SUSD: Structured Unsupervised Skill Discovery through State Factorization

Seyed Mohammad Hadi Hosseini
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
hadi.hosseini17@sharif.edu

Mahdieh Soleymani Baghshah
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
soleymani@sharif.edu

Abstract

Unsupervised Skill Discovery (USD) aims to autonomously learn a diverse set of skills without relying on extrinsic rewards. One of the most common USD approaches is to maximize the mutual information (MI) between skill latent variables and states. However, MI-based methods tend to favor simple, static skills due to their invariance properties, limiting the discovery of dynamic, task-relevant behaviors. Distance-Maximizing Skill Discovery (DSD) promotes more dynamic skills by leveraging state-space distances, yet still fall short in encouraging comprehensive skills that engage all controllable factors or entities in the environment. In this work, we introduce SUSD, a novel framework that harnesses the compositional structure of environments by factorizing the state space into independent components (e.g., objects or controllable entities). SUSD allocates distinct skill variables to different factors, enabling more fine-grained control on the skill discovery process. A dynamic model also tracks learning across factors, adaptively steering the agent's focus toward underexplored dimensions as mastery develops elsewhere. This structured approach fosters the discovery of richer and more generalizable skills. Our experimental results across three environments, with factors ranging from 1 to 10, demonstrate that our method can discover diverse and complex skills without supervision, significantly outperforming existing unsupervised skill discovery methods in both factorized and unfactorized environments.

1 Introduction

Reinforcement Learning (RL) [Sutton et al., 1998] has made significant strides in solving complex tasks, such as strategic games [Schrittwieser et al., 2020, Badia et al., 2020, Mnih et al., 2013] and robotic manipulation [Gu et al., 2016, Andrychowicz et al., 2020], especially when supported by well-shaped, dense reward functions [Booth et al., 2023, Tang et al., 2025, Silver et al., 2018]. However, in many real-world settings, rewards are sparse and tasks are long-horizon, making learning much harder.

Humans often solve new problems by leveraging previously acquired skills rather than learning from scratch. This intuition can be transferred to RL by enabling agents to acquire a repertoire of reusable skills, reducing the need for task-specific training. This strategy, known as USD, has become widely used in recent RL research [Eysenbach et al., 2019, Liu and Abbeel, 2021, Campos et al., 2020, Achiam et al., 2018]. Precisely, in USD approach, an agent is placed in an environment without any predefined reward function and learns to acquire as many diverse and distinct skills as possible solely through interaction with the environment. Afterwards, these learned skills are leveraged by a high-level policy or zero-shot goal-reaching methods to solve a variety of downstream tasks.

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

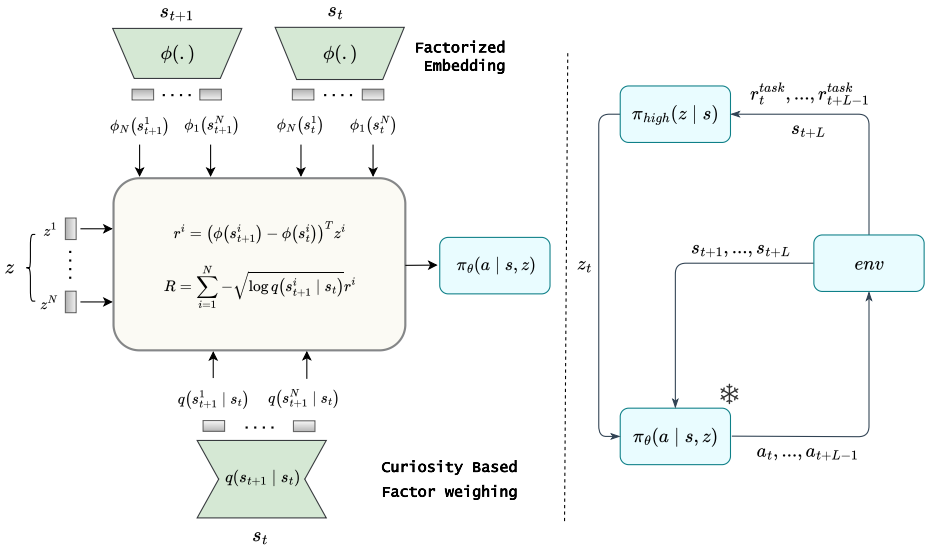


Figure 1: **Illustration of the SUSD Method.** The method consists of two stages: **Left:** In the skill learning stage, factorized embeddings are used, where factor i, $i \in \{1, ..., N\}$, of the current and next states is passed through its corresponding mapping function $\phi_i(.)$ to obtain a skill latent embedding. These embeddings, together with the skill factor inputs, are used to compute the intrinsic reward r^i of factor i. In the curiosity-based factor weighting module, a density model takes the full current state as input and estimates the probability of each next-state factor given the current state $-\log q(s^i_{t+1}|s_t)$. These probabilities are then used as weights to scale the factor-wise intrinsic rewards, which are summed to form the final intrinsic reward for training the skill policy. **Right:** In the task learning stage, the learned skill policy is frozen as a low-level policy, while a high-level policy π_{high} is trained to select a skill z every L steps by maximizing the task reward r^{task}.

