i=1}^{K} n_i}{\sum_{i=1}^{K} n_i}. \tag{12}$$

The β_t is used to guide the agent to visit the states that need more exploration. The strategy entropy is proved to be the measure of strategy's uncertainty, and it can be used as the indicator of convergence degree to balance the exploration and exploitation [55,56]. Therefore, we propose to utilize the entropy of the action distribution to dynamically adjust the weight of exploration bonus. Specifically, we first obtain the output of the policy given state s_t, and it is denoted as $\mathbf{A} = [p_1, p_2, \cdots, p_Z]$, where Z is the size of action space and p_i is the probability of taking action a_i. In this case, β_t can be formulated as:

$$\beta_t = -\sum_{i=1}^{Z} p_i \ln(p_i). \tag{13}$$

4.3. Smooth intrinsic reward estimation

Typically, the intrinsic reward, for example, r_t^N, is obtained with one transition. Since different states are visited unevenly, similar states may still have different intrinsic rewards. To reduce the fluctuation of estimated intrinsic rewards, we introduce how to get a smoothed reward with novelty estimation from neighbor transitions as a smoothness regularization.

We define the δ-neighborhood of state s_t as $B_\delta(s_t)$, where all its neighbors in $B_\delta(s_t)$ are within a specific small distance. We utilize an external memory E to store transitions generated from the interaction between the agent and the environment. At every time step t, the transition (s_t, a_t, s_{t+1}) is stored in E. When we need to calculate the smoothed intrinsic reward, for example, $\overline{r_t^N}$, we firstly retrieve a batch of transitions from the δ-neighborhood of s_t, $\{(s_{t,j}, a_{t,j}, s_{t+1,j})\}_{j=1}^{D}$, where D is the batch size.

As introduced in Eq. (7), the next state intrinsic reward for every sampled transition $(s_{t,j}, a_{t,j}, s_{t+1,j})$ can be formulated as:

$$r_{t,j} = g(s_{t,j}; \theta_N). \tag{14}$$

As shown in Fig. 4, the corresponding smoothed intrinsic reward for r_t^N is estimated from the weighted average of rewards from neighbor transitions. And it can be formulated as:

$$\overline{r_t^N} = \sum_{j=1}^{D} w_j r_{t,j}, \tag{15}$$

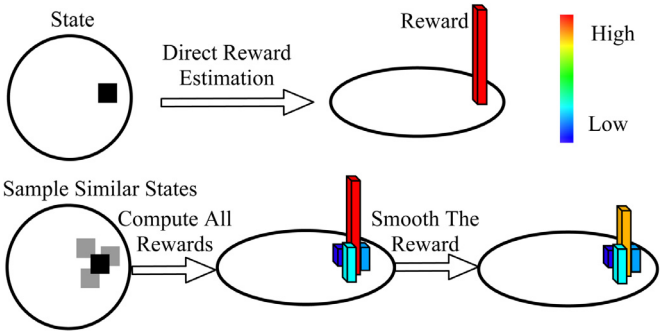

Fig. 4. The comparison of different methods for estimating intrinsic reward. The first row shows the general way the intrinsic reward is computed by previous methods. The second row illustrates the procedure for calculating smoothed intrinsic reward by our method. The black rectangle in the circle represents the state for which we estimate intrinsic reward, and the gray rectangles are the neighboring states used for smoothing the estimated intrinsic reward. The cuboids in the oval represent the intrinsic rewards computed, and the color and height of them are consistent, and they are used to show the magnitude of intrinsic rewards. The color of the rectangle in the circle does not represent the value of states, but to distinguish them.

where w_j is the similarity between $s_{t,j}$ and s_t, and it is defined as:

$$w_j = \frac{\exp(-d(s_{t,j}, s_t))}{\sum_{j=1}^{D} \exp(-d(s_{t,j}, s_t))}. \tag{16}$$

where $d(\cdot,\cdot)$ is a distance function that is inversely proportional to the similarity.

$\overline{r_t^C}$ and $\overline{r_t^F}$, the smoothed intrinsic rewards for r_t^C and r_t^F, can be obtained in the same way as $\overline{r_t^N}$. In this case, the unified intrinsic reward can be formulated as:

$$\overline{r_t^{int}} = \beta_t \overline{r_t^F} + \eta_t \overline{r_t^C} + \eta_{t+1} \overline{r_t^N}. \tag{17}$$

We also illustrate the T-SNE [57] results of methods with and without smoothing in Fig. 5. As shown in Fig. 5, the clustering results with and without smoothing are very similar both at 0.1 hours and 4.0 hours after the training, which is shown in the first and third columns. The result with smoothing in the second column is more similar to the result in the third column than the method without smoothing. This proves the effectiveness of the smoothing method, which can improve the training speed of RL.

4.4. Attention module

Recent works [58–60] demonstrate that RL with attention modules could achieve impressive results.

[61,62] show that randomly-initialized neural networks can achieve impressive results in RL tasks and image processing tasks with the proper network architecture. Thus, the feature extractor we adopt is a convolutional neural network adopted from [17], which is fixed after random initialization.

The random feature is more compact and representative than raw pixels to describe the state. Furthermore, it is more stable than a trainable neural network, and it stabilizes the training procedure. However, we find the feature map can be more representative if the random feature is selected by an attention module. The network structure is shown in Fig. 6, and the corresponding estimation procedure is formulated in Eq. (18).

$$\mathbf{H}' = \mathbf{H} \odot \mathbf{M}, \tag{18}$$

where $\mathbf{H}$ is the feature map generated by the convolution layer, $\mathbf{M}$ is the attention mask and $\mathbf{H}'$ is the attention feature map. To further investigate the effect of the attention module, we visualize the attention masks on the corresponding states in Fig. 7.

As shown in Fig. 7, the mask is sparse and state-relevant. The mask can cover most of the important areas in the state. For example, the attention mask only highlights the paddles and ball in (a) and (d), and these are critical objects for Breakout and Pong. In this case, the prediction on irrelevant parts will be ignored, and the intrinsic reward can be more precise and reliable.

