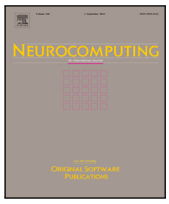
Neurocomputing 633 (2025) 129748

Contents lists available at ScienceDirect

Neurocomputing

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



Adaptive multi-model fusion learning for sparse-reward reinforcement learning

Giseung Park [a], Whiyoung Jung [b], Seungyul Han [c], Sungho Choi [a], Youngchul Sung [a,*]

[a] School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daehak-ro 291, 34141, Daejeon, South Korea
[b] LG AI Research, Magokjungang 10-ro 30, 07796, Seoul, South Korea
[c] Graduate School of Artificial Intelligence, Ulsan National Institute of Science and Technology, UNIST-gil 50, 44919, Ulsan, South Korea

ARTICLE INFO

Communicated by J. Andreu-Perez

Keywords:
Deep reinforcement learning
Neural network
Sparse-reward reinforcement learning
Intrinsic reward
Multiple prediction models
Adaptive fusion

ABSTRACT

In this paper, we address intrinsic reward generation for sparse-reward reinforcement learning, where the agent receives limited extrinsic feedback from the environment. Traditional approaches to intrinsic reward generation often rely on prediction errors from a single model, where the intrinsic reward is derived from the discrepancy between the model's predicted outputs and the actual targets. This approach exploits the observation that less-visited state–action pairs typically yield higher prediction errors. We extend this framework by incorporating multiple prediction models and propose an adaptive fusion technique specifically designed for the multi-model setting. We establish and mathematically justify key axiomatic conditions that any viable fusion method must satisfy. Our adaptive fusion approach dynamically learns the best way to combine prediction errors during training, leading to improved learning performance. Numerical experiments validate the effectiveness of our method, showing significant performance gains across various tasks compared to existing approaches.

1. Introduction

Artificial intelligence (AI) has become increasingly prevalent in a wide range of real-world applications [1–5]. Among various AI techniques, reinforcement learning (RL) has garnered significant attention, particularly with the integration of neural networks. This integration has not only improved the overall learning performance of RL methods [6,7] but also broadened their applicability [8,9], surpassing traditional approaches [10,11]. Deep RL, leveraging the expressive capabilities of neural networks [12,13], has extended its scope to encompass domains such as multi-agent RL [14,15], multi-objective RL [16–18], and meta-RL [19].

One area of particular interest within the realm of Deep RL is the handling of sparse rewards [20–24]. Sparse-reward RL scenarios involve environments where non-trivial rewards are provided only under specific conditions, rather than for every action taken. This challenge is commonly encountered in various real-world tasks, such as robotic manipulation or the control of unmanned vehicles [20,23,25–27].

Exploration remains a critical aspect of sparse-reward RL as it is essential for achieving optimal performance when rewards are infrequently given to the agent. In recent years, to address the scarcity of extrinsic rewards, numerous algorithms have been developed to enhance exploration in sparse-reward settings by enabling the agent to generate *intrinsic rewards* in addition to extrinsic rewards, using various concepts such as curiosity, surprise, or random network distillation [25,28–30].

In those intrinsic reward generation methods, agents employ neural network models to estimate target values or distributions, such as the next state or the transition probability distribution within the environment. These intrinsic rewards are designed as the discrepancy between the model's prediction and the target [25,28–30]. This design stems from the tendency that less-visited or unvisited state–action pairs result in larger prediction errors, while frequently-visited pairs yield smaller errors.

Most previous methods rely on a single neural network prediction model to estimate target values or distributions. However, single neural networks can struggle to accurately capture the underlying dynamics, particularly when training data is limited [31,32]. Therefore, relying on a single neural network model can also be insufficient in RL. A straightforward way to address this issue is to use multiple models, which provide more reliable estimates for capturing the uncertainty of the underlying system [31,32]. While prior work has utilized multiple models [33], the fusion rule in this method is fixed and is not explicitly designed to maximize the cumulative extrinsic return.

* Corresponding author.
E-mail addresses: gs.park@kaist.ac.kr (G. Park), whiyoung.jung@lgresearch.ai (W. Jung), syhan@unist.ac.kr (S. Han), sungho.choi@kaist.ac.kr (S. Choi), ycsung@kaist.ac.kr (Y. Sung).

https://doi.org/10.1016/j.neucom.2025.129748
Received 19 April 2024; Received in revised form 12 February 2025; Accepted 15 February 2025
Available online 26 February 2025

In this paper, we introduce a novel framework for adaptively fusing multiple values from neural network models that directly align to maximize the cumulative extrinsic return. The crucial aspect lies in designing an effective fusion rule to optimize performance throughout the learning process. One possible approach to adaptive fusion could involve a high-level neural network architecture, similar to a hypernetwork [34], that combines the outputs of the multiple models. However, as we shall see in Section 5.1.1, our method leverages inductive bias in the fusion rule, significantly reducing the number of control parameters compared to a full hypernetwork, enhancing learning efficiency, and improving RL performance.

Our contributions to this multi-model problem are as follows:

- We establish axiomatic conditions on the fusion rule that any legitimate fusion function should satisfy. This imposition of an inductive bias is mathematically justified and enhances fusion learning efficiency by narrowing the search space to a single-parameter search.
- We propose a method to learn the fusion rule at each stage of policy learning. This approach aims to maximize the cumulative return of extrinsic reward through meta-gradient optimization.
- Our empirical results demonstrate that our multi-model intrinsic reward generation with fusion learning outperforms existing intrinsic reward generation methods.

2. Related work

Intrinsically motivated RL and exploration methods are generally categorized into two main groups. The first category involves the explicit generation of intrinsic rewards, where the agent is trained using the sum of extrinsic rewards and appropriately scaled intrinsic rewards. The second one includes indirect methods that enhance exploration without explicitly generating intrinsic rewards. Our work belongs to the first category.

2.1. First category: explicit intrinsic reward generation

In the context of explicit intrinsic reward generation, various approaches encourage agents to explore less frequently visited states. [25] leveraged the concept of curiosity, using information gain on the prediction model to generate additional rewards. [21] addressed high-dimensional state space exploration by mapping state features into a hash table using count-based exploration. [29] introduced intrinsic rewards based on the concept of surprise. [28] defined intrinsic rewards using prediction errors in a feature state space, and [22] incorporated the idea of homeostasis into their work. [35] proposed a method involving training a module to generate intrinsic rewards separately from the policy, using a delayed reward environment, distinct from previous threshold-based sparse-reward environments [25]. [30] introduced Random Network Distillation (RND), a method that employs a randomly initialized and fixed neural network as the target for another prediction network. The intrinsic reward in this framework is derived from the prediction error between the target and the predictor outputs.

While these approaches rely on a single neural network prediction model to measure visitation frequency, single networks are often prone to inaccurately capture the underlying dynamics when the learning data is not sufficient [31,32]. This issue arises from the constraints of finite datasets, leading to persistent inaccuracies during RL and reducing the effectiveness of single models in generating appropriate intrinsic rewards. A practical solution is to use multiple models, which provide more reliable estimates [31,32]. For example, [33] interpreted the disagreement among multiple models as the variance in predicted next states and used this variance as a differentiable intrinsic reward. However, their method is not specifically designed to maximize the cumulative extrinsic return, and the fusion rule remains fixed throughout the learning process, potentially limiting dynamic adaptation. In contrast, we propose a novel framework that adaptively fuses multiple outputs from neural network models, ensuring alignment to maximize the cumulative extrinsic return.

Orthogonal to the methods in the first category, several other recent works focus on generating intrinsic rewards based solely on states, ensuring that the cumulative reward over a certain period is proportional to state entropy [36–38]. These methods are grounded in state entropy maximization, and the concept primarily targets scenarios where extrinsic rewards are absent.

2.2. Second category: exploration in the absence of intrinsic reward generation

Recent indirect methods are classified mainly into two groups: (i) revising the original objective function to stimulate exploration, exploiting intrinsic motivation implicitly, and (ii) perturbing the parameter space of the considered policy.

In the first group, [20] proposed the idea of sampling additional states from the replay buffer to use as new goals in sparse and binary extrinsic reward environments, where their policy considered both state and goal inputs. In contrast, our approach does not require the concept of a goal. [23] suggested exploration by leveraging novel state–action pairs from the past, especially in sparse-reward environments, through the delay of extrinsic rewards. [39] modified the original training objective by emphasizing the maximization of the divergence between the current policy and recent policies, employing an adaptive scaling technique. The PPO algorithm incorporated the dropout concept to promote episodic stochastic agent behavior [40]. [41] introduced a novel exploratory policy, utilizing a convex combination of the target policy and any given policy, effectively addressing exploration in a manner that extends traditional methods such as ϵ-greedy or Gaussian noise by solving surrogate MDPs.

In the second group, [42] introduced a goal-based exploration approach for continuous control, involving the generation of parameter-outcome pairs and perturbing specific parameters based on randomly selected goals from the outcome space. [43] proposed an algorithm for parameter distribution optimization to enhance exploration by introducing uncertainty in neural network weights while clipping two consecutive parameter updates to improve stability. Taking inspiration from [31,44] explored pure exploration MDPs without any extrinsic rewards, using the concept of utility. The utility in their approach is based on JSD and Jensen–Rényi divergence [45], and they considered various models for transition functions, using them to compute utility based on the average entropy of multiple models.

3. Background

3.1. Setup

We consider a discrete-time continuous-state Markov Decision Process (MDP), denoted as $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, respectively, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition probability (density), $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *extrinsic* reward function, ρ_0 is the distribution of the initial state, and γ is the discounting factor. A (stochastic) policy $\pi(a|s)$ represents the probability (density) of choosing action $a \in \mathcal{A}$ for given state $s \in \mathcal{S}$. In sparse-reward RL, the environment returns a non-trivial reward only when certain conditions are met by the current state, the action and the next state [20,23,25]. Our goal is to optimize the policy π to maximize the expected return $J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, \tau_0 \sim \pi}[\sum_{t=0}^{\infty} \gamma^t r_t]$ by properly generating intrinsic reward in such a sparse-reward environment, where $\tau_t := (s_t, a_t, s_{t+1}, a_{t+1}, \ldots)$ and $r_t = r(s_t, a_t, s_{t+1})$ is the extrinsic reward at time t. We assume that the true transition distribution P is unknown to the agent.