Two main branches of USD have gained popularity. The first focuses on MI methods [Eysenbach et al., 2019, Sharma et al., 2029, Kamienny et al., 2022, Kim et al., 2023], that aim to maximize the mutual information between latent skills and states. The second branch centers on DSD methods [Park et al., 2024, 2023, 2022, Kim et al., 2024, Rho et al., 2025], that seek to maximize state changes along skill-specific directions. While MI-based methods often suffer from issues related to invariance to transformations, an aspect discussed in prior work [Park et al., 2024, 2023], DSD methods attempt to encourage more dynamic behaviors by maximizing state-space distances. However, the latent space of DSD methods mostly emphasize only easily controllable factors. Therefore, although state-of-the-art DSD methods perform well in unfactorized and simple environments such as Ant, HalfCheetah, and Walker [Todorov et al., 2012, Brockman et al., 2016, Towers et al., 2024], their performance degrades in more complex environments consisting several controllable elements. This bias arises because DSD lacks explicit mechanisms to ensure diversity across all controllable components of the environment. As a result, the learned skill latent spaces tend to underrepresent harder-to-control or less immediately responsive elements, limiting the overall expressiveness and coverage of the discovered skill set.

To address these limitations, we propose a factorized embedding approach that explicitly leverages the inherent factorization of the environment's state space, an inductive bias commonly present in structured domains such as multi-object environments or modular systems. In our framework, the state space is decomposed into factors, each corresponding to a distinct subset of controllable features. We associate a dedicated subset of skill variables with each factor, enabling the agent to develop localized skills within each subspace. This decomposed skill representation also naturally supports composability of skills through chaining which makes the learned behaviors more reusable for downstream tasks. Furthermore, in contrast to existing DSD methods that treat the state space holistically and fail to account for the varying difficulty of controlling different factors, we introduce a curiosity-driven factor weighting mechanism. This component dynamically adjusts the learning

emphasis across factors based on the agent's progress, encouraging exploration of underdeveloped or harder-to-control components as others are mastered.

The main contributions of our work are as follows: (1) We propose a novel method for skill learning in environments where the state space can be meaningfully decomposed into subspaces, leveraging this inductive bias to learn a repertoire of skills that collectively cover the entire state space. (2) We design an adaptive weighting mechanism that assigns greater emphasis to less explored factors, encouraging balanced skill acquisition. (3) We evaluate our approach across multiple environments and experiments, demonstrating that our method outperforms leading DSD methods as well as the MI-based approaches.

2 Related Works

2.1 Unsupervised Skill Discovery

USD methods aim to enable an agent to learn a diverse repertoire of skills without relying on predefined tasks. These learned skills can later be used to solve downstream tasks either in a zero-shot manner or as components of high-level policy strategies. Research in this area can be broadly categorized into two main directions: (1) Approaches that maximize the MI between skill latent variables and the states. (2) DSD methods, which introduce various distance metrics to explicitly encourage distinguishable behaviors across skills. Our method is built upon the DSD framework.

2.1.1 Mutual Information-Based Skill Discover

These methods aim to maximize the MI between states and latent skills, $I(S;Z)$, encouraging different skills to induce distinct and diverse behaviors. This objective is generally intractable, and previous studies approached it using two main strategies: (1) reverse mutual information (Reverse-MI), and (2) forward mutual information (Forward-MI). As regard to Reverse-MI [Mazzaglia et al., 2023, Kamienny et al., 2022, Wang et al., 2024, Kim et al., 2023, Hu et al., 2024, Eysenbach et al., 2019], optimization is in the form of $I(S;Z) = H(Z) - H(Z|S)$, where $H(Z)$ is a constant due to assuming a the skill prior distribution $p(z)$ is fixed. Thus, to maximize the $I(S;Z)$ we should minimize the $H(Z|S)$ we can modeled with $q_\theta(z|s)$, which is a varitional distribution of $p(z|s)$:

$$
\begin{aligned}
I(S;Z) &= -H(Z \mid S) + H(Z) && (1)\\
&= \mathbb{E}_{z,s}[\log p(z \mid s)] - \mathbb{E}_z[\log p(z)] && (2)\\
&\geq \mathbb{E}_{z,s}[\log q_\theta(z \mid s)] + \text{const.} && (3)
\end{aligned}
$$

Regarding Forward-MI [Liu and Abbeel, 2021, Laskin et al., 2022, Sharma et al., 2029], optimization can be expressed as $I(S;Z) = H(S) - H(S|Z)$, where $H(S)$ denotes the entropy of the states, encouraging trajectory diversity, and $H(S|Z)$ is the conditional entropy of states given the skill. To approximate $H(S|Z)$, a neural network is typically used to model a variational distribution. Additionally, maximizing $H(S)$ can be achieved through entropy estimation techniques. A key limitation of these methods is their invariance to scaling or any invertible transformation of the input variables. As a result, they tend to learn simple and static behaviors—such as opening the arm—while failing to capture more dynamic and complex skills [Park et al., 2022, 2023].