4.5. Implementation

Our proposed method focuses on estimating intrinsic reward with sampled transitions. This intrinsic reward is used to learn the policy when extrinsic reward is sparse or absent. Additionally, it is independent of any RL algorithm, and it can be applied to variants kinds of RL algorithms. We implement our method with Proximal Policy Optimization as an example. The learning procedure can be summarized in Algorithm 1. Specifically, when the state s_t arrives,

Algorithm 1 Unified Curiosity-driven Learning with Smoothed Intrinsic Reward Estimation.

Initialize policy parameters θ_0, KL penalty λ, memory E, update frequency $Freq$, max training count K, episode length T.
for $k \leftarrow 1 \rightarrow K$ **do**
  Initialize s_0
  for $t \leftarrow 1 \rightarrow T$ **do**
    Take action a_t, observe s_{t+1} and r_t^{ext} (if it exists)
    Store (s_t, a_t, s_{t+1}) in memory E
    Sample transitions which are similar with transition (s_t, a_t, s_{t+1})
    Estimate the unified intrinsic rewards with (11)
    Calculate smoothed unified intrinsic reward $\overline{r_t^{int}}$ with Eq.~(15),~(17)
    Obtain summarized reward $r_t = \overline{r_t^{int}} + r_t^{ext}$
    $s_t = s_{t+1}$
    if $k\%Freq = 0$ **then**
      Collect set of partial trajectories G_k on policy $\pi_k = \pi(\theta_k)$
      Estimate advantages $\hat{A}^{\pi_k}$ using any advantage estimation algorithm
      Compute policy update $\theta_{k+1} = \arg\max_{\theta} L_{\theta_k(\theta)} - \lambda \bar{D}_{KL}(\theta \| \theta_k)$
    end if
  end for
end for

the agent takes action a_t, and then it receives next state s_{t+1}, we then store this transition (s_t, a_t, s_{t+1}) in memory. During the training procedure, we sample transitions similar to (s_t, a_t, s_{t+1}), and calculate the intrinsic rewards from different aspects. Afterwards, these intrinsic rewards are smoothed with Eq. (15). Finally, the smoothed unified intrinsic reward $\overline{r^{int}}$ for transition (s_t, a_t, s_{t+1}) is obtained with Eq. (17). The policy can be updated with the transition and new reward $\overline{r_t^{int}} + r_t^{ext}$.

5. Experiments

In this section, we mainly investigate and answer the following questions:

1. Can our proposed method improve performance significantly across different games when the extrinsic reward is sparse or absent? How does it compare to the state-of-the-art methods in exploration for deep RL?

2. What is the impact of different modules, *i.e.*, the unifying module, the smoothing module, the attention module and two adaptive weighting methods?

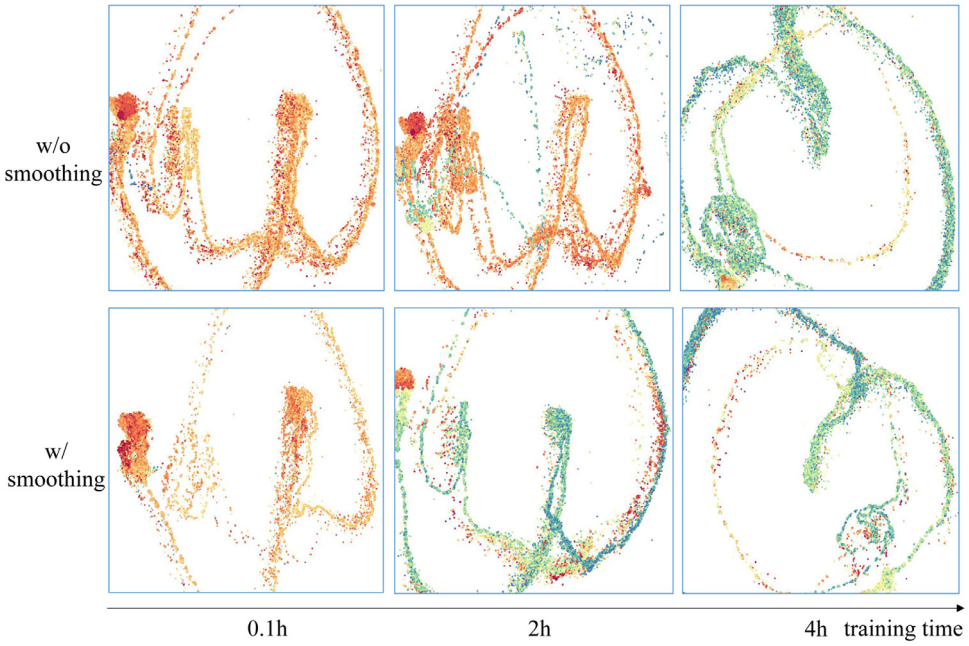

Fig. 5. The comparison of T-SNE results in the UpNDown task with and without smoothing. The points represent the states during the training procedure. The color shows the magnitude of its value $v(s)$. The results in these columns are obtained after training for 0.1 hours, 2 hours and 4 hours.

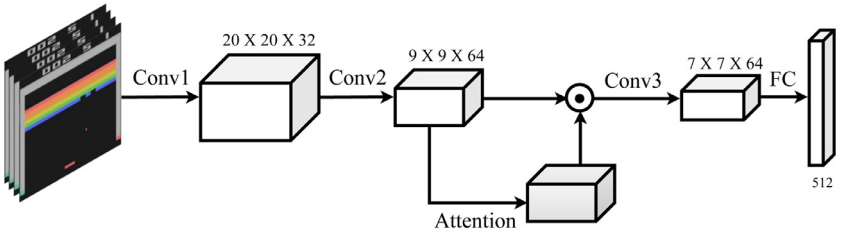

Fig. 6. The illustration of the structure for the feature extractor network. The attention mask is learned after the feature map of the state. The dot product of this mask and the feature map is passed to the next layer.