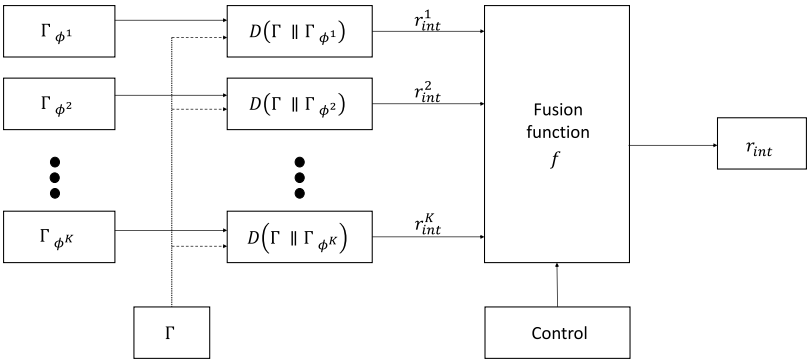


Fig. 1. Adaptive fusion of K prediction errors from the multiple models. (Γ represents the target and Γ_{ϕ^i} represents the ith prediction model.).

3.2. Intrinsic reward design based on model prediction error

In the model-prediction-error-based intrinsic reward design, the agent has a prediction model parameterized by ϕ for a specific target value or distribution, and the intrinsic reward is designed as the error between the target and the model prediction [22,25,28–30]. The intrinsic-reward-incorporated problem is formulated to maximize the cumulative return $J_{total}(\pi) = \mathbb{E}_{s_0\sim\rho_0,\tau_0\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r_t^{total}]$ with

$$r_t^{total} = r_t(s_t, a_t, s_{t+1}) + \beta\ D(\Gamma||\Gamma_\phi|(s_t, a_t, s_{t+1})) \tag{1}$$

for some $\beta > 0$ and some divergence function $D(\cdot \parallel \cdot)$, where Γ is the desired target value or distribution and Γ_ϕ is the output of the prediction model parameterized by ϕ estimating the target Γ. For the divergence D, one can use the mean squared error (MSE) when the target is a value or the Kullback–Leibler divergence (KLD) when the target is a distribution.

4. Method

4.1. The proposed adaptive fusion learning

We consider the use of multiple prediction models and the design of prediction-error-based intrinsic reward from these multiple models. Suppose we have a collection of $K(\geq 2)$ models parameterized by $\phi^1, \dots, \phi^K$ to generate K prediction error values at timestep t: $r_{t,int}^j(s_t, a_t, s_{t+1}) = D(\Gamma||\Gamma_{\phi^j}|(s_t, a_t, s_{t+1})), j = 1, \dots, K$, one by each model. The key problem of multi-model prediction-error-based intrinsic reward design is how to optimally fuse the K values $r_{t,int}^j(s_t, a_t, s_{t+1})$, $j = 1, \dots, K$, to generate a single intrinsic reward to be added to the scalar extrinsic reward r_t for policy update.

The considered multi-model fusion structure is shown in Fig. 1. In order to fuse the K values for a single intrinsic reward value, one can use some known method such as average, minimum, or maximum. However, *there is no guarantee of optimality for such an arbitrary choice. Furthermore, one fixed fusion rule may not be optimal for the entire learning phase.* Thus, we consider learning the fusion rule together with policy parameter learning.

Let the fusion function be denoted as

$$r_{int} = f(r_{int}^1, r_{int}^2, \dots, r_{int}^K), \tag{2}$$

where $r_{int}^1, r_{int}^2, \dots, r_{int}^K$ are the K input values from the K prediction models, and r_{int} is the output of the fusion function. To learn the optimal fusion rule yielding the maximum cumulative return of extrinsic reward, one can consider implementing the function f by using a neural network based on the universal approximation theorem [12] and learning the parameters of the deep neural network with some loss function. (We will compare this method to our proposed method in Section 5.1.1.)

However, we do not apply this direct learning method but exploit some necessary conditions that any legitimate fusion function should satisfy. By imposing such inductive bias, we simplify the fusion structure and corresponding learning, which yields a very efficient way of fusion learning. Our inductive bias on the fusion function is based on axiomatic conditions for a legitimate fusion function. The imposed requirements for the fusion function f are as follows.

Condition 1. *The fusion function f varies with some control parameter to adapt to the relative importance of the K input values.*

We require Condition 1 so that the fusion of the K input values adapts to the learning situation. In other words, we want this adaptation to be learned over time based on data to yield the maximum cumulative return of extrinsic reward.

In addition, we impose the following condition for the fusion function:

Condition 2. *The fusion function f is homogeneous, i.e.,*

$$f(cr_{int}^1, cr_{int}^2, \dots, cr_{int}^K) = c\,f(r_{int}^1, r_{int}^2, \dots, r_{int}^K). \tag{3}$$

Condition 2 implies that if we scale all the input values by the same factor c, then the fusion output is the c-scaled version of the fusion output of the non-scaled inputs. Condition 2 is a proper requirement for a legitimate fusion function, considered in generalized mean [46]. The reason why we require Condition 2 is mathematically explained in detail in Appendix A.

The unique fusion function that satisfies Conditions 1 and 2 is found based on the α-mean of positive measures in the field of information geometry [47]. For any K positive[1] values $x_1, \dots, x_K > 0$, the α-mean of $x_1, \dots, x_K$ is defined as [47]

$$f_\alpha(x_1, \dots, x_K) = h^{-1}\left(\frac{1}{K}\sum_{i=1}^{K} h(x_i)\right) \tag{4}$$

where $h(x)$ is given by the α-embedding transformation:

$$h(x) = \begin{cases} x^{\frac{1-\alpha}{2}}, & \text{if } \alpha \neq 1 \\ \log x, & \text{if } \alpha = 1. \end{cases} \tag{5}$$

It is proven that the unique class of fusion functions satisfying Condition 2 under the twice-differentiability and the strict monotonicity is given by the α-mean [47,48]. In the proof, Condition 2 is used to write $f_\alpha(cx_1, \dots, cx_K) = h^{-1}\left(\frac{1}{K}\sum_{i=1}^{K} h(cx_i)\right) = c\,f_\alpha(x_1, \dots, x_K)$. Taking $h(\cdot)$ on both sides yields $h(cf_\alpha(x_1, \dots, x_K)) = \frac{1}{K}\sum_{i=1}^{K} h(cx_i)$. Then, taking partial derivative with respect to x_i $(1 \leq i \leq K)$ on both sides, we show that Eq. (5) is the unique class of mapping functions.

[1] When an input value to the α-mean is negative due to divergence approximation in some cases, we use exponentiation $\exp(-x)$ at the input stage with input x and negative logarithm at the output stage as the inverse of input exponentiation.

Note that the α-mean has a single scalar parameter α to control the fusion. This α gives us a tool to satisfy Condition 1. By varying α, the α-mean includes all numeric fusions satisfying the homogeneity condition in Condition 2 such as minimum, maximum, and conventional mean functions such as arithmetic, geometric, and harmonic mean [47]. When $\alpha = -\infty$, $f_\alpha(x_1, \dots, x_K) = \max_i x_i$. On the other hand, when $\alpha = \infty$, $f_\alpha(x_1, \dots, x_K) = \min_i x_i$. As α increases from $-\infty$ to ∞, the α-mean output varies monotonically from maximum to minimum. Hence, we perform aggressive fusion to conservative fusion by controlling the parameter α.

Furthermore, another desired property of *permutation invariance* for any legitimate fusion function is automatically satisfied by the α-mean due to its structure in (4).

4.2. Learning of α with meta-gradient optimization

By imposing the inductive bias of Condition 2 on the fusion function, we have reduced the number of control parameters to only one. Now, we need to adaptively control α to maximize the expected cumulative extrinsic return $J(\pi)$. To learn α maximizing $J(\pi)$, we apply the meta gradient method [35,49,50]. It will be shown that the updated α with the proposed method varies according to the stage of learning depending on the task. For a policy π_θ with policy parameter θ, let us define the following quantities.

- $J(\pi_\theta) = \mathbb{E}_{\tau_0 \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$: the expected sum of extrinsic rewards which we actually want to maximize.
- $J_{total}(\pi_\theta) = \mathbb{E}_{\tau_0 \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t^{total}\right]$: the expected sum of the total reward which is given by

$$r_t^{total}(s_t, a_t, s_{t+1}) = r_t + \beta f_\alpha(\{r_{t,int}^j(s_t, a_t, s_{t+1})\}_{j=1}^K) \tag{6}$$

with which the policy π_θ is updated, where $\beta > 0$.

Then, for a given sample trajectory generated by π_θ, applying policy gradient, we update θ towards the direction of maximizing $J_{total}(\pi_\theta)$:

$$\tilde{\theta} = \theta + \delta_\theta \nabla_\theta J_{total}(\pi_\theta) \tag{7}$$

where δ_θ is the learning rate for θ. Then, the fusion parameter α is updated to maximize the expected sum of only extrinsic rewards for the updated policy $\pi_{\tilde{\theta}}$:

$$\tilde{\alpha} = \alpha + \delta_\alpha \nabla_\alpha J(\pi_{\tilde{\theta}}) \tag{8}$$

where δ_α is the learning rate for α. Note that we update the policy parameter θ to maximize $J_{total}(\pi_\theta)$. Since $J_{total}(\pi_\theta)$ is a function of α as seen in its definition above, the updated policy parameter $\tilde{\theta}$ is a function of α. Therefore, $\nabla_\alpha J(\pi_{\tilde{\theta}})$ in (8) is computed by the chain rule as

$$\nabla_\alpha J(\pi_{\tilde{\theta}}) = \nabla_{\tilde{\theta}} J(\pi_{\tilde{\theta}})\ \nabla_\alpha \tilde{\theta}. \tag{9}$$

To jointly learn α together with θ, we adopt an alternating optimization based on (7) and (8), widely used in meta-parameter optimization [35,49,50]. That is, we iterate the policy update by (7) and the fusion parameter α update by (8) in an alternating manner. In this way, we learn proper α adaptively over timesteps to maximize the extrinsic return performance. We set the initial value of α to zero for all the considered experiments.