2.1.2 Distance-Maximizing Skill Discovery

Several recent works have adopted DSD to discover diverse and meaningful skills [Park et al., 2022, 2023, 2024, Rho et al., 2025, Kim et al., 2024]. DSD utilize the Wasserstein dependency measure as a learning objective for USD, defined as:

$$I_W(S;Z) = \mathcal{W}(p(s,z), p(s)p(z)) \tag{4}$$

where $\mathcal{W}(\cdot,\cdot)$ denotes the Wasserstein distance. In contrast, the MI can be defined as:

$$I(S;Z) = D_{\text{KL}}(p(s,z)||p(s)p(z)) \tag{5}$$

where D_{KL} is the Kullback–Leibler divergence, which is not sensitive to the underlying geometry of the state space. The Wasserstein-based objective, however, is distance-aware.

DSD methods can intuitively be simplified and expressed as aiming to maximize the state change along the direction specified by the skill z with the following objective [Park et al., 2024]:

$$\begin{aligned} &\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} (\phi(s_{t+1}) - \phi(s_t))^\top z \right] \\ &\text{s.t.} \quad \|\phi(s) - \phi(s')\|_L \leq d(s', s),\ \forall (s, s') \in \mathcal{S}_{\text{adj}} \end{aligned} \tag{6}$$

where, $\mathcal{S}_{\text{adj}}$ denotes the set of adjacent state pairs in the Markov Decision Process (MDP), and L is a norm ($L2$ norm is common in prior works).

The main difference between our work and others is that they typically consider relatively simple environments, such as Ant and HalfCheetah [Todorov et al., 2012, Brockman et al., 2016, Towers et al., 2024], where the state can be easily mapped to a low-dimensional latent space via $\phi(\cdot)$. Such simple environments with well-defined mapping from state space to latent space can allow them to easily cover the state space. In contrast, they struggle in more complex, object-centric environments such as the Multi-Particle environments [Lowe et al., 2017], where multiple objects and agents are present and even a simple action can simultaneously affect several factors.

2.2 State Space Factorization in RL

Many works leverage knowledge of state factorization for various purposes, including planning [Wang et al., 2022], data augmentation [Pitis et al., 2020], providing intrinsic rewards [Wang et al., 2023, Choi et al., 2024], and goal-conditioned learning [Chuck et al., 2025]. Recently, some methods [Hu et al., 2024, Wang et al., 2024, Chuck et al., 2023] have been proposed that consider interactions among objects in the environment and leverage this inductive bias with mutual information objectives to discover diverse skills. For example, DUSDi [Hu et al., 2024] learns disentangled skills where each skill component influences a specific factor, while SkiLD [Wang et al., 2024] induces different interaction graphs to capture object relationships. Although, utilizing the compositional structure of environments help to discover more diverse skills, the existing DSD methods have not yet exploited the state factorization and miss an opportunity to better structure the skill learning process and uncover more comprehensive and composable behaviors.

3 Preliminaries and problem setting

An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, r, p)$, where $s \in \mathcal{S}$ denotes a state in the state space, $a \in \mathcal{A}$ an action in the action space, $p(\cdot|s, a)$ the transition probability function, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ the reward function. In this work, we focus on a specialized class of MDPs designed for unsupervised skill discovery in reward-free Factored Markov Decision Processes (FMDPs) Kearns and Koller [1999], Osband and Van Roy [2014], Mohan et al. [2024]. There is a long line of prior work Wang et al. [2024], Hu et al. [2024], Wang et al. [2023], Eysenbach et al. [2019], Pitis et al. [2020], Choi et al. [2024] has adopted Factored Markov Decision Processes (FMDPs) as their environmental framework. In such settings, the reward function is unavailable, and the state space is structured as a Cartesian product of N subspaces: $\mathcal{S} := \mathcal{S}^1 \times \cdots \times \mathcal{S}^N$, where each $\mathcal{S}^i$ corresponds to a distinct factor of the environment. Each skill is represented by a latent vector $z \in \mathcal{Z}$, where the skill space is factored as $\mathcal{Z} := \mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N$, with each factor $\mathcal{Z}^i \in \mathbb{R}^D_i$. As a result, the overall skill space lies in $\mathbb{R}^{\sum_{i=1}^N D_i}$. A shared skill-conditioned policy $\pi(a|s, z)$ maps states and skill vectors to action distributions. While the skill space $\mathcal{Z}$ can be either discrete or continuous, we assume a continuous setting in this work. Nonetheless, our method is readily applicable to discrete skill spaces as well. To collect a skill trajectory, we sample a skill z from a predefined skill prior distribution $p(z)$ at the beginning of an episode. We then roll out the skill policy $\pi(a|s, z)$ with the sampled z for the entire episode. For the skill prior, we use a standard normal distribution.

In settings where the true state vector is available, the underlying factors can typically be derived directly. When only pixel-based observations are accessible, an encoder can be employed alongside representation learning techniques to extract disentangled factors from the visual input. However, in this work, we assume direct access to the underlying state vector that is available in many environments.