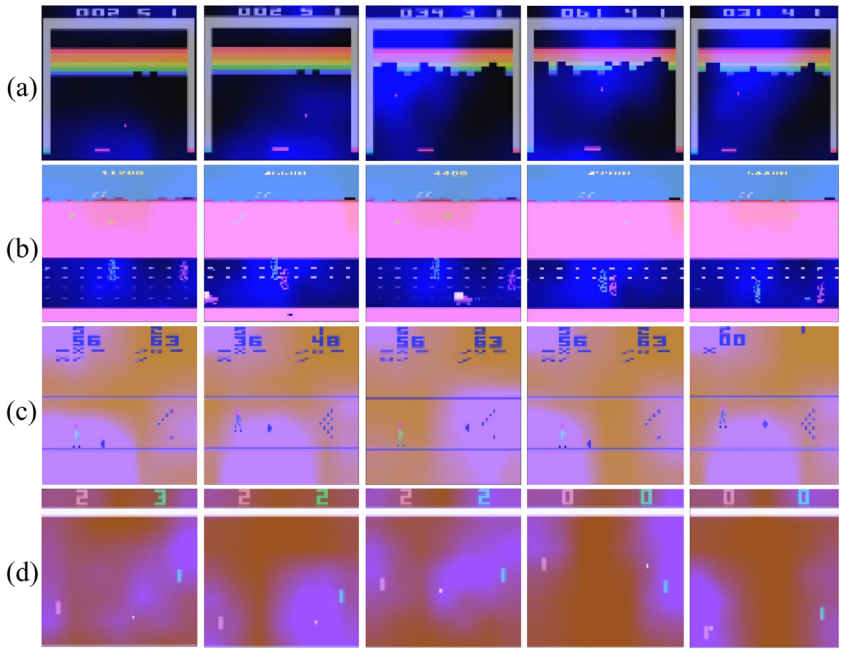

Fig. 7. Illustration of the attention mask in several Atari games, and from (a) to (d) are Breakout, RoadRunner, Bowling, Pong. The purple regions denote where the attention mask focuses on. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Hyper-parameters for our approach.

Hyper-parameters	Values
replay buffer size	$2e6$
fingerprint length	14
number of transitions for smoothing	10
number of timesteps	$1e8$
discounting factor	0.99
advantage estimation discounting factor	0.95
state shape	84×84
learning rate	$1e-4$
environments per process	128
backbone algorithm	PPO

To answer question 1, we run the proposed method on two groups of games on the RL benchmark. One group is without extrinsic reward, the other is with sparse extrinsic reward. We compare our method with four algorithms, ICM [16], RND [17], Count-based [39], NS-GA [51] and NS-ES [50]. This question is studied in Sections 5.2 and 5.3. Question 2 is answered by studying the effectiveness of different modules in the ablation experiments. This question is studied in Section 5.4.

5.1. Setup

The experiments are carried out on the Arcade Learning Environment (ALE [63]). ALE is a popular platform to test RL algorithms, especially for those CNN-based methods, because it provides raw images as states. The ALE contains dozens of Atari games in various categories. In this experiment, we choose two groups of games to evaluate the proposed method. The extrinsic reward for the first group of games is not sparse, and it is relatively easy to learn the optimal policy for these games. Thus, we evaluate our method on these games without the extrinsic reward. The second group of games is famous for the sparse extrinsic reward. In this case, we evaluate our method with the mixed intrinsic reward and extrinsic reward for these games.

The setting of hyper-parameters is presented in Table 2. Specifically, the hyper-parameters in the first three rows are used for the smoothing module, and the lambda values are used for the unifying module. Other hyper-parameters are general for PPO algorithm, and we follow the default settings of the baseline method [16]. Furthermore, our algorithm is implemented in Tensorflow on NVIDIA GTX TITAN GPU and Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz.

5.2. Only intrinsic reward

In this section we evaluate the performance of our method without any extrinsic reward on the first group of games. The first group has 12 games belonging to 4 classes, including the shooter games, the racing games, the adversarial games, and the strategy games. The agent in shooter games aims to destroy obstacles using weapons [Fig. 8(a)]. The agent in racing games is required to move faster towards the destination than competitors [Fig. 8(b)]. The adversarial games focus on winning the opponents [Fig. 8(c)]. The agent in strategy games need to have an overall consideration of the game world [Fig. 8(d)].

Since there is no extrinsic reward to guide the agent to explore where the "prize" is, the agent is only encouraged to access more interesting states by pursuing higher cumulative intrinsic reward. During this procedure, it can learn some useful skills to see more novel states. For example, in Breakout the agent has to learn to move the paddle to bounce back the ball in order to destroy the bricks. Only in this way, more novel states can be seen, or else there is one state with all the bricks stacked on the top. In this case, the training efficiency and mean episodic return can reflect the exploration effect to some extent.

We compare with the baseline methods in terms of learning speed and game score. As shown in Fig. 9, our method converges much faster than other methods on Breakout, UpNDown, Gopher, Asterix, *etc.* As shown in Table 3, our method achieves the highest game scores on 12 games, which is consistent with the analysis before. Compared with the best results of ICM and RND on these games, our method improves around 33.4% on the average scores of all these games.

5.3. Intrinsic reward with sparse extrinsic reward

In this section, we evaluate our method on the second group of games, which are Gravitar, MontezumaRevenge, Freeway, and Pitfall. These games are difficult to be trained well for the sparse extrinsic reward. To remedy this problem, we propose the complementary intrinsic reward to promote the learning ability of the agent.

The training results are shown in Fig. 10 and Table 3. We can see that our approach outperforms the baseline methods by a large margin on Gravitar and MontezumaRevenge. However, we just attain fair results on Freeway and Pitfall. The corresponding analyses are given as follows. The player in Freeway needs to cross the road while avoiding being hit by vehicles running from right to left. Its action space dimension 3 is and the size of its state space is small. The extrinsic reward can be obtained after several steps. Thus, both our approach and ICM achieve high scores, which are nearly the top machine score[2]. The player in Pitfall, whose action space dimension is 18, needs to accomplish a series of complicated tasks, which is very hard to be learned well. The extrinsic reward can only be obtained after many steps. Thus, to the best of our knowledge, there is no method that can solve this problem without pre-defined rules [64].

5.4. Ablation study

In this section, we study the effectiveness of each module and the influence of the hyper-parameters. Firstly, we perform the ablation studies on the components, *i.e.*, the unifying module, the smoothing module, the attention module, distribution-aware weighting method, and policy-aware weighting method. Later, we study the influence of different hyper-parameters, *i.e.*, the position of attention layer and size of the smoothing cluster.