4.3. Implementation

While the presented method for generating intrinsic rewards can be integrated with various RL algorithms, in this context, we focus on Proximal Policy Optimization (PPO) [51], a widely used on-policy algorithm. In addition to the usual value function estimating $J_{total}(\pi_\theta)$ in (7) in PPO, we require an extrinsic value function $V_\zeta(s)$ which does not consider intrinsic reward to estimate $J(\pi_{\tilde{\theta}})$ in (8) using the generalized advantage estimation (GAE) technique [35,52]. ζ is learned with a similar procedure to that of the fusion parameter α in Section 4.2. Implementation details are provided in Appendix C, and the source code is provided in https://drive.google.com/file/d/1RoAfD6YhzdSP0jysZaHSa6Pc_kxxmo1-/view?usp=sharing.

For the prediction-based intrinsic reward generation method, we consider the following two examples.

4.3.1. Multi-model random network distillation (RND)

First, we consider generalization of RND [30] to the multi-model case. In this case, we have a fixed target network g^{tar} generating the target value and K prediction networks $g_{\phi^j}^{pre}$, $j = 1, \dots, K$ all predicting the output of g^{tar}. The divergence reduces to the Euclidean distance. All prediction models are randomly initialized independently. The jth prediction network $g_{\phi^j}^{pre}$ is independently trained to minimize $\sum_t \|g^{tar}(s_{t+1}) - g_{\phi^j}^{pre}(s_{t+1})\|^2$ with different order of mini-batched samples, where $g^{tar}(s_{t+1})$ and $g_{\phi^j}^{pre}(s_{t+1})$ are the outputs of the target network and the jth prediction network with input s_{t+1}, respectively. The jth intrinsic reward candidate at timestep t is formulated as the prediction error of the jth model, i.e., $r_{t,int}^j(s_t, a_t, s_{t+1}) = \|g^{tar}(s_{t+1}) - g_{\phi^j}^{pre}(s_{t+1})\|^2$. The pseudocode is provided as shown in Algorithm 1.

Algorithm 1 Fusion Learning with Multiple Prediction Models for Intrinsic Reward Generation: Multi-model RND.

1: L : batch size for policy training.
2: L_{mini} : minibatch size for policy training, L'_{mini} : minibatch size for model training.
3: N : epoch size for policy training, N' : epoch size for model training.
4: MAX : the maximum index of batch period l, K : the number of prediction models.
5: Initialize the policy π_{θ_0}, the fixed target network g^{tar}, the K prediction networks $g_{\phi_0^1}^{pre}, \cdots, g_{\phi_0^K}^{pre}$, the fusion parameter α, and the extrinsic value function parameter ζ.
6: **for** Batch period $l = 0, \cdots, MAX - 1$ **do**
7: **for** Timestep $t = lL, lL+1, \cdots, lL+L-1$ **do**
8: Collect (s_t, a_t, r_t, s_{t+1}) in the current batch D.
9: **end for**
10: Train each $g_{\phi_l^j}^{pre}$, $j = 1, \cdots, K$ by minimizing $\sum_t \|g^{tar}(s_{t+1}) - g_{\phi^j}^{pre}(s_{t+1})\|^2$ with different order of mini-batched samples in D of size L. For each $g_{\phi_l^j}^{pre}$, we perform the minimization updates with minibatches of size L'_{mini} drawn from D for N' epochs.
11: Acquire the intrinsic reward $r_{t,int}$ of the current batch D of size L.
12: Train π_{θ_l} by using PPO with the total rewards (6) and minibatch size L_{mini} for N epochs.
13: Train α and ζ by using the parameter learning method described in Section 4.2 with the total rewards (6) and minibatch size L_{mini} for N epochs.
14: Empty the current batch D.
15: **end for**

4.3.2. Prediction of the distribution of the next state

This case corresponds to a multi-model extension of [29]. In this case, the target is the true distribution of the next state $P(s_{t+1}|s_t, a_t)$, and we have K prediction models $P_{\phi^1}, \dots, P_{\phi^K}$ for P. The error of the jth prediction model P_{ϕ^j} is given by [29]

$$\begin{aligned} D_{KL}(P||P_{\phi^j})|(s_t, a_t) &= \mathbb{E}_P\left[\log \frac{P(\cdot|s_t, a_t)}{P'(\cdot|s_t, a_t)} \frac{P'(\cdot|s_t, a_t)}{P_{\phi^j}(\cdot|s_t, a_t)}\right] \\ &\geq \mathbb{E}_P\left[\log \frac{P'(\cdot|s_t, a_t)}{P_{\phi^j}(\cdot|s_t, a_t)}\right] \end{aligned} \tag{10}$$

where P' is an arbitrary probability distribution. The lower bound (10) is valid since $\mathbb{E}_P\left[\log \frac{P(\cdot|s_t,a_t)}{P'(\cdot|s_t,a_t)}\right] = D_{KL}(P \parallel P') \geq 0$. In order to

obtain a tight lower bound, P' should be learned to be close to the true distribution P. To increase degree-of-freedom for better learning and estimation, we use the mixture distribution of $P_K = \sum_{i=1}^{K} q_i P_{\phi^i}$ for P' with the learnable mixing coefficients $q_i \geq 0$ and $\sum_{i=1}^{K} q_i = 1$. Then, the intrinsic reward from the jth model P_{ϕ^j} at timestep t is formulated as $r^j_{t,int}(s_t, a_t, s_{t+1}) = \log \frac{P_K(s_{t+1}|s_t,a_t)}{P_{\phi^j}(s_{t+1}|s_t,a_t)}$, $j = 1, \ldots, K$. Note that $r^j_{t,int}$ can be negative in this case although the KLD is always nonnegative. We use input exponentiation and output logarithm for the fusion function, mentioned in the footnote 1. In the case of the K prediction models, denoted as $P_{\phi^1}, \ldots, P_{\phi^K}$, we employ the fully-factorized Gaussian distribution [25]. Consequently, P_K represents a class of K-modal Gaussian mixture distributions.

We update the prediction models $P_{\phi^1}, \ldots, P_{\phi^K}$ and the corresponding mixing coefficients $q_1, \ldots, q_K$ through the following process. Initially, the parameters $\phi^1, \ldots, \phi^K$ are independently initialized, and the mixing coefficients q_i are set to $\frac{1}{K}$ for all $i = 1, \ldots, K$. During each batch period of PPO, to jointly optimize ϕ^i and q_i, we employ maximum-likelihood estimation with an L_2-norm regularization while adhering to KL constraints [29,53]:

$$\begin{aligned} \underset{\phi^i,\ q_i,\ 1\leq i\leq K}{\text{maximize}} \quad & \underbrace{\mathbb{E}_{(s,a,s')} \log \left\{ \sum_{i=1}^{K} q_i P_{\phi^i}(s'|s,a) \right\}}_{=:\mathcal{L}_{\text{likelihood}}} - c_{\text{reg}} \underbrace{\sum_{i=1}^{K} \|\phi^i\|^2}_{=:\mathcal{L}_{reg}} \\ \text{subject to} \quad & \mathbb{E}_{(s,a)} \left[D_{KL}(P_{\phi^i} \parallel P_{\phi^i_{\text{old}}})(s,a) \right] \leq \kappa, \ \sum_{i=1}^{K} q_i = 1 \end{aligned} \tag{11}$$

where ϕ^i_{old} represents the parameter of the ith model before the update triggered by (11), c_{reg} is the regularization coefficient, and κ is a positive constant. To address this optimization problem concerning ϕ^i, we employ a method based on a second-order approximation [54].

For the update of $\{q_i\}$, we apply the EM method proposed in [55] and set q_i as

$$q_i = \mathbb{E}_{(s,a,s')} \frac{q_i^{\text{old}} P_{\phi^i}(s'|s,a)}{\sum_{j=1}^{K} q_j^{\text{old}} P_{\phi^j}(s'|s,a)} \quad (1 \leq i \leq K) \tag{12}$$

where q_i^{old} is the mixing coefficient of the ith model before the update caused by (12). To ensure numerical stability, we apply the "log-sum-exp" trick when calculating (12), as well as when working with $\mathcal{L}_{\text{likelihood}}$ as defined in (11), and computing the gradient $\nabla\phi^i \mathcal{L}_{\text{likelihood}}$. In addition, we apply simultaneous update of all ϕ^i's and q_i's, which was discovered to yield superior performance compared to the alternative method of updating the K models one by one. The pseudocode is provided as shown in Algorithm 2 in Appendix B.

5. Experimental results

In Section 5.1, we first show that the proposed fusion learning is more efficient than directly using a deep neural network as an adaptive fusion function, producing outperforming performance with a single learnable parameter α. In Section 5.2, we will show the performance comparison between the proposed method and the considered baselines in various sparse-reward environments: exploration tasks 5.2.1, continuous control with delayed reward setting 5.2.2, and practical sparse-reward tasks including robotic arm control and drone control in noisy environments 5.2.3.

5.1. Study on the efficiency of the proposed fusion learning

5.1.1. Comparison with direct neural network-based fusion learning

In the proposed fusion scheme, we drastically reduced the search space for adaptive fusion function by imposing our inductive bias of Condition 2, which is considered to be a necessary condition for any legitimate fusion function as explained in Appendix A, and proposed the α-mean-based fusion learning with a single control parameter α. One can implement the fusion function in Fig. 1 directly by using a widely-used function approximator, deep neural network, to learn adaptive fusion weights with the same meta-gradient optimization as we used.