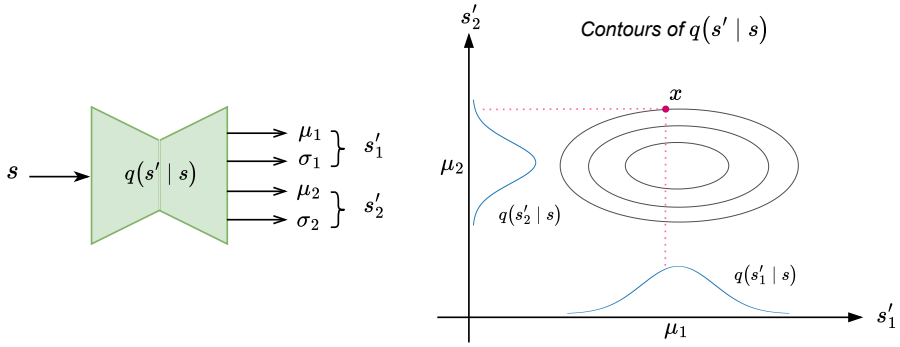


Figure 2: **Left**: The state s is passed to the density model, which estimates the mean and variance of $q(s'|s)$. These statistics are then partitioned by factors to obtain $q(s'_i|s_t)_{i=1}^2$. **Right**: Point x shows high probability in factor 1 but low probability in factor 2 which can't be utilized by the CSD method which assigns a weight to the state transition instead of state factor transitions.

4 Method

Existing DSD methods tend to focus on easily controllable state factors and lack mechanisms to encourage diversity across all controllable aspects. To address this, we propose a factorized embedding framework for DSD methods that decomposes the state space into distinct controllable factors, enabling localized and composable skill learning. Additionally, we introduce a curiosity-driven factor weighting mechanism that adaptively emphasizes harder-to-control factors during training.

In this section, we first describe how the observation space, skill latent space, and mapping function are factorized (Section 4.1). Then, we explain how the weight of each factor is computed, which reflects the extent of the agent's struggle with that factor and, consequently, its curiosity to acquire the corresponding skill (Section 4.2). Finally , we present the SUSD method in its entirety, detailing its intrinsic reward, the loss functions used for network updates, and the overall algorithm (Section 4.3).

4.1 Factorized DSD

We factorize the observation space into N factors $\{s^i\}_{i=1}^N$, each corresponding to a meaningful component of the environment. In essence, we leverage the compositional structure of the environment as an inductive bias to guide this partitioning of the observation space. More precisely, different controllable elements in the environment (e.g., the target, agent, and ammo in the 2D-Gunner [Lowe et al., 2017]) can be considered as different state factors. Moreover, each skill vector z is composed of N factors, each of dimension D. Specifically, for every state factor s^i, we associate a latent skill factor z^i within the skill vector. This design enables the agent to learn localized skills specialized to each subspace, improving both disentanglement and control. Furthermore, the function ϕ, which maps the state space to the skill latent space, is structured as N separate networks where each network i takes the s^i as input and outputs $\phi_i(s^i)$. Therefore, the original optimization problem defined in Eq. (6) is converted to the following factorized version:

$$\sup_{\pi, \{\phi_i\}_{i=1}^N} \mathbb{E}_{p(\tau,z)} \sum_{i=1}^{N} \sum_{t=0}^{T-1} (\phi_i(s_{t+1}^i) - \phi_i(s_t^i))^\top z^i$$
$$\text{subject to} \quad \sum_{i=1}^{N} \|\phi_i(s'^i) - \phi_i(s^i)\|_2 \leq 1, \quad \forall (s, s') \in \mathcal{S}_{\text{adj}} \tag{7}$$

4.2 Curiosity-based Factor Weighting

Inspired by Park et al. [2023] which prioritizes state transitions that are difficult to achieve under the current skill policy, we aim to encourage the discovery of hard-to-learn behaviors. To this end, we train a density model $q_\theta(s'|s) = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$ on (s, s') tuples collected by the skill policy and a negative log-likelihood of a transition from the current skill policy, $-\log q_\theta(s_{t+1}|s_t)$, is used as a controllability-aware distance function. It assigns high values for rare transitions while assigns small values for frequently visited transitions.

As discussed in Section 4.1, we factorize the state space into N distinct factors. This raises the question of which factors the agent should attend to at each timestep and how much importance each factor should receive in developing its skill set. In this section, we begin with a lemma that allows us to reformulate Eq. (6) so that the distance term is incorporated directly into the objective, rather than appearing in a constraint. Using this lemma, we then derive our final objective function.

Lemma 4.1. *In the DSD optimization problem (Eq. 6), we can include the distance term as a coefficient alongside the intrinsic reward. More generally, the equation presented in the related work (Eq. 6) can be rewritten as follows:*

$$\begin{aligned} \sup_{\pi,\phi} \ & \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} d(s', s)(\phi(s_{t+1}) - \phi(s_t))^\top z \right] \\ \textit{s.t.} \quad & \|\phi(s) - \phi(s')\|_L \leq 1, \ \forall (s, s') \in \mathcal{S}_{adj} \end{aligned} \tag{8}$$

The proof of this lemma is provided in Appendix A and is adapted from [Kim et al., 2024].