Proposed modules. We mainly show the benefits of each component in two aspects: 1) *comparing with the independent component*, *i.e.*, the unifying module, DAW and PAW. As shown in Table 4, Unify, DAW, and PAW are proved to be effective. They all achieve higher game scores than baseline on both sparse extrinsic reward and only intrinsic reward cases. Additionally, as shown in Fig. 11 (1), the weights for PAW and DAW are higher in the early stage to encourage the agent to explore the environment, and as the unknown space to explore decreases, the weights gradually decrease. As shown in Fig. 11 (2), the value function loss of the smoothing method fluctuates in a smaller range compared to baseline, and this indicates that the proposed learning procedure is more likely to be stable due to using the smoothing strategy; 2) *adding the attention and smoothing module to baseline*, *i.e.*, ICM-A (ICM with the attention module), RND-A (RND with the attention module), ICM-S (ICM with the smoothing module), RND-S (RND with the smoothing module), Unify-A (Unify with the attention module) and Unify-S (Unify with the smoothing module). As shown in Table 4, we find ICM-A, RND-A, ICM-S, RND-S, Unify-A, and Unify-S outperform ICM, RND, and Unify in terms of game score. For example, ICM-A,

[2] https://github.com/cshenton/atari-leaderboard

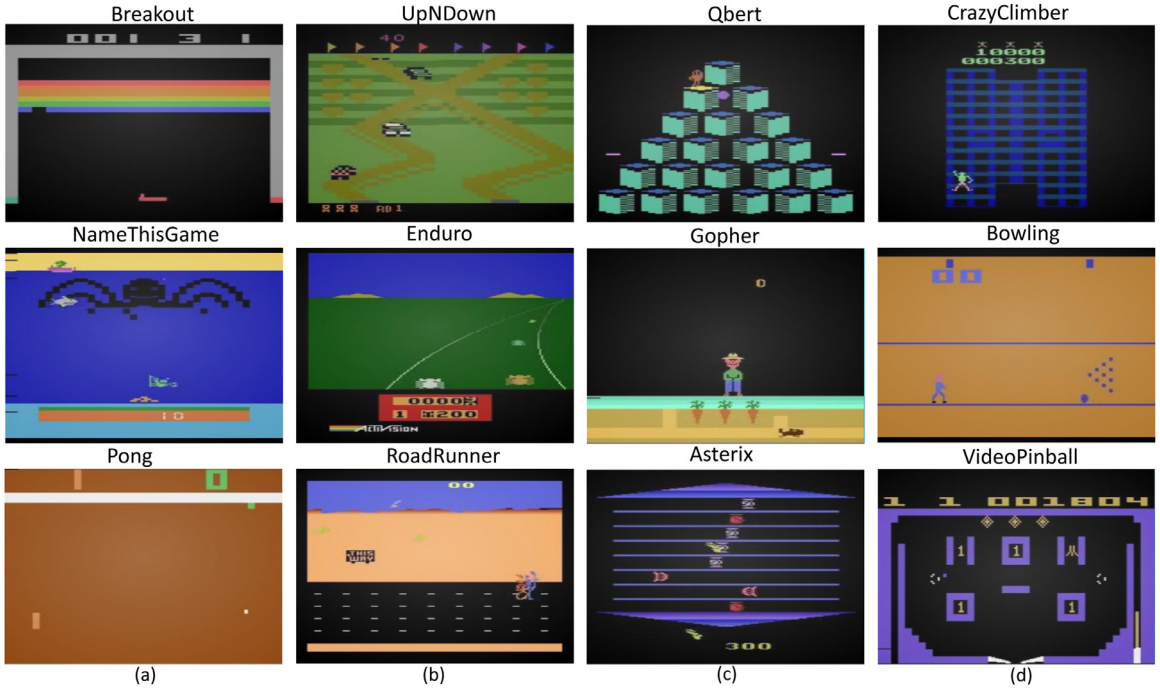

Fig. 8. A screen shot of 12 Atari games tested in this paper. (a) Three shooter games: Breakout, NameThisGame, and Pong. (b) Three racing games: UpNDown, Enduro, and RoadRunner. (c) Three adverthesarial games: Qbert, Gopher, and Asterix. (d) Three strategy games: CrazyClimber, Bowling, and VideoPinball.

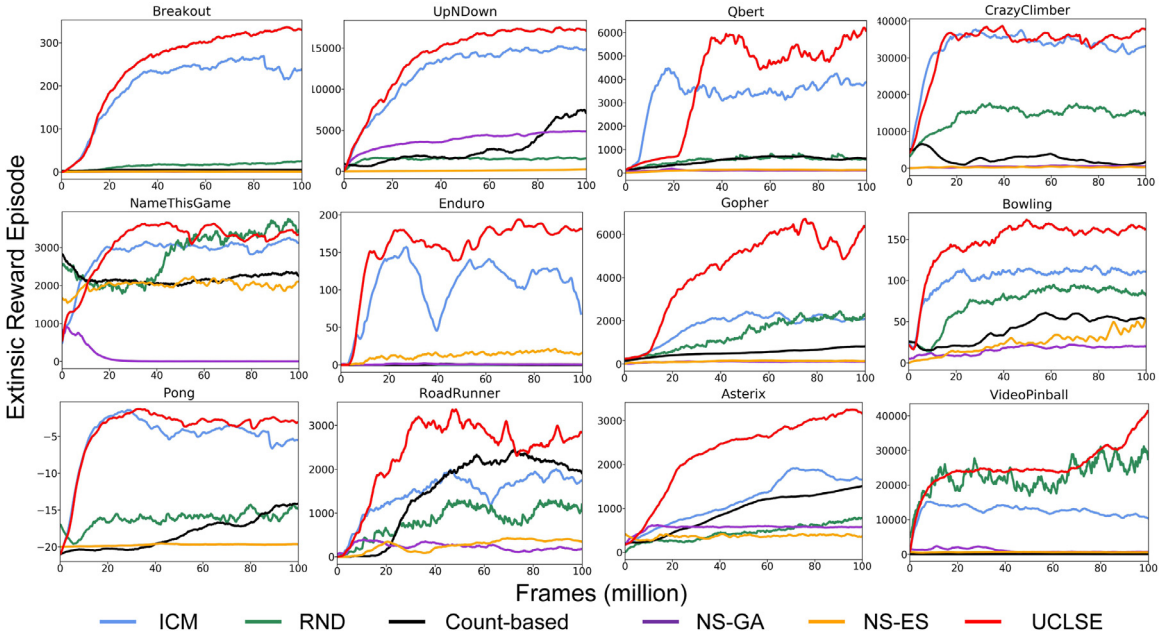

Fig. 9. The experimental comparison among ICM (blue), RND (orange), Count-based (black), NS-GA (purple), NS-ES (orange), UCLSE (red). The results show that our approach can improve the performance of different games under the condition of no extrinsic reward. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

RND-A achieve 17% and 11% improvement over ICM and RND on average, respectively. These experimental results indicate the effectiveness of the proposed modules.