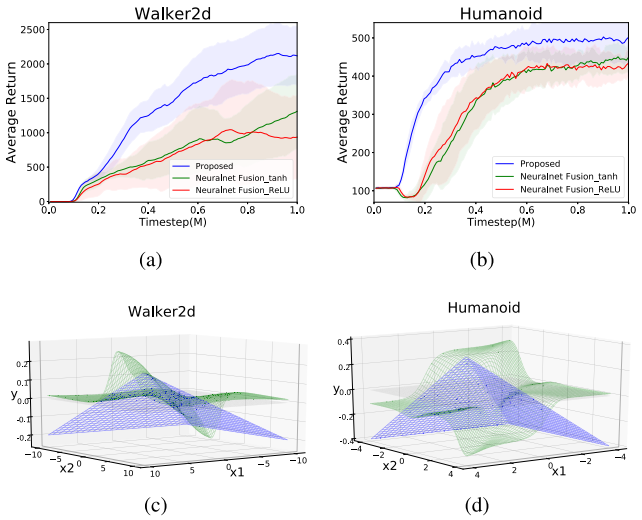


Fig. 2. Comparison between the proposed α-fusion learning and the fusion with deep neural network learning: (a) average return - DelayedWalker2d, (b) average return — DelayedHumanoid. On the y-axes, we illustrate the mean value of extrinsic returns obtained from the most recent one hundred episodes, averaged across the ten random seeds. (c) Learned fusion function graph — DelayedWalker2d (proposed — blue, neural network (tanh) - green, (x_1, x_2) plane — gray), and (d) learned fusion function graph — DelayedHumanoid.

To check the effectiveness of the proposed α-fusion learning over the fusion with deep neural network learning, we actually designed a neural network fusion function $f_\xi(x_1, \ldots, x_K)$ with $K = 2$ inputs with hidden layer number 1, 2, and 3 using a nonlinear (tanh or ReLU) activation, followed by a linear output layer of size 1. The size of every hidden layer of f_ξ is $2K$, and we chose the best hidden layer number for each environment. f_ξ is trained to maximize the cumulative return, as described in Section 4.2.

Figs. 2(a) and 2(b) show the comparison results in two Mujoco tasks: Walker2d and Humanoid. (To create a sparse-reward environment, we implemented the delay method as described in [23]. Detail for this delayed setup will be explained in Section 5.2.2. We adopted Algorithm 2 for the implementation in these environments.) It is seen that the proposed fusion learning based on the single parameter α significantly outperforms the fusion learning directly based on neural network. Figs. 2(c) and 2(d) show the graphs of the proposed fusion function $y = f_\alpha(x_1, x_2)$ (blue surface) and the neural-network-based fusion function $y = f_\xi(x_1, x_2)$ with tanh activation (green surface) after training in Walker2d and Humanoid, respectively. For the neural network-based fusion, Fig. 2(c) shows that two negative inputs give a positive output, and Fig. 2(d) shows that two positive inputs give a negative output. Both observations deviate our intuition. In addition, the learned fusion function by neural network is not symmetric around the line $x_1 = x_2$, so it is not permutation-invariant.

Note that our fusion output with a learnable fusion parameter α and a weighting factor $\beta > 0$ for the intrinsic reward part in (6) is expressed as $f_\alpha(x_1, x_2) = -\frac{2\beta}{1-\alpha} \log[\frac{\exp(-\frac{1-\alpha}{2} x_1) + \exp(-\frac{1-\alpha}{2} x_2)}{2}]$ in the case of input exponentiation and output logarithm used to handle possibly negative inputs. This nonlinear function is mathematically simple but require nontrivial complexity and learning time to be implemented with a deep neural network as seen in this example. The proposed adaptive α-fusion structure indeed captures the legitimate fusion behavior by using only a single parameter α, and the corresponding learning is efficient. The proposed α-fusion structure seems to be a useful built-in inductive bias [56] for fusion learning.

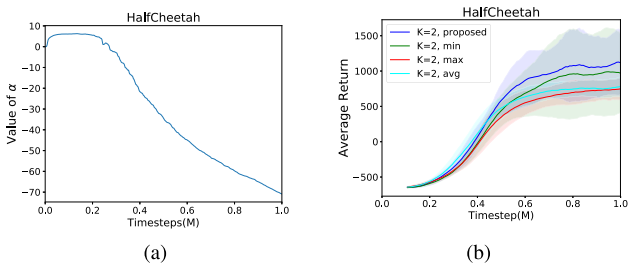


Fig. 3. Comparison with fixed fusion rules — min, max, and average: (a) Learning curve of during the proposed fusion learning in HalfCheetah and (b) Performance comparison with fixed fusion methods in HalfCheetah. On the y-axis, we illustrate the mean value of extrinsic returns obtained from the most recent one hundred episodes, averaged across the ten random seeds.

5.1.2. Learning behavior of fusion parameter α

With the verification of the effectiveness of our fusion structure, we then investigated how the fusion parameter α changed adaptively during the training. Fig. 3(a) shows the learning curve of the fusion parameter α in delayed HalfCheetah. In the case of prediction of the distribution of the next state, the KL divergence is approximated by its lower bound as seen in (10). Hence, the exponentiation $\exp(-x)$ at the input stage and the negative logarithm at the output stage to the α-fusion were applied. Hence, smaller α means less aggressive fusion towards minimum operation. It is seen that starting from the initial value $\alpha = 0$, the fusion parameter α increases until it reaches approximately 5, maintains the level until approximately 0.25 million timesteps, and then decreases monotonically. Thus, the fusion learning takes relatively more aggressive fusion strategies with α being around 5 (but this is not the too aggressive maximum corresponding to $\alpha = \infty$) in the early stage of learning. Then, the fusion learning takes more and more conservative fusion strategy by decreasing α more and more to large negative values (i.e., towards minimum taking) in the cased delayed HalfCheetah.

We compared the proposed multi-model method with adaptive fusion learning with the fixed fusion rules: minimum, maximum, and average taking in the case of $K = 2$. The result is shown in Fig. 3(b). It is seen that in the fixed fusion case, the method using average has higher performance than that with minimum or maximum in the early stage of training. However, the minimum selection method yields better performance than average or maximum at the later stage. It is seen that the proposed adaptive fusion yields the best performance because the proposed adaptive fusion takes advantage of both fast performance improvement in the early stage and high final performance at the end by learning α optimally over the learning stages.

5.2. Performance comparison

5.2.1. Exploration tasks

We conducted experiments on two exploration tasks: (i) a continuous 4-room maze environment for pure exploration task (i.e., no extrinsic reward is given to the agent) and (ii) the OpenAI Gym Acrobot environment [57] for exploration with extrinsic reward. For the 4-room maze, we modified the grid map environment at https://github.com/huyaoyu/GridMap. The proposed method uses the multi-model RND described in Algorithm 1 and we check whether the proposed fusion learning improves performance over RND with a single prediction network ($K = 1$), and how the fusion parameter α behaves over the learning phase.

Fig. 4(a) shows the considered 4-room maze of size $(100, 100)$. The (x, y) position of the agent is the state. The agent starts from the position $(0.5, 0.5)$ near the lower-left corner of the maze. At each timestep, the agent performs its two-dimensional continuous action (dx, dy) bounded in $[-1, 1]^2$. The next state is $(x + dx, y + dy)$, but the agent is given

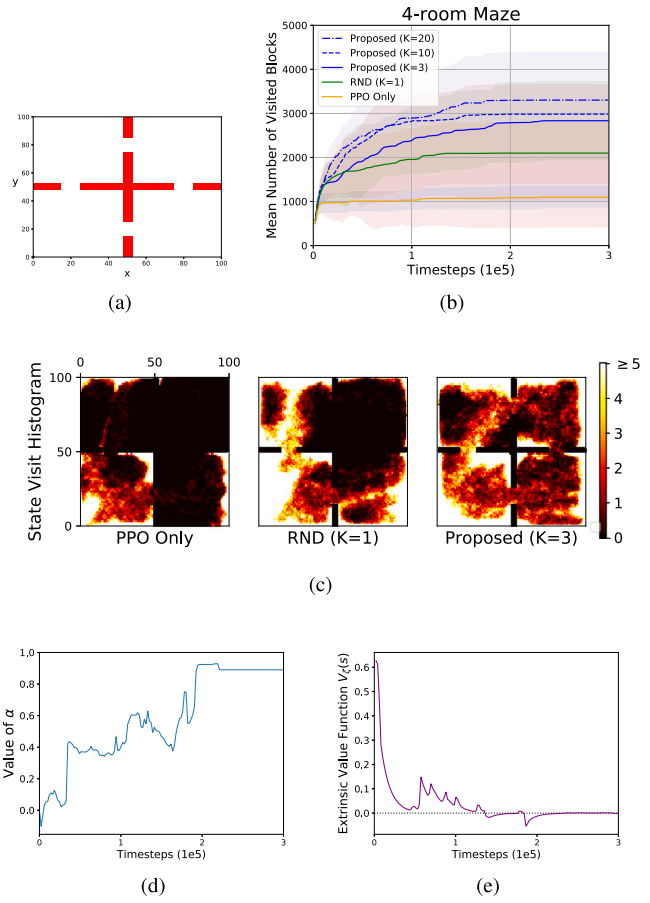
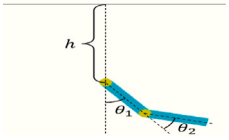


Fig. 4. (a) Continuous 4-room maze for pure exploration task (with no extrinsic reward). (b) Mean number of visited states over ten random seeds. (c) Histogram of the mean number of visitations based on 1×1 square quantization: (left) PPO Only, (middle) RND, and (right) the proposed multi-model fusion learning method with $K = 3$. (d) Learning curve of α during the fusion learning. (e) Learning curve of the extrinsic value function $V_\xi(s)$.

no reward due to the pure exploration setting. The agent's goal is to explore as many states as possible with the aid of intrinsic reward. Since the state space is continuous, we count how many 1×1 unit blocks the agent visited among the total 100×100 blocks.

Fig. 4(b) shows the mean number of visited blocks with respect to time averaged over ten random seeds for PPO without any intrinsic reward (denoted by PPO Only), RND with a single prediction network ($K = 1$), and the proposed multi-model fusion method based on RND. It is seen that the proposed multi-model method with fusion learning outperforms RND, as expected. As K increases, the performance also increases. We applied Welch's t-test on our data at the end of the training to statistically check the proposed multi-model method's gain over the baseline. Welch's t-test does not require the variances of the two tested methods to be the same, and it is robust compared with other tests for comparison of two different RL algorithms [58]. The proposed multi-model method with $K = 3$, $K = 10$ and $K = 20$ outperforms RND with 97% 99% and 99% confidence level, respectively. We observed that the multi-model method with model order $K = 3$ already achieves quite a good portion of performance improvement. This observation is consistent with a recent result [59] which states that the model order need not be too large in practice.