To move beyond the coarse-grained weighting used in [Park et al., 2023], which assigns a single curiosity score per transition, we propose a fine-grained factor-wise weighting mechanism that dynamically adjusts the contribution of each state factor during training. Specifically, to compute the factor-wise curiosity signals $-\log q_\theta(s^i_{t+1} \mid s_t)$, we feed s_t into the network to obtain $\mu_\theta(s_t)$ and diagonal $\Sigma_\theta(s_t)$ of a multivariate Gaussian distribution. Since the marginals of a Gaussian distribution are themselves Gaussian, we can easily extract the marginal mean and variance for each state factor by partitioning $\mu_\theta(s_t)$ and $\Sigma_\theta(s_t)$ according to the predefined factorization of the observation space. The resulting factor-wise parameters are used to calculate the curiosity weight for each factor as:

$$-\log q_\theta(s^i_{t+1} \mid s_t) \propto (s^i_{t+1} - \mu^i_\theta(s_t))^\top \Sigma^i_\theta(s_t)^{-1} (s^i_{t+1} - \mu^i_\theta(s_t)) \tag{9}$$

The square root of $-\log q_\theta(s^i_{t+1} \mid s_t)$ can be interpreted as a valid distance metric and thus incorporated into the objective defined in Eq. (8) according to Lemma 4.1. The curiosity-based factor weighting module is shown on the bottom of Figure 1. Furhermore, Figure 2 shows how this mechanism provides curiosity-based factor weighting. In this figure, the point x may have a high probability mass under the first factor while it has a low probability mas under the second one. This indicates that more weight should be assigned to the second factor transition than the first one while coarse-grained view of states, as used in CSD Park et al. [2023], overlook such factor-level controllability.

Accordingly, we our final optimization problem as follows:

$$\begin{aligned} \sup_{\pi, \{\phi_i\}_{i=1}^N} \ & \mathbb{E}_{p(\tau,z)} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \sqrt{-\log q_\theta(s^i_{t+1}|s_t)} (\phi_i(s^i_{t+1}) - \phi_i(s^i_t))^\top z^i \\ \text{subject to} \quad & \sum_{i=1}^{N} \|\phi_i(s'^i) - \phi_i(s^i)\|_2 \leq 1, \quad \forall (s, s') \in \mathcal{S}_{\text{adj}} \end{aligned} \tag{10}$$

4.3 SUSD Training

We optimize our objective using dual gradient descent. That is, with a Lagrange multiplier $\lambda \geq 0$, we use the following dual objectives to train SUSD:

$$r_i^{\text{SUSD}} := (\phi_i(s_{t+1}^i) - \phi_i(s_t^i))^\top z^i, \tag{11}$$

$$J^{\text{SUSD},\phi_i} := \mathbb{E}\left[r_i^{\text{SUSD}} + \lambda \cdot \min\left(\varepsilon, 1 - \|\phi(s_{t+1}^i) - \phi(s_t^i)\|\right)\right], \tag{12}$$

$$J^{\text{SUSD},\lambda} := -\lambda \cdot \mathbb{E}\left[\min\left(\varepsilon, 1 - \|\phi(s_{t+1}^i) - \phi(s_t^i)\|\right)\right], \tag{13}$$

$$R := \sum_{i=1}^{N} \sqrt{-\log q_\theta(s_{t+1}^i | s_t)}\, r_i^{\text{SUSD}} \tag{14}$$

where R is the intrinsic reward for the skill policy, and J^{SUSD,ϕ_i} and $J^{\text{SUSD},\lambda}$ are the objectives for ϕ_i and λ, respectively. We update for each factors independently. The variables s_{t+1} and s_t are sampled from a state pair distribution $p(s_{t+1}, s_t)$ that imposes the constraint in Eq. (10). The slack variable $\varepsilon > 0$ prevents the gradient of λ from always being nonnegative. Using these objectives, we train SUSD by optimizing the policy using Eq. (14) as the intrinsic reward, while updating the other components using objectives in Eqs. (12) and (13).

Algorithm 1 SUSD: Structured Unsupervised Skill Discovery

Initialize skill policy $\pi(a \mid s, z)$, function $\left\{\phi_i(s^i)\right\}_{i=1}^N$ conditional density model $q_\theta(s'^i \mid s)$, Lagrange multiplier λ
for $i \leftarrow 1$ to #epochs **do**
  for $j \leftarrow 1$ to #episodes per epoch **do**
    Sample skill $z \sim p(z)$
    Sample trajectory τ with $\pi(a \mid s, z)$
  end for
  Fit conditional density model $q_\theta(s'^i \mid s)$ using current trajectory samples
  Update $\phi_i(s^i)$ with gradient ascent on J^{SUSD,ϕ_i} ▷ Eq. (12)
  Update λ with gradient ascent on $J^{\text{SUSD},\lambda}$ ▷ Eq. (13)
  Update $\pi(a \mid s, z)$ using SAC with intrinsic reward R ▷ Eq. (14)
end for

The skill policy $\pi(a \mid s, z)$ is trained with Soft Actor-Critic (SAC) [Haarnoja et al., 2018] according to the obtained reward in Eq. (14) as an intrinsic reward. We train the other components with stochastic gradient descent. We summarize the training procedure of SUSD in Algorithm 1. Implementation details are provided in Appendix E.