Position of attention layer. We have shown the influence of the attention layer on different positions. As shown in Table 5, we find that the most suitable convolutional layer for different games is different. Thus, we just tune this parameter on several games, and we choose to set it on the first convolution layer.

Size of the smoothing cluster. We have shown the influence of different size of cluster in Table 6, and different games do not have the same best cluster size. Therefore, we just tune this parameter on several games, and we choose to set the cluster size to be 14.

5.5. Complexity discussion

The added complexity is mainly caused by three modules: the unifying module, the attention module, and the smoothing module. The unifying module results in approximately 1.6 million parameters and 18.7 million FLOPs by adding a subnet, which accounts for approximately 67% of the computation of the original network.

Table 3
Game scores of ICM, RND, Count-based, Novelty Search, and UCLSE. The higher the better.

Game	ICM(±*std*)	RND(±*std*)	Count-based(±*std*)	NS-GA(±*std*)	NS-ES(±*std*)	UCLSE(±*std*)
Pong	-5.5±0.0	-14.8±0.0	-14.2±0.0	-19.6±0.0	-19.6±0.0	**-3.1±0.0**
Breakout	237.6±5.2	24.6±0.0	4.6±0.0	0.6±0.0	0.4±0.0	**334.7±0.4**
Bowling	110.7±0.1	83.5±0.2	52.3±0.8	20.0±0.4	51.0±1.0	**163.0±0.2**
Enduro	71.2±3.4	0.0±0.0	0.0±0.0	0.5±0.0	15.6±0.1	**181.3±0.1**
Asterix	1657.2±3.7	780.9±9.6	1503.7±3.2	578.8±0.8	352.9±7.6	**3178.4±29.7**
Gopher	2079.6±27.9	2248.3±59.8	809.2±2.2	96.0±0.3	128.9±0.1	**6377.5±151.4**
Qbert	3864.3±101.4	574.1±3.6	615.7±2.4	104.9±0.1	123.9±0.0	**6123.5±61.6**
NameThisGame	3123.6±18.1	**3453.9±23.3**	2261.6±66.2	3.0±0.0	2099.2±3.9	3334.8±35.8
RoadRunner	1732.6±32.7	1076.4±46.3	1934.5±119.2	175.4±1.8	350.7±2.9	**2848.5±73.2**
UpNDown	14827.1±151.5	1570.1±5.1	7220.1±67.9	4848.8±9.7	246.9±2.7	**17143.5±714.4**
CrazyClimber	33145.1±571.9	14804.5±757.4	1655.4±87.9	512.1±67.6	296.5±27.1	**37834.8±387.6**
VideoPinball	10434.3±255.4	28471.4±381.3	52.2±0.0	706.6±14.8	556.6±2.1	**41185.8±896.5**
Pitfall	-2.7±0.0	-2.7±0.0	-3.1±0.0	-73.0±0.4	-214.5±2.4	**-0.2±0.0**
Freeway	**33.9±0.0**	22.1±0.0	29.5±0.0	3.9±0.4	3.0±0.0	**33.9±0.0**
Gravitar	827.7±0.7	1059.1±15.1	400.1±0.1	95.8±0.1	112.6±0.2	**2910.3±29.6**
Montezuma's Revenge	400.0±0.0	4153.8±85.8	0.5±0.0	0.0±0.0	0.0±0.0	**6648.7±10.8**

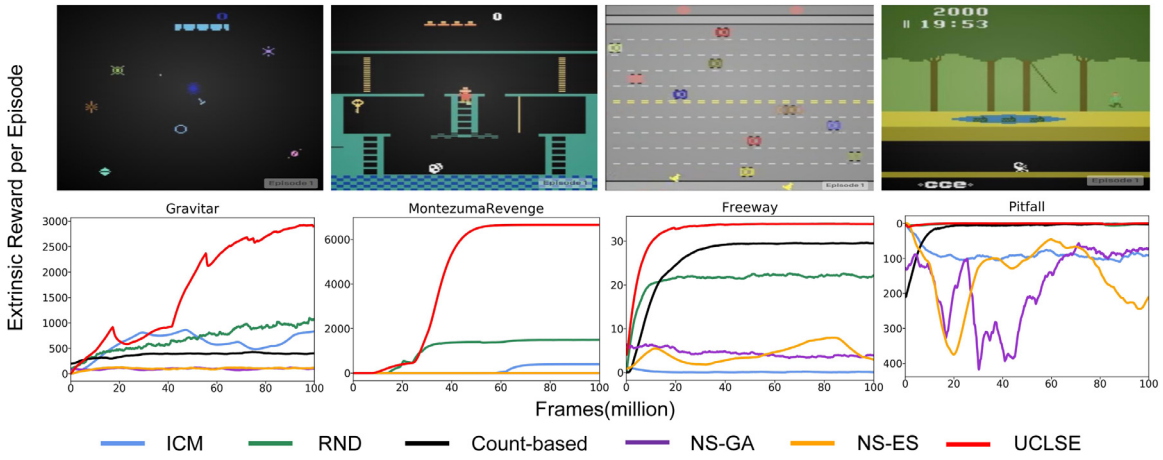

Fig. 10. The experimental comparison among ICM (blue), RND (green), Count-based (black), NS-GA (purple), NS-ES (orange), and UCLSE (red). The results show that our approach can improve the efficiency and game score in some games with the sparse extrinsic reward, especially for Gravitar and MontezumaRevenge. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Game scores of ICM, ICM-A, ICM-S, RND, RND-A, RND-S, Unify, Unify-A, Unify-S, and UCLSE on Breakout, Asterix, UpNDown, Gravitar and MontezumaRevenge. The higher the better.