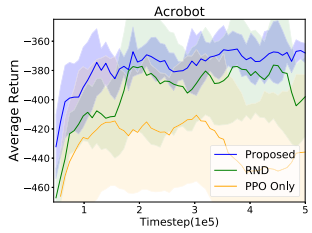
Fig. 4(c) shows the histogram of the number of visitations based on 1×1 square quantization. It is seen that the proposed multi-model method with $K = 3$ visited a much broader area in the upper-right room than the baselines through trajectories via both the upper-left and lower-right rooms. Fig. 4(d) shows the learning curve of α during



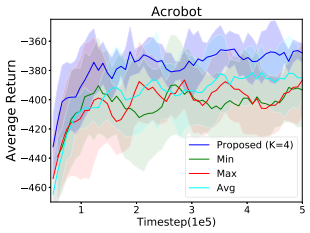
(a)

Method	Performance	p-value
Proposed	$\mathbf{-368.0 \pm 6.6}$	-
RND	-398.0 ± 27.5	0.047
PPO Only	-435.8 ± 52.9	0.031

(b)



(c)



(d)

Fig. 5. (a) OpenAI Gym Acrobot environment for exploration with extrinsic reward. There are two links of unit length and two joints. We set the target height as $h = 1.99$ to enforce sparsity of the non-trivial reward, (b) Average task performance over five random seeds and the p-value of the null hypothesis $H_0 : \mu_{\text{proposed}} \leq \mu_{\text{baseline}}$ based on Welch's t-test, where μ_{proposed} and μ_{baseline} are the performance means of the proposed method and each baseline, respectively (c) Performance comparison with baselines, and (d) Performance comparison with static fusion methods: minimum, maximum, and average taking.

the proposed fusion learning with $K = 3$. We observe that in the 4-room maze case, the value of α gradually increases so that our fusion rule selects relatively conservative fusion strategies as timestep goes. (Note that as α increases, the α-mean behaves more like minimum.) At around timestep $2.2 \cdot 10^5$, α stops to be learned. Recall that the 4-room maze does not give any extrinsic reward to the agent. Therefore, the extrinsic value function $V_\zeta(s)$ in the proposed algorithm should ideally be a zero function. Fig. 4(e) shows that the learned extrinsic value function converges to zero around timestep $2.2 \cdot 10^5$. Since the extrinsic value function is almost zero above this point, the estimated advantage value using the GAE [52] method becomes zero, $\nabla_\alpha J(\pi_{\tilde{\theta}})$ in (8) becomes zero, and learning of α stops.

Next, we considered the Acrobot environment [57], which is an exploration task with the extrinsic reward given to the agent. The agent consists of two links of unit length and two joints, as seen in Fig. 5(a). One joint is fixed at the center. The agent can move the other joint by applying an action $a \in \{-1, 0, 1\}$. The state consists of two angles θ_1, θ_2 and velocity information. The nonnegative reward 0 is given only when θ_1 and θ_2 satisfy $-\cos\theta_1 - \cos(\theta_1 + \theta_2) > h$ with $h < 2$, and reward -1 otherwise. We set $h = 1.99$ to enforce sparsity of the non-trivial reward. Figs. 5(b) and 5(c) show that the proposed multi-model fusion learning method with $K = 4$ noticeably improves the performance of the RND baseline. Note that the performance variance of the multi-model method with fusion learning with $K = 4$ is significantly reduced compared to RND.

Fig. 5(d) shows the comparison between the proposed multi-model method with adaptive fusion learning and the fixed fusion rules: minimum, maximum, and average taking in Acrobot. Again, it is clearly seen that the proposed method outperforms the fixed fusion methods. Even if we do not know *a priori* which fusion method is adequate in a specific environment, the proposed adaptive fusion performs well by learning α adaptively.

5.2.2. Continuous control tasks with delayed reward

To demonstrate the effectiveness of incorporating next-state distribution prediction (Algorithm 2) into our proposed method, we conducted experiments on six sparse-reward Mujoco tasks [57,60]: Walker2d, Hopper, InvertedPendulum, HalfCheetah, Ant, and Humanoid. To implement the sparse-reward settings of the considered environments, we adopted the delay method [23]. Initially, we collect extrinsic rewards produced by the relevant environments for every Δ timesteps or until the episode concludes. Subsequently, we provide the cumulative sum of rewards to the agent at the conclusion of each Δ timesteps or at the conclusion of an episode, and continue this cycle. We set $\Delta = 40$, as used in [35].

We compared the proposed multi-model method with fusion learning with existing state-of-the-art intrinsic reward generation methods. The baselines under consideration encompass the intrinsic reward module [35], RND (Random Network Distillation) [30], the single-model surprise [29], and a disagreement method utilizing the variance of multiple predicted next states [33]. We verified that three other baseline algorithms [21,22,28] underperform the other baselines, so the corresponding plots are omitted for readability. To ensure a fair comparison, we employed PPO with an identical neural network architecture and consistent hyperparameters across all approaches. Hence, any performance disparities observed are solely attributed to variations in intrinsic reward generation methods. For an overview of simulation hyperparameters and reproducibility verification, please refer to the supplementary material.

It is seen in Fig. 6 that the proposed fusion-based intrinsic reward generation method yields top-level performance. We found that setting among $K = 2$, 3 or 4 was enough for good performance, which conforms to [59] regarding the model order.

5.2.3. Practical sparse-reward tasks

We first conducted additional experiments on a sparse-reward robotic arm control task [61], which is different from the control tasks with delayed rewards in the previous Section 5.2.2. As shown in Fig. 7, the robot agent manipulates its arm and gripper, and at every timestep, the agent receives a 39-dimensional state. For every episode of length 500 timestep, the agent receives reward 1 if the center of mass of the gripper's two fingers reaches a 3-dimensional goal position (the orange sphere in Fig. 7) before the episode ends. Otherwise, the reward is zero. The proposed method uses the multi-model RND described in Algorithm 1.

Table 1 shows that the proposed method outperforms the other baselines. Compared with PPO Only (success rate 20%), the considered intrinsic reward methods help the agent's exploration. However, the success rate of the proposed method using the multi-model RND performs best with the success rate of 85%, which is higher than the other baselines such as Disagreement (66%). We also observe the gap between the proposed method and RND (48%), which shows the effectiveness of the proposed fusion method.

Finally, we performed experiments in gym-pybullet-drones [62], a challenging dynamic environment simulator for drone control. Drones become popular for public/industrial applications such as communication, searching areas, delivery, or cleaning [63–65]. Recently, researchers have given attention to devising stable control methods using drone control simulators.

Fig. 8(a) shows a rendered snapshot in the considered dynamic environment simulator. Each input observation is a vector of the drone's kinematic information: current position, orientation, velocity, and angular velocity. Action is a vector containing the RPM values of each motor. In the physics module of the simulator, forces and torques are calculated from action RPM values and are applied to the drone to fly to the target position. In real-world scenarios, degradation of control accuracy may occur due to the transmission error of the control signal. To reflect this situation, we revised the original simulator to inject perturbation noise to the received RPM values and give errors in the applied forces. This change makes the considered task more dynamic and the problem more challenging.

The goal of the drone is (i) to fly to the target position and (ii) to hover around it for a certain period. We considered the following sparse reward setting: if the drone receives the reward zero if it achieves the goal, or -1 otherwise. For example, if the drone manages to reach the

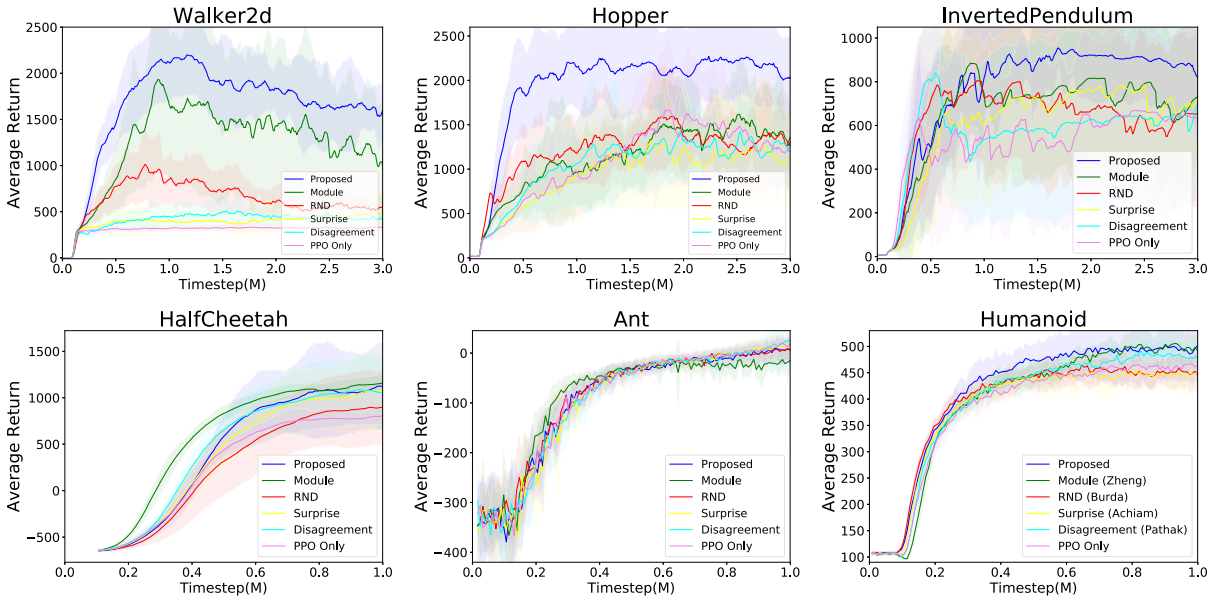


Fig. 6. Performance comparison on six delayed reward environments. All simulations were carried out using ten specific random seeds. On the *y*-axes, we illustrate the mean value of extrinsic returns obtained from the most recent one hundred episodes, averaged across the ten random seeds. To give sufficient time steps for each environment, for the three environments in the top row, the experiments were performed for 3M timesteps. For the environments in the bottom row, the experiments were conducted for 1M timesteps. (For clarity, the first author of each of the algorithms are shown in the Humanoid plot.).