5 Experiments

In evaluating SUSD, we begin by describing the experimental setup, including details of the environments and baselines (Section 5.1) and then address three key questions: Q1: In factorized environments, do our discovered skills outperform other unsupervised reinforcement learning methods on downstream tasks? (Section 5.2) Q2: In unfactorized environments, does our method remain competitive with alternative baselines?

5.1 Experimental Setup

We compare our method against four state-of-the-art unsupervised skill discovery approaches including three DSD-based methods and one MI-based method. We evaluate our method in three environments: two factorized, selected to demonstrate the capability of our method, and one unfactorized, included to show that our approach remains competitive with other baselines in environments that consist of a single factor.

Baselines: We evaluate our approach against three DSD based methods, namely LSD [Park et al., 2022], CSD [Park et al., 2023], and METRA [Park et al., 2024], as well as DIAYN [Eysenbach

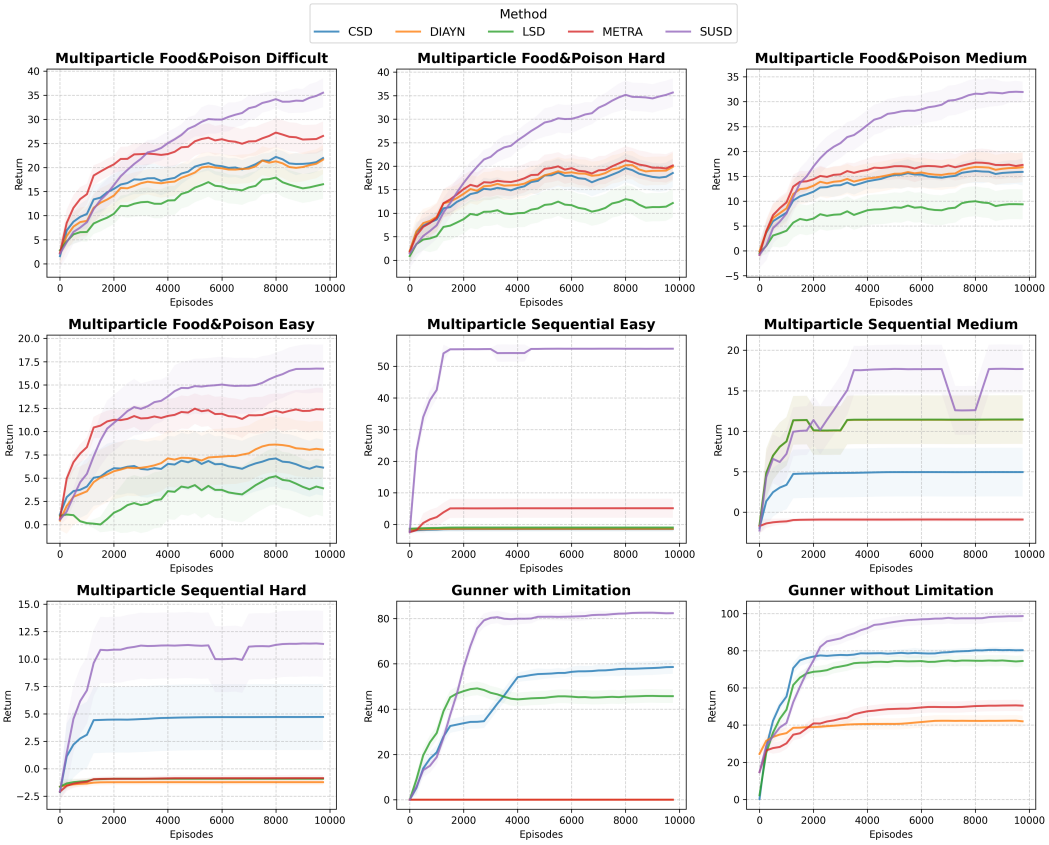


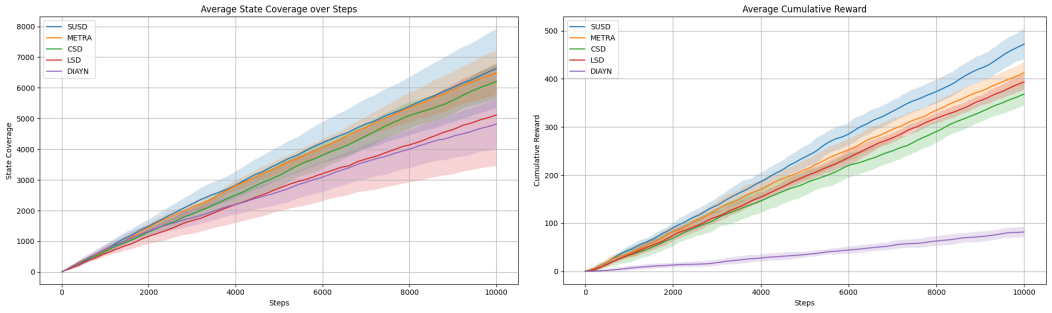
Figure 3: Training curves of SUSD and baseline methods on multiple downstream tasks in the Multi-Particle and 2D-Gunner environments. Each plot shows the mean and standard deviation of returns over 3 random seeds. SUSD consistently outperforms the baselines, including DSD and MI methods.

et al., 2019], which serves as a representative MI based method. LSD measures the distance between states using the ℓ_1 norm, defined as $d(s', s) = \|s' - s\|$. CSD defines distance in terms of transition probabilities, $d(s', s) = p(s'|s)$. METRA employs temporal distance, given by $d(s', s) = 1$ for the adjacent states in the trajectories. In contrast, DIAYN trains a discriminator to predict the corresponding latent variable z from a given state. For all baselines, we use their original hyperparameters and adopt the settings that yield the best performance for each method. Since this work focuses exclusively on continuous skills, we only consider the continuous versions of these methods. The dimensionality of the latent skill for all methods is set to $D = 2$.

Environments: We evaluate our method on the Ant environment [Todorov et al., 2012] to demonstrate its applicability to unfactorized settings. However, the majority of our evaluation focuses on environments tailored to SUSD, namely 2D-Gunner and Multi-Particle [Lowe et al., 2017]. The 2D-Gunner is a relatively simple domain, where a point agent can navigate inside a continuous 2D plane, collecting ammo and shooting at targets. Multi-Particle is a multi-agent domain modified based on [Lowe et al., 2017] and this modified version has been introduced in [Hu et al., 2024]. In this domain, a centralized controller simultaneously controls 10 heterogenous point-mass agents to interact with 10 stations, where each agent can only interact with a specific station. An overview of all environments is shown in Figure 5. Further details on the environments and downstream tasks are provided in Appendices B and C, respectively.

5.2 Evaluating Factorized Environments

In this section, we address the following question (Q1): In factorized environments, do the skills discovered by our method outperform other USD baselines? To answer this, we evaluate the



(a) State Coverage (b) Zero-shot Goal Reaching