Method	Breakout	Asterix	UpNDown	Gravitar	MontezumaRevenge
ICM	237.6±5.2	1657.2±3.7	14827.1±151.5	827.7±0.7	400.0±0.0
ICM-A	272.9±0.4	2160.7±9.5	15924.4±298.3	910.0±6.5	500.0±0.0
ICM-S	283.4±0.1	2209.0±5.6	15490.5±390.9	1072.6±0.4	400.0±0.0
RND	24.6±0.0	780.9±9.6	1570.1±5.1	1059.1±15.1	4153.8±85.8
RND-A	30.7±0.0	719.2±12.4	13115.9±107.9	1392.0±1.8	4410.6±19.8
RND-S	26.4±0.0	816.6±7.1	8243.6±62.4	1277.9±0.3	4338.0±51.3
Unify	237.6±0.2	1657.2±3.7	14827.1±151.5	1302.4±0.4	5204.3±197.8
Unify-A	326.0±0.1	2856.7±4.8	16719.1±75.3	1419.0±0.1	5719.1±75.3
Unify-S	321.4±0.0	2793.9±3.3	16665.7±31.0	1386.7±5.3	5665.7±31.0
Unify-PAW	268.6±1.3	2915.8±0.5	16816.6±451.0	1602.3±30.3	5897.6±5.2
Unify-DAW	276.1±0.3	2891.3±1.8	15953.8±20.2	1692.8±7.5	5623.7±2.7
Unify-PAW+DAW	279.1±0.8	3075.4±2.5	16976.1±517.0	1885.7±18.9	6352.4±9.2
UCLSE	**334.7±0.4**	**3178.4±29.7**	**17143.5±714.4**	**2910.3±29.6**	**6648.7±10.8**

Table 5
The ablation study of the attention layer on the first, second and third convolutional layer respectively.

Attention Position	Breakout	Asterix	UpNDown
Conv1	284.5±0.1	**3075.1±3.3**	**16725.3±794.0**
Conv2	**293.0±0.1**	2980.2±63.0	15457.2±580.8
Conv3	154.7±1.1	2826.3±12.4	16356.0±153.2

The attention module, adding a two-layer neural network, causes around 0.2 million parameters and 0.3 million FLOPs, which accounts for less than 4% of the computation of the original network. The smoothing module has linear complexity given the number of states considered for the smoothing, but it only affects a part of the whole computation procedure. To alleviate the induced computation, we set the number of sampled transitions for the smoothing to 10. In this case, our approach increases around 11% of the total training time compared with the original method. Generally, the interaction between the agent and the environment consumes the most time.

We show the training times of different methods in Table 7. Compared with the baseline, ICM, our approach takes limited additional time, which is around 11% more time than ICM. However,

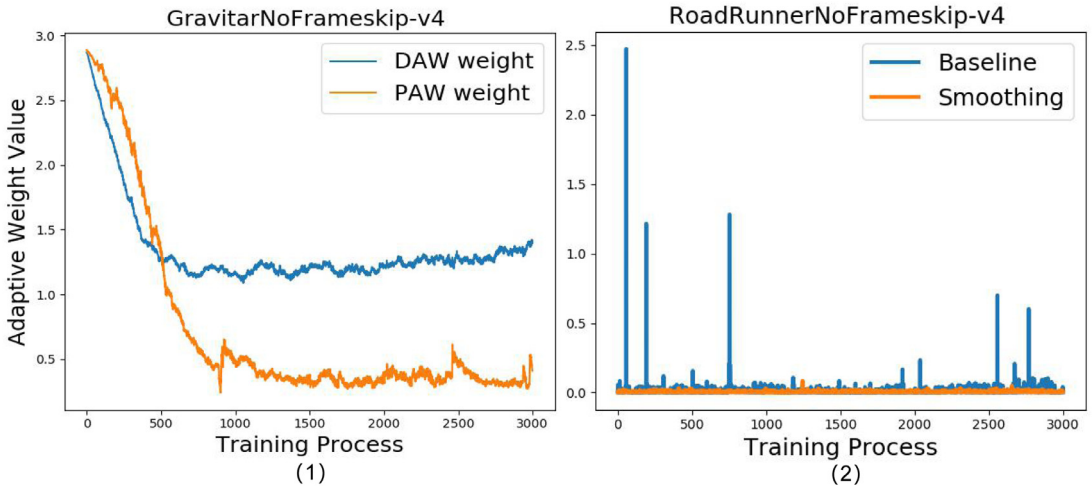

Fig. 11. (1) PAW weight and DAW weight decrease along with the training. (2) The comparison between the value function loss of the smoothing method and that of the baseline method during the training.

Table 6
The ablation study of the size of cluster, and the clustering is obtained with SimHash. The size of the fingerprint in SimHash is set 14, 16, 18 and, 20 respectively.

Cluster Size	Breakout	Asterix	UpNDown
14	**270.2±0.5**	**2349.3±3.6**	16204.3±97.8
16	266.4±0.1	1847.1±0.8	16719.1±75.3
18	225.2±0.1	2127.4±72.2	16665.7±31.0
20	168.1±0.2	1670.0±36.5	**16931.9±347.5**

Table 7
Training time of different methods (hours).

Games	ICM	RND	Count-based	NS-GA	NS-ES	UCLSE
Breakout	**31.13**	54.57	34.54	91.25	46.06	35.58
UpNDown	**33.02**	51.93	36.66	85.52	49.26	37.85
Asterix	**34.25**	57.95	38.02	88.67	47.83	36.32

compared with other methods, our approach is more efficient, and it only takes around 65% time of RND and around 41% time of novelty search.

6. Conclusion

In this paper, we propose a novel distribution-aware and policy-aware unified curiosity-driven learning framework. It has benefits as follows: first, it dynamically unifies state novelty and state-action novelty to enable the agent to explore the environment diversely and effectively; second, it alleviates the fluctuation of learning by introducing smoothness in the estimated intrinsic reward; third, it adopts an attention module to extract task-relevant features for reliable intrinsic reward estimation.

In practice, our method may suffer from unstable learning on some games, because it involves a lot of learning about these three different modules. In future, we will explore more approaches to improve the stability by effectively fusing different modules to reduce the learning burden.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Many thanks to Siyu Huang for his valuable comments and constructive suggestions on improving the paper's quality.