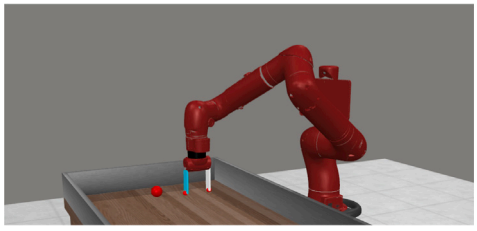


Fig. 7. Considered sparse-reward robot task from [61].

Table 1
Success rate comparison. We report the average success rate at the end of the training (1M timesteps), averaged across ten random seeds.

	Success rate(%)
Proposed	**85**
Module	50
RND	48
Surprise	59
Disagreement	66
PPO Only	20

target but deviates from the position earlier than the period, it receives a reward of −1. Every episode has a length of timestep 482. The proposed method uses the multi-model RND described in Algorithm 1.

Figs. 8(b) and 8(c) show that the proposed fusion-based method yields top-level performance. In Fig. 8(d), it is seen that setting the model order $K = 3$ produced the best performance, which aligns with the previous results in Section 5.2.1 and explanation in Section 5.2.2 regarding the model order. The performance exhibits an enhancement as the value of K increases. However, upon reaching the optimal model order, further improvements cease, and in some cases, there may be a slight degradation due to the increased complexity of model estimation, aligning with our intuition.

The formulation of the fusion function is $f_\alpha(x_1, \dots, x_K) = \left(\frac{1}{K}\sum_{i=1}^{K} x_i^{\frac{1-\alpha}{2}}\right)^{\frac{2}{1-\alpha}}$. When $\alpha = -\infty$, the fusion output $f_\alpha(x_1, \dots, x_K) = \max_i x_i$, while $f_\alpha(x_1, \dots, x_K) = \min_i x_i$ as $\alpha \to \infty$. As seen in Fig. 8(e), starting from the initial value $\alpha = 0$, the learning fusion parameter α dynamically varies during the training procedure, ending in an increase of α approaching the minimum of the input values.

We also calculated the training time in a sparse-reward drone control environment, comparing our algorithm ($K = 3$) with the RND method ($K = 1$). These computations were conducted on an Intel Core i9-10900X CPU @ 3.70 GHz. As shown in Table 2, the training time ratio is significantly lower than 3. This indicates that the increase in training time is not as substantial as the increase in the number of models.

6. Conclusion and future work

We have proposed a new adaptive fusion method with multiple prediction models for intrinsic reward generation for sparse-reward RL. By imposing inductive bias on the fusion rule, our adaptive fusion learning uses multiple models for better intrinsic reward generation. Our method also reduces the number of control parameters and improves the efficiency of fusion learning to maximize RL performance. Experimental results show that the proposed approach outperforms existing intrinsic reward generation methods in numerous sparse-reward environments.

We propose three promising future directions to extend the work presented in this paper. First, while we ensure that the learning of α is aligned with maximizing the cumulative extrinsic return and has demonstrated practical effectiveness, formal proof of the convergence of α remains an open problem. Proving this convergence is an important area for future research.

Second, reward-shaping with expert knowledge to accelerate exploration presents an interesting opportunity. In this work, we focused on intrinsic reward generation without relying on expert knowledge to create general-purpose algorithms, as expert knowledge or data is not always available. However, leveraging expert data to improve learning efficiency is a practical and relevant direction, especially since some studies have already explored using limited expert data to enhance exploration [66].

Third, extending the proposed fusion method to multi-agent RL offers another valuable direction. Many existing works in multi-agent RL adopt the concept of *centralized training with decentralized execution* (CTDE). Adapting our proposed fusion rule for CTDE with minimal learning parameters would be a fruitful extension.

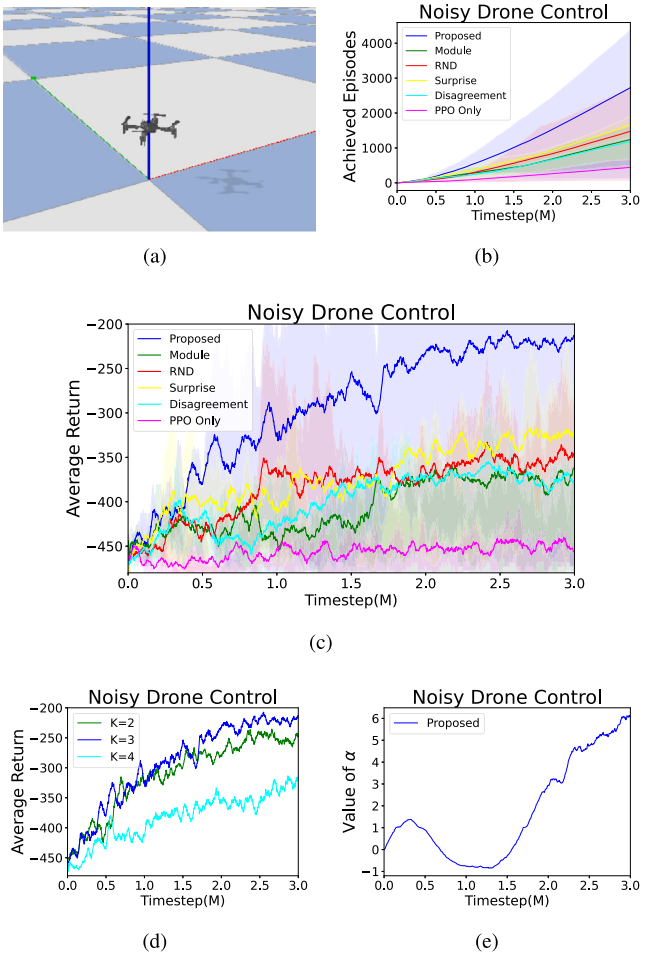


Fig. 8. (a) A rendered snapshot in the considered sparse-reward drone control task, a dynamic environment simulator for practical applications, (b) the performance comparison: the number of episodes during which the drone achieved the goal, (c) the performance comparison: average return. For (b) and (c), we report the 25%–75% percentile band for better clarity. (d) a parameter study on the effect of the model order K, and (e) a learning curve of fusion parameter α during the proposed fusion learning. The y-axes in (b), (c), and (d) represent the mean values of the return averaged over five random seeds.

Table 2
Average total training time per episode in seconds in a sparse-reward drone environment. 'Ratio' refers to the ratio of the training time with the proposed method to the training time with RND. Each episode consists of 482 timesteps.

Proposed ($K = 3$)	RND ($K = 1$)	Ratio
5.68	5.27	1.08

CRediT authorship contribution statement

Giseung Park: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Whiyoung Jung:** Validation, Software, Investigation, Formal analysis. **Seungyul Han:** Resources, Formal analysis. **Sungho Choi:** Validation, Investigation. **Youngchul Sung:** Writing – review & editing, Supervision, Resources, Funding acquisition, Conceptualization.

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.

Acknowledgments

This work was supported by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD), South Korea, under UD230017TD.

Appendix A. Necessity of the homogeneity condition (Condition 2)

Note the fusion function is some averaging function. Condition 2 is a proper condition for any reasonable averaging function [46]. Suppose that the homogeneity condition in Condition 2 for the function f is not satisfied and thus assume the following nonhomogeneous relationship:

$$f(cx_1, cx_2) = c' f(x_1, x_2), \tag{A.1}$$

where c' ($\neq c$) is mapped for each scaling factor c for the completeness of the scaling operation.

Now suppose that there exist two constants c, c' such that $c > 1$ and $0 < c' < 1$ and $f(cx_1, cx_2) = c' f(x_1, x_2)$ for two input x_1 and x_2. Then, the monotonicity is broken. The two inputs x_1 and x_2 are increased as cx_1 and cx_2 but the corresponding output is reduced as $c' f(x_1, x_2)$ as compared to the original output $f(x_1, x_2)$. This is not the situation that we want. Furthermore, by repeatedly applying Eq. (A.1), we have

$$\lim_{n\to\infty} [f(c^n x_1, c^n x_2) = (c')^n f(x_1, x_2)] \tag{A.2}$$

yielding

$$f(\infty, \infty) = 0 \cdot f(x_1, x_2) = 0 \tag{A.3}$$

assuming $x_1, x_2 > 0$. Therefore, we have a contradiction. In the case of $0 < c < 1$ and $c' > 1$, we have a similar contradiction.

Now assume that there exist $c, c' > 1$ and $c \neq c'$. Set two inputs as $x_1 = x_2 = x$. Then, we have

$$g(x) \triangleq f(x, x) \tag{A.4}$$

$$g(cx) = f(cx, cx) = c' f(x, x) = c' g(x) \tag{A.5}$$

By repeating the iteration, we have

$$g(c^n x) = (c')^n g(x). \tag{A.6}$$

In Eq. (A.6), the input to the function $g(x)$ is exponentially increasing as c^n and the output increases exponentially as $(c')^n$. Such function $g(x)$ is uniquely given by the form

$$g(x) \sim (c')^{\log_c x}, \tag{A.7}$$

where $\sim$ means the scaling equivalence. However, for a different pair $\tilde{c}$ and $\tilde{c}'$, we also require

$$g(x) \sim (\tilde{c}')^{\log_{\tilde{c}} x}. \tag{A.8}$$

The two functions in (A.7) and (A.8) are not the same for general pairs (c, c') and $(\tilde{c}, \tilde{c}')$. Note that $g(x)$ should be the same regardless of the selected pair (c, c') by its definition (A.4). This cannot be satisfied in general in the nonhomogeneous case. So, we have contradiction in this case. A similar situation happens for $0 < c, c' < 1$.