Figure 4: State coverage and zero-shot goal reaching of policies learned by five skill discovery methods across 8 random seeds. SUSD shows competitive performance even in the unfactorized Ant environment.

performance of our method and the baselines on all downstream tasks in the MP and 2D-Gunner environments, using three random seeds. As shown in Figure 3, our method consistently outperforms all baselines by a good margin, demonstrating its effectiveness and its ability to leverage the structure of the environment. See Appendix D for the ablation study.

5.3 Unfactorized environments

In factorized environments, we can leverage the environment's structure, as shown in Section 5.2. To answer Q2, does our method remain competitive with other baselines in unfactorized environments?, we evaluate the performance of our method in the Ant environment [Todorov et al., 2012]. This environment is relatively simple, consisting of a single agent (the ant) that can move freely, without any explicit structure. We compare our method to other USD baselines using two metrics: (i) state coverage, measured as the number of unique (x, y) positions visited by the agent across eight runs, and (ii) Zero-shot goal-reaching, as described in METRA [Park et al., 2024], evaluates the agent's ability to achieve goals without additional training. In this task, the agent is allowed 10000 steps to accumulate as much reward as possible. To reduce the impact of randomness, we run the experiment across eight different seeds. The results are shown in Figure. 4.

6 Conclusion & Future works

Although excellent prior works in unsupervised skill discovery have successfully learned diverse behaviors without supervision, these methods often struggle in complex environments—settings with multiple objects or agents where actions can simultaneously affect more than one factor. To address this limitation, we introduced a DSD-based method designed to handle factorized environments by leveraging the environment's structure as an inductive bias to learn diverse skills. We propose a factorized embedding architecture and allocate a subset of skill variables to each controllable factor to avoid underrepresention of harder-to-control elements of the environment in the skill latent space. Moreover, we reformulate the DSD optimization problem and integrate the concept of curiosity-based factor weighting, which dynamically identifies the factors requiring more attention and adjusts the reward weights for each factor accordingly. We empirically demonstrate that SUSD enables agents to acquire diverse and dynamic skills in factorized environments.

As future work, our current method only handles continuous skill spaces, but it can be adapted to support discrete skill spaces as well. Additionally, our approach can be extended to work with pixel-based state environments. These represent two promising directions for further extending our work.

References

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

Serena Booth, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5920–5929, 2023.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International conference on machine learning*, pages 1317–1327. PMLR, 2020.

Jongwook Choi, Sungtae Lee, Xinyu Wang, Sungryull Sohn, and Honglak Lee. Unsupervised object interaction learning with counterfactual dynamics models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11570–11578, 2024.