References

[1] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, C. Wu, Self-supervised discovering of interpretable features for reinforcement learning, in: IEEE Trans. Pattern Anal. Mach. Intell., 2020, pp. 1–12.
[2] W. Li, X. Wang, B. Jin, D. Luo, H. Zha, Structured cooperative reinforcement learning with time-varying composite action space, in: IEEE Trans. Pattern Anal. Mach. Intell., 2021, pp. 1–18.
[3] Z. Teng, B. Zhang, J. Fan, Three-step action search networks with deep q-learning for real-time object tracking, in: Pattern Recognit., volume 101, 2020, p. 107188.
[4] J. Yang, Y. Zhang, R. Feng, T. Zhang, W. Fan, Deep reinforcement hashing with redundancy elimination for effective image retrieval, in: Pattern Recognit., volume 100, 2020, p. 107116.
[5] M. Sun, J. Xiao, E.G. Lim, Y. Xie, J. Feng, Adaptive roi generation for video object segmentation using reinforcement learning, in: Pattern Recognit., volume 106, 2020, p. 107465.
[6] P. Zhao, D.L. Lee, How much novelty is relevant? it depends on your curiosity, in: Proc. Int. ACM SIGIR Conf. on Research and Devel. in Inf. Retrieval, 2016, pp. 315–324.
[7] G. Pang, A. van den Hengel, C. Shen, L. Cao, Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data, in: Proc. of ACM SIGKDD Conf. on Knowledge Discovery & Data Mining, 2021, pp. 1298–1308.
[8] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O.P. Abbeel, W. Zaremba, Hindsight experience replay, in: Proc. Advances Neural Inf. Process. Syst., 2017, pp. 5048–5058.
[9] R. Zhao, V. Tresp, Energy-based hindsight experience prioritization, in: Conf. on Robot Learn., 2018, pp. 113–122.
[10] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, in: Proc. Int. Conf. Learn. Representations, 2016.
[11] H. Zou, T. Ren, D. Yan, H. Su, J. Zhu, Reward shaping via meta-learning, in: arXiv preprint arXiv:1901.09330, 2019.
[12] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: Proc. Int. Conf. Mach. Learn., volume 99, 1999, pp. 278–287.
[13] S. Reddy, A.D. Dragan, S. Levine, Sqil: Imitation learning via reinforcement learning with sparse rewards, in: Proc. Int. Conf. Learn. Representations, 2019.
[14] A.Y. Ng, S.J. Russell, et al., Algorithms for inverse reinforcement learning, in: Proc. Int. Conf. Mach. Learn., volume 1, 2000, p. 2.
[15] B.D. Ziebart, A.L. Maas, J.A. Bagnell, A.K. Dey, Maximum entropy inverse reinforcement learning, in: Proc. AAAI Conf. Artif. Intell., volume 8, 2008, pp. 1433–1438.
[16] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, A.A. Efros, Large-scale study of curiosity-driven learning, in: Proc. Int. Conf. Learn. Representations, 2019a.
[17] Y. Burda, H. Edwards, A. Storkey, O. Klimov, Exploration by random network distillation, in: Proc. Int. Conf. Learn. Representations, 2019b.
[18] J. Schmidhuber, Formal theory of creativity, fun, and intrinsic motivation (1990–2010), in: IEEE Trans. on Autonomous Mental Development, volume 2, 2010, pp. 230–247.