However, if we have $c = c'$ for all scaling factor c (i.e., we impose the homogeneity condition), then the two functions in (A.7) and (A.8) are consistent and we have no contradiction. In this case, we have

$$g(x) \sim (c)^{\log_c x} = (\tilde{c})^{\log_{\tilde{c}} x} = x. \tag{A.9}$$

This makes sense because this simply means that if the input values to the fusion function are all the same, then the output of the fusion is simply the input. (Note the definition of $g(x)$ in (A.4).) Therefore, we require the homogeneity condition so as to have a self-consistent fusion function.

Algorithm 2 Fusion Learning with Multiple Prediction Models for Intrinsic Reward Generation: Multi-model Distribution Prediction.

L : batch size for policy training, L' : batch size for model training.
L_{mini} : minibatch size for policy training, L'_{mini} : minibatch size for model training.
N : epoch size for policy training, N' : epoch size for model training.
MAX : the maximum index of batch period l, K : the number of prediction models.
Initialize the policy π_{θ_0}, the K transition probability models $P_{\phi_0^1}, \cdots, P_{\phi_0^K}$, the mixing coefficients $q_1, \cdots, q_K$, the fusion parameter α, and the extrinsic value function parameter ζ.
Generate trajectories with π_{θ_0} and store them to the initially empty replay buffer M.
for Batch period $l = 0, \cdots, MAX - 1$ **do**
  Train $P_{\phi_l^1}, \cdots, P_{\phi_l^K}$ by performing gradient updates for (11), and update $q_1, \cdots, q_K$ by performing iterations with (12). For each $P_{\phi_l^i} (1 \le i \le K)$, we draw a batch D'_i of size L' randomly and uniformly from M, and perform the updates with minibatches of size L'_{mini} drawn from D'_i for N' epochs.
  for Timestep $t = lL, lL+1, \cdots, lL+L-1$ **do**
    Collect (s_t, a_t, r_t, s_{t+1}) in the current batch D and add (s_t, a_t, s_{t+1}) to M.
  end for
  Acquire the intrinsic reward $r_{t,int}$ of the current batch D of size L.
  Train π_{θ_l} by using PPO with the total rewards (6) and minibatch size L_{mini} for N epochs.
  Train α and ζ by using the parameter learning method described in Section 4.2 with the total rewards (6) and minibatch size L_{mini} for N epochs.
  Empty the current batch D.
end for

Appendix B. Algorithm: Multi-model distribution prediction

Appendix C. Details of implementation

C.1. Details in PPO algorithm

The update of policy by using PPO is as follows. Let D be the batch of experiences for training the policy, i.e., $D = (s_t, a_t, r_t^{total}, s_{t+1}, \cdots, r_{t+L-2}^{total}, s_{t+L-1}, a_{t+L-1}, r_{t+L-1}^{total})$, where $a_t \sim \pi_{\theta_l}(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$, and r_t^{total} is the total reward described below. Here, π_{θ_l} is the parameterized policy at the batch period l corresponding to timestep $t, \ldots, t+L-1$ (the batch period index l is included in π_{θ_l} for clarity). Then, the policy π_{θ_l} is updated at every batch period l with D by following the standard PPO procedure based on the total reward (6).

To improve numerical stability, we further applied the normalization technique [29] to the output $f_\alpha(s_t, a_t, s_{t+1})$ of the α-fusion in the delayed reward environments. The final intrinsic reward given the current batch D is expressed as

$$r_{t,int}(s_t, a_t, s_{t+1}) = \frac{f_\alpha(s_t, a_t, s_{t+1})}{\max\left\{\frac{|\sum_{(s,a,s')\in D} f_\alpha(s,a,s')|}{|D|}, 1\right\}}, \tag{C.1}$$

where (C.1) describes the applied normalization.

C.2. 1-step technique in the prediction-of-distribution case

In the prediction-of-distribution case, the intrinsic reward at timestep t is set as

$$r^j_{t,int}(s_t, a_t, s_{t+1}) = \log \frac{P_K(s_{t+1}|s_t, a_t)}{P_{\phi^j}(s_{t+1}|s_t, a_t)} \tag{C.2}$$

where $P_K = \sum_{i=1}^K q_i P_{\phi^i}$. However, for computation of the intrinsic reward, we applied the 1-step technique in [29]. With the 1-step technique, the prediction models $P_{\phi^1}, \ldots, P_{\phi^K}$ constructing P_K in the numerator of the right-hand-side (RHS) of (C.2) are set as the current models, but the model ϕ^j in the denominator of the RHS of (C.2) is set as the 1-step past model before update.

Appendix D. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2025.129748.

Data availability

The source code link are provided in Section 4.3 in the paper.

References

[1] V. Djordjevic, H. Tao, X. Song, S. He, W. Gao, V. Stojanovic, Data-driven control of hydraulic servo actuator: An event-triggered adaptive dynamic programming approach, Math. Biosci. Eng. 20 (5) (2023) 8561–8582, http://dx.doi.org/10.3934/mbe.2023376.

[2] M. Yousefi Nejad Attari, A. Ala, M. Ahmadi, E. Jami, A highly effective optimization approach for managing reverse warehouse system capacity across diverse scenarios, Process. Integr. Optim. Sustain. 8 (2023) 16–31, http://dx.doi.org/10.1007/s41660-023-00388-x.

[3] M. Songhorabadi, M. Rahimi, A.M.M. Farid, M.H. Kashani, Fog computing approaches in smart cities: A state-of-the-art review, 2020, CoRR abs/2011.14732 arXiv:2011.14732.

[4] A. Sharifi, A.T. Beris, A.S. Javidi, M. Nouri, A.G. Lonbar, M. Ahmadi, Application of artificial intelligence in digital twin models for stormwater infrastructure systems in smart cities, Adv. Eng. Informatics 61 (2024) 102485, http://dx.doi.org/10.1016/J.AEI.2024.102485.

[5] M. Ahmadi, A.G. Lonbar, A. Sharifi, A.T. Beris, M. Nouri, A.S. Javidi, Application of segment anything model for civil infrastructure defect assessment, 2023, http://dx.doi.org/10.48550/ARXIV.2304.12600, CoRR abs/2304.12600 arXiv:2304.12600.

[6] C. Atkinson, B. McCane, L. Szymanski, A.V. Robins, Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting, Neurocomputing 428 (2021) 291–307, http://dx.doi.org/10.1016/J.NEUCOM.2020.11.050.

[7] T. Théate, A. Wehenkel, A. Bolland, G. Louppe, D. Ernst, Distributional reinforcement learning with unconstrained monotonic neural networks, Neurocomputing 534 (2023) 199–219, http://dx.doi.org/10.1016/J.NEUCOM.2023.02.049.

[8] R. Wang, Z. Zhuang, H. Tao, W. Paszke, V. Stojanovic, Q-learning based fault estimation and fault tolerant iterative learning control for MIMO systems, ISA Trans. 142 (2023) 123–135, http://dx.doi.org/10.1016/j.isatra.2023.07.043.

[9] M. Ahmadi, A. Taghavirashidizadeh, D. Javaheri, A. Masoumian, S.J. Ghoushchi, Y. Pourasad, DQRE-SCnet: A novel hybrid approach for selecting users in federated learning with deep-Q-reinforcement learning based on spectral clustering, J. King Saud Univ. Comput. Inf. Sci. 34 (9) (2022) 7445–7458, http://dx.doi.org/10.1016/J.JKSUCI.2021.08.019.

[10] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, 2000, pp. 1057–1063.

[11] J. Peters, S. Schaal, Natural actor-critic, Neurocomputing 71 (7–9) (2008) 1180–1190, http://dx.doi.org/10.1016/J.NEUCOM.2007.11.026.

[12] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control. Signals Syst. 2 (4) (1989) 303–314, http://dx.doi.org/10.1007/BF02551274.

[13] X. Song, N. Wu, S. Song, Y. Zhang, V. Stojanovic, Bipartite synchronization for cooperative-competitive neural networks with reaction-diffusion terms via dual event-triggered mechanism, Neurocomputing 550 (2023) 126498, http://dx.doi.org/10.1016/J.NEUCOM.2023.126498.

[14] L. Kraemer, B. Banerjee, Multi-agent reinforcement learning as a rehearsal for decentralized planning, Neurocomputing 190 (2016) 82–94, http://dx.doi.org/10.1016/J.NEUCOM.2016.01.031.

[15] R. Jiang, X. Zhang, Y. Liu, Y. Xu, X. Zhang, Y. Zhuang, Multi-agent cooperative strategy with explicit teammate modeling and targeted informative communication, Neurocomputing 586 (2024) 127638, http://dx.doi.org/10.1016/J.NEUCOM.2024.127638.
[16] T.G. Karimpanal, E. Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering, Neurocomputing 263 (2017) 39–47, http://dx.doi.org/10.1016/J.NEUCOM.2017.04.074.
[17] P. Vamplew, R. Dazeley, C. Foale, Softmax exploration strategies for multiobjective reinforcement learning, Neurocomputing 263 (2017) 74–86, http://dx.doi.org/10.1016/J.NEUCOM.2016.09.141.
[18] S. Parisi, M. Pirotta, J. Peters, Manifold-based multi-objective policy search with sample reuse, Neurocomputing 263 (2017) 3–14, http://dx.doi.org/10.1016/J.NEUCOM.2016.11.094.
[19] A. Lazaridis, I.P. Vlahavas, REIN-2: Giving birth to prepared reinforcement learning agents using reinforcement learning agents, Neurocomputing 497 (2022) 86–93, http://dx.doi.org/10.1016/J.NEUCOM.2022.05.004.
[20] M. Andrychowicz, et al., Hindsight experience replay, in: Advances in Neural Information Processing Systems, 2017, pp. 5048–5058.
[21] H. Tang, et al., # Exploration: A study of count-based exploration for deep reinforcement learning, in: Advances in Neural Information Processing Systems, 2017, pp. 2750–2759.
[22] I.M. de Abril, R. Kanai, Curiosity-driven reinforcement learning with homeostatic regulation, in: 2018 International Joint Conference on Neural Networks, Ijcnn, IEEE, 2018, pp. 1–6.
[23] J. Oh, Y. Guo, S. Singh, H. Lee, Self-imitation learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 3878–3887.
[24] H. Kim, J. Kim, Y. Jeong, S. Levine, H.O. Song, Emi: Exploration with mutual information, 2018, arXiv preprint arXiv:1810.01176.
[25] R. Houthooft, et al., VIME: Variational information maximizing exploration, in: Advances in Neural Information Processing Systems, 2016, pp. 1109–1117.
[26] R. Cui, C. Yang, Y. Li, S.K. Sharma, Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning, IEEE Trans. Syst. Man Cybern. Syst. 47 (6) (2017) 1019–1029, http://dx.doi.org/10.1109/TSMC.2016.2645699.
[27] H. Wu, et al., Depth control of model-free AUVs via reinforcement learning, IEEE Trans. Syst. Man, Cybern.: Syst. 49 (12) (2019) 2499–2510, http://dx.doi.org/10.1109/TSMC.2017.2785794.
[28] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: International Conference on Machine Learning, PMLR, 2017, pp. 2778–2787.
[29] J. Achiam, S. Sastry, Surprise-based intrinsic motivation for deep reinforcement learning, 2017, arXiv preprint arXiv:1703.01732.
[30] Y. Burda, H. Edwards, A. Storkey, O. Klimov, Exploration by random network distillation, 2018, arXiv preprint arXiv:1810.12894.
[31] K. Chua, et al., Deep reinforcement learning in a handful of trials using probabilistic dynamics models, in: Advances in Neural Information Processing Systems, 2018, pp. 4754–4765.
[32] T. Kurutach, et al., Model-ensemble trust-region policy optimization, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=SJJinbWRZ.
[33] D. Pathak, D. Gandhi, A. Gupta, Self-supervised exploration via disagreement, in: International Conference on Machine Learning, PMLR, 2019, pp. 5062–5071.
[34] D. Ha, A. Dai, Q.V. Le, Hypernetworks, 2016, arXiv preprint arXiv:1609.09106.
[35] Z. Zheng, et al., On learning intrinsic rewards for policy gradient methods, in: Advances in Neural Information Processing Systems, 2018, pp. 4644–4654.
[36] H. Liu, P. Abbeel, Behavior from the void: Unsupervised active pre-training, Adv. Neural Inf. Process. Syst. 34 (2021) 18459–18473.
[37] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, K. Lee, State entropy maximization with random encoders for efficient exploration, in: International Conference on Machine Learning, PMLR, 2021, pp. 9443–9454.
[38] M. Yuan, M. Pun, D. Wang, Rényi state entropy maximization for exploration acceleration in reinforcement learning, IEEE Trans. Artif. Intell. 4 (5) (2023) 1154–1164, http://dx.doi.org/10.1109/TAI.2022.3185180.
[39] Z. Hong, et al., Diversity-driven exploration strategy for deep reinforcement learning, in: Advances in Neural Information Processing Systems, 2018, pp. 10510–10521.
[40] S. Xie, J. Huang, L. Lei, C. Liu, Z. Ma, W. Zhang, L. Lin, NADPEx: an on-policy temporally consistent exploration method for deep reinforcement learning, 2018, arXiv preprint arXiv:1812.09028.
[41] L. Shani, Y. Efroni, S. Mannor, Exploration conscious reinforcement learning revisited, in: International Conference on Machine Learning, PMLR, 2019, pp. 5680–5689.
[42] C. Colas, O. Sigaud, P.-Y. Oudeyer, Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms, in: International Conference on Machine Learning, PMLR, 2018, pp. 1039–1048.
[43] X. Wang, T. Li, Y. Cheng, Proximal parameter distribution optimization, IEEE Trans. Syst. Man Cybern. Syst. 51 (6) (2021) 3771–3780, http://dx.doi.org/10.1109/TSMC.2019.2931946.
[44] P. Shyam, W. Jaśkowski, F. Gomez, Model-based active exploration, in: International Conference on Machine Learning, PMLR, 2019, pp. 5779–5788.
[45] A. Rényi, On measures of entropy and information, in: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics, Vol. 4, University of California Press, 1961, pp. 547–562.
[46] G.H. Hardy, J.E. Littlewood, G. Pólya, Inequalities, Cambridge University Press, 1952.
[47] S. Amari, Information Geometry and Its Applications, vol. 194, Springer, 2016.
[48] S. Amari, Integration of stochastic models by minimizing *alpha*-divergence, Neural Comput. 19 (10) (2007) 2780–2796, http://dx.doi.org/10.1162/NECO.2007.19.10.2780.
[49] Z. Xu, et al., Meta-gradient reinforcement learning, in: Advances in Neural Information Processing Systems, 2018, pp. 2396–2407.
[50] T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H.P. van Hasselt, D. Silver, S. Singh, A self-tuning actor-critic algorithm, Adv. Neural Inf. Process. Syst. 33 (2020) 20913–20924.
[51] J. Schulman, et al., Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[52] J. Schulman, et al., High-dimensional continuous control using generalized advantage estimation, 2015, arXiv preprint arXiv:1506.02438.
[53] C.K.I. Williams, C.E. Rasmussen, Gaussian processes for machine learning, MIT Press, 2006.
[54] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[55] A.P. Dempster, et al., Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–22.
[56] A. Zhou, T. Knowles, C. Finn, Meta-learning symmetries by reparameterization, 2020, arXiv preprint arXiv:2007.02933.
[57] G. Brockman, et al., Openai gym, 2016, arXiv preprint arXiv:1606.01540.
[58] C. Colas, et al., A Hitchhiker's guide to statistical comparisons of reinforcement learning algorithms, 2019, arXiv preprint arXiv:1904.06979.
[59] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[60] E. Todorov, et al., Mujoco: A physics engine for model-based control, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, 2012, pp. 5026–5033.
[61] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, S. Levine, Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, in: Conference on Robot Learning, PMLR, 2020, pp. 1094–1100.
[62] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, A.P. Schoellig, Learning to fly—a gym environment with PyBullet physics for reinforcement learning of multi-agent quadcopter control, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2021, pp. 7512–7519, http://dx.doi.org/10.1109/IROS51168.2021.9635857.
[63] J. Fan, Y. Liu, G. Sun, H. Pan, A. Wang, S. Liang, Z. Liu, 3D position scheduling of UAV secure communications with multiple constraints, in: IEEE International Conference on Systems, Man, and Cybernetics, SMC 2022, Prague, Czech Republic, October 9-12, 2022, IEEE, 2022, pp. 143–149, http://dx.doi.org/10.1109/SMC53654.2022.9945385.
[64] A.M. Farid, M. Mouhoub, Evolutionary mapping with multiple unmanned aerial vehicles, in: IEEE International Conference on Systems, Man, and Cybernetics, SMC 2022, Prague, Czech Republic, October 9-12, 2022, IEEE, 2022, pp. 1443–1449, http://dx.doi.org/10.1109/SMC53654.2022.9945496.
[65] Y. Li, H. Cui, S. Zhang, Modeling and control algorithm design of a new curtain wall cleaning UAV, in: IEEE International Conference on Systems, Man, and Cybernetics, SMC 2022, Prague, Czech Republic, October 9-12, 2022, IEEE, 2022, pp. 755–760, http://dx.doi.org/10.1109/SMC53654.2022.9945518.
[66] S. Yue, J. Liu, X. Hua, J. Ren, S. Lin, J. Zhang, Y. Zhang, How to leverage diverse demonstrations in offline imitation learning, 2024, arXiv preprint arXiv:2405.17476.

Giseung Park received the B.S. degree in Electrical Engineering (minor in Mathematical Science) from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2016, and the M.S. and Ph.D. degree in Electrical Engineering from KAIST, Daejeon, South Korea in 2018 and 2024, respectively. He is currently a postdoctoral researcher at the Institute of Information Electronics in KAIST. His current research interests include multi-modal, multi-objective, and multi-agent reinforcement learning.

Whiyoung Jung received the B.S. degree in Mathematical Sciences from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2015, and the M.S. and Ph.D. degree in Electrical Engineering from KAIST, Daejeon, South Korea in 2017 and 2022, respectively. He is currently an AI Scientist at LG AI Research. His current research interests include reinforcement learning and robust and safe reinforcement learning.

Seungyul Han is an assistant professor in the Artificial Intelligence Graduate School (AIGS) and the Department of Electrical Engineering (EE) at the Ulsan National Institute of Science and Technology (UNIST). He received the B.S. (Double major in Mathematical Science) and M.S. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2013 and 2016, respectively. He also received his Ph.D. degree in Electrical Engineering from KAIST. Before joining UNIST, he was a postdoctoral researcher at the Institute of Information Electronics in KAIST, and he is now an assistant professor at UNIST. His main research interests include reinforcement learning, machine learning, deep learning, multi-agent systems, signal processing, and intelligent control systems.

Sungho Choi received the B.S. degree in the Department of Electrical and Electronic Engineering from Yonsei University, Seoul, South Korea in 2018 and the M.S. degree in School of Electrical Engineering from KAIST, Daejeon, South Korea in 2020. He is currently a Ph.D. student working with Professor Youngchul Sung in the School of Electrical Engineering, KAIST, Daejeon, South Korea. His current research interests include reinforcement learning and imitation learning.

Youngchul Sung (Senior Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1993 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2005. Before joining the Ph.D. program, he was with the LG Electronics Ltd., Seoul, from 1995 to 2000. From 2005 to 2007, he was a Senior Engineer with the Corporate Research and Development Center, Qualcomm Inc., San Diego, CA, USA, and participated in the design of Qualcomm's 3GPP R6 WCDMA base station modem. Since 2007, he has been on the Faculty of the School of Electrical Engineering, KAIST, Daejeon, South Korea. His research interests include reinforcement learning, signal processing for communications, statistical signal processing, and statistical inference and learning with applications to reinforcement learning, wireless communications, and related areas. He is currently the Vice Director of the IEEE Communications Society Asia–Pacific Board and an Executive Editor of ICT Express. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2017 to 2021 and the IEEE SIGNAL PROCESSING LETTERS from 2012 to 2014.