[19] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, in: Proc. Advances Neural Inf. Process. Syst., 2016, pp. 1471–1479.
[20] B.C. Stadie, S. Levine, P. Abbeel, Incentivizing exploration in reinforcement learning with deep predictive models, in: Proc. Advances Neural Inf. Process. Syst. Workshop on Deep Reinforcement Learning, 2015.
[21] J. Schmidhuber, A possibility for implementing curiosity and boredom in model-building neural controllers, in: Proc. Int. Conf. Simul. Adapt. Behav. From Animals to Animals, 1991, pp. 222–227.
[22] R. Zhao, V. Tresp, Curiosity-driven experience prioritization via density estimation, in: Proc. Advances Neural Inf. Process. Syst., 2018.
[23] W. Li, F. Huang, X. Li, G. Pan, F. Wu, State distribution-aware sampling for deep q-learning, in: Neural Processing Letters, volume 50, 2019, pp. 1649–1660.
[24] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, J.T. Springenberg, Learning by playing solving sparse reward tasks from scratch, in: Proc. Int. Conf. Mach. Learn., PMLR, 2018, pp. 4344–4353.
[25] A.L. Strehl, M.L. Littman, An analysis of model-based interval estimation for markov decision processes, in: J. Computer Sys. Sciences, volume 74, 2008, pp. 1309–1331.
[26] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, in: Robot. and Automat. Syst., volume 57, 2009, pp. 469–483.
[27] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Proc. Advances Neural Inf. Process. Syst., 2016, pp. 4565–4573.
[28] T.D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, in: Proc. Advances Neural Inf. Process. Syst., 2016, pp. 3675–3683.
[29] A.S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, in: Proc. Int. Conf. Mach. Learn., PMLR, 2017, pp. 3540–3549.
[30] A. Levy, G. Konidaris, R. Platt, K. Saenko, Learning multi-level hierarchies with hindsight, in: Proc. Int. Conf. Learn. Representations, 2018.
[31] J. Randløv, P. Alstrøm, Learning to drive a bicycle using reinforcement learning and shaping, in: Proc. Int. Conf. Mach. Learn., volume 98, 1998, pp. 463–471.
[32] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: Proc. Int. Conf. Robot. and Automat., 2017, pp. 3389–3396.
[33] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proc. Int. Conf. Mach. Learn., 2009, pp. 41–48.
[34] S. Forestier, Y. Mollard, P.-Y. Oudeyer, Intrinsically motivated goal exploration processes with automatic curriculum learning, in: arXiv preprint arXiv:1708.02190, 2017.
[35] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, S. Levine, Divide-and-conquer reinforcement learning, in: Proc. Int. Conf. Learn. Representations, 2018.
[36] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al., Emergence of locomotion behaviours in rich environments, in: arXiv preprint arXiv:1707.02286, 2017.
[37] A.A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, R. Hadsell, Sim-to-real robot learning from pixels with progressive nets, in: Conf. on Robot Learn., 2017, pp. 262–270.
[38] G. Ostrovski, M.G. Bellemare, A. van den Oord, R. Munos, Count-based exploration with neural density models, in: Proc. Int. Conf. Mach. Learn., volume 70, 2017, pp. 2721–2730.
[39] H. Tang, R. Houthooft, D. Foote, A. Stooke, O.X. Chen, Y. Duan, J. Schulman, F. DeTurck, P. Abbeel, exploration: A study of count-based exploration for deep reinforcement learning, in: Proc. Advances Neural Inf. Process. Syst., 2017, pp. 2753–2762.
[40] T.L. Lai, H. Robbins, Asymptotically efficient adaptive allocation rules, in: Advances in Applied Math., volume 6, 1985, pp. 4–22.
[41] J. Lehman, K.O. Stanley, Novelty Search and the Problem with Objectives, in: Gene. Program. Theory and Pract. IX, 2011a, pp. 37–56.
[42] J. Lehman, K.O. Stanley, Abandoning objectives: Evolution through the search for novelty alone, in: Evolutionary Computation, volume 19, 2011b, pp. 189–223.
[43] J. Lehman, K.O. Stanley, Exploiting open-endedness to solve problems through the search for novelty, in: Artificial Life, volume 11, 2008, p. 329.
[44] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, in: Annu. Conf. Genetic Evol. Comput., 2017, pp. 145–152.
[45] A. Gajewski, J. Clune, K.O. Stanley, J. Lehman, Evolvability es: scalable and direct optimization of evolvability, in: Proc. Gene. and Evolut. Comput. Conf. Compan., 2019, pp. 107–115.
[46] J. Lehman, K.O. Stanley, Revising the evolutionary computation abstraction: minimal criteria novelty search, in: Proc. Gene. and Evolut. Comput. Conf. Compan., 2010a, pp. 103–110.
[47] J. Lehman, K.O. Stanley, Efficiently evolving programs through the search for novelty, in: Proc. Gene. and Evolut. Comput. Conf. Compan., 2010b, pp. 837–844.
[48] S. Risi, C.E. Hughes, K.O. Stanley, Evolving plastic neural networks with novelty search, in: Adaptive Behavior, volume 18, 2010, pp. 470–491.
[49] J.-B. Mouret, S. Doncieux, Encouraging behavioral diversity in evolutionary robotics: An empirical study, in: Evolutionary computation, volume 20, 2012, pp. 91–133.
[50] E. Conti, V. Madhavan, F.P. Such, J. Lehman, K.O. Stanley, J. Clune, Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents, in: Proc. Advances Neural Inf. Process. Syst., 2018, pp. 5032–5043.
[51] E.C. Jackson, M. Daley, Novelty search for deep reinforcement learning policy network weights by action sequence edit metric distance, in: Proc. Gene. and Evolut. Comput. Conf. Compan., 2019, pp. 173–174.
[52] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. Lillicrap, S. Gelly, Episodic curiosity through reachability, in: Proc. Int. Conf. Learn. Representations, 2019.
[53] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: IEEE Symposium on Founda. Comput. Science, 2006, pp. 459–468.
[54] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: Proc. Int. Conf. Comput. Vis. Theory and Applicat., volume 2, 2009, p. 2.
[55] X. Zhuang, Z. Chen, Strategy entropy as a measure of strategy convergence in reinforcement learning, in: Proc. Int. Conf. Intel. Netw. and Intel. Syst., 2008, pp. 81–84.
[56] X. Zhuang, The strategy entropy of reinforcement learning for mobile robot navigation in complex environments, in: Proc. Int. Conf. Robot. and Automat., 2005, pp. 1742–1747.
[57] L.v. d. Maaten, G. Hinton, Visualizing data using t-sne, in: J. Mach. Learn. Research, volume 9, 2008, pp. 2579–2605.
[58] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, A. Ignateva, Deep attention recurrent q-network, in: Proc. Advances Neural Inf. Process. Syst. Workshop, 2015.
[59] J. Choi, B.-J. Lee, B.-T. Zhang, Multi-focus attention network for efficient deep reinforcement learning, in: Proc. AAAI Conf. Artif. Intell. Workshop, 2017.
[60] R. Zhang, Attention guided imitation learning and reinforcement learning, in: Proc. AAAI Conf. Artif. Intell., volume 33, 2019, pp. 9906–9907.
[61] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep image prior, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9446–9454.
[62] A. Gaier, D. Ha, Weight agnostic neural networks, in: Proc. Advances Neural Inf. Process. Syst., 2019, pp. 5365–5379.
[63] M.G. Bellemare, Y. Naddaf, J. Veness, M.H. Bowling, The arcade learning environment: an evaluation platform for general agents, in: J. Artif. Intell. Research, volume 47, 2013, pp. 253–279.
[64] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, Go-explore: a new approach for hard-exploration problems, in: arXiv preprint arXiv:1901.10995, 2019.

Fuxian Huang received the B.S. degree in computer science and technology from Xidian University, Xi'an, ShaanXi, China, in 2016. He is currently pursuing the Ph.D. degree with the College of Computer Science, Zhejiang University, Hangzhou, China, under the supervision of Prof.

Xi Li His current research interests are primarily in machine learning and computer vision. Weichao Li received the B.S. and M.S. degrees in computer science and technology from Northwestern Polytechnical University, Xian, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the College of Computer Science, Zhejiang University, Hangzhou, China, under the supervision of Prof. Xi Li. His current research interests are primarily in computer vision and machine learning, especially, visual object tracking and reinforcement learning.

Jiabao Cui received the B.S. degree in computer science and technology from Xidian University, Xi'an, ShaanXi, China. He is currently studying for the PhD degree at the College of Computer Science at ZhejiangUniversity, Hangzhou, China. His research interests include computer vision, image classification, information retrieval and machine learning.

Yongjian Fu received the B.S. degree from school of Computer Science and Technology,Wuhan University, Wuhan, China, in 2018. He is currently a PhD student in College of Computer Science at Zhengjiang University, China. His advisors are Prof. Zhijie Pan and Prof. Xi Li. His current research interests are primarily in machine learning and image processing.

Xi Li is currently a full professor at Zhejiang University, China. Prior to that, he was a senior researcher at the University of Adelaide, Australia. From 2009 to 2010, he worked as a postdoctoral researcher at CNRS Telecom ParisTech, France. In 2009, he got the doctoral degree from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China. His research interests include visual tracking, motion analysis, face recognition, web data mining, image and video retrieval.