Caleb Chuck, Kevin Black, Aditya Arjun, Yuke Zhu, and Scott Niekum. Granger-causal hierarchical skill discovery. *arXiv e-prints*, pages arXiv–2306, 2023.

Caleb Chuck, Fan Feng, Carl Qi, Chang Shi, Siddhant Agarwal, Amy Zhang, and Scott Niekum. Null counterfactual factor interactions for goal-conditioned reinforcement learning. In *International Conference on Learning Representations*, 2025.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Machine Learning*, 2019.

Shixiang Gu, Ethan Holly, Timothy P Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1610.00633*, 1(1), 2016.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. In *International conference on machine learning*, 2018.

Jiaheng Hu, Zizhao Wang, Peter Stone, and Roberto Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 76529–76552, 2024.

Pierre-Alexandre Kamienny, Jean Tarbouriech, Sylvain Lamprier, Alessandro Lazaric, and Ludovic Denoyer. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. In *International Conference on Machine Learning*, 2022.

Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *IJCAI*, volume 16, pages 740–747, 1999.

Hyunseung Kim, Byung Kun Lee, Hojoon Lee, Dongyoon Hwang, Sejik Park, Kyushik Min, and Jaegul Choo. Learning to discover skills through guidance. *Advances in Neural Information Processing Systems*, 36: 28226–28254, 2023.

Hyunseung Kim, BYUNG KUN LEE, Hojoon Lee, Dongyoon Hwang, Donghu Kim, and Jaegul Choo. Do's and don'ts: Learning desirable skills with instruction videos. *Advances in Neural Information Processing Systems*, 37:47741–47766, 2024.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, MA NYU, Aravind Rajeswaran, and Pieter Abbeel. Contrastive intrinsic control for unsupervised reinforcement learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 34478–34491, 2022.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Alexandre Lacoste, and Sai Rajeswar. Choreographer: Learning and adapting skills in imagination. In *International Conference on Learning Representations*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Aditya Mohan, Amy Zhang, and Marius Lindauer. Structure in deep reinforcement learning: A survey and open problems. *Journal of Artificial Intelligence Research*, 79:1167–1236, 2024.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. *Advances in Neural Information Processing Systems*, 27, 2014.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2022.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27225–27245. PMLR, 23–29 Jul 2023.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations*, 2024.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 33:3976–3990, 2020.

Seungeun Rho, Laura Smith, Tianyu Li, Sergey Levine, Xue Bin Peng, and Sehoon Ha. Language guided skill discovery. In *International Conference on Learning Representations*, 2025.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Machine Learning*, 2029.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28694–28698, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. In *Proceedings of the 40th International Conference on Machine Learning*, 2022.

Zizhao Wang, Jiaheng Hu, Peter Stone, and Roberto Martín-Martín. Elden: Exploration via local dependencies. *Advances in Neural Information Processing Systems*, 36:15456–15474, 2023.

Zizhao Wang, Jiaheng Hu, Caleb Chuck, Stephen Chen, Roberto Martín-Martín, Amy Zhang, Scott Niekum, and Peter Stone. Skild: Unsupervised skill discovery guided by factor interactions. *Advances in Neural Information Processing Systems*, 37:77748–77776, 2024.

A Proof of Lemma 4.1

We first start with Eq. (6):

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} (\phi(s_{t+1}) - \phi(s_t))^\top z \right] \quad \text{s.t.} \quad \|\phi(s) - \phi(s')\|_2 \leq d(s,s'),\ \forall (s,s') \in S_{\text{adj}}. \tag{15}$$

Let the scaled state function be defined as $\tilde{\phi}(s) := \frac{\phi(s)}{d(s,s')}$. Then, we can transform the constraint term in Eq. (15) as follows (since $d(s,s') \geq 0$):

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} (\phi(s_{t+1}) - \phi(s_t))^\top z \right] \quad \text{s.t.} \quad \|\tilde{\phi}(s) - \tilde{\phi}(s')\|_2 \leq 1,\ \forall (s,s') \in S_{\text{adj}}. \tag{16}$$

By replacing $\phi(s)$ with $\tilde{\phi}(s) \cdot d(s,s')$ in Eq. (16), we obtain:

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} d(s_t, s_{t+1}) \big(\tilde{\phi}(s_{t+1}) - \tilde{\phi}(s_t)\big)^\top z \right] \quad \text{s.t.} \quad \|\tilde{\phi}(s) - \tilde{\phi}(s')\|_2 \leq 1,\ \forall (s,s') \in S_{\text{adj}}. \tag{17}$$

B Environment Details

B.0.1 Ant

As shown in Figure 5(a), the Ant environment has an episode length of 200 steps. The observation space consists of a single factor representing the state of the Ant and is 29-dimensional. The action space is continuous, corresponding to the control of the Ant's joints, and has 8 dimensions.

B.0.2 Multi-Particle

As shown in Figure 5(b), agents are represented by small circles, while stations are represented by large circles. Agents can only interact with stations of the same color. The Multi-Particle environment has a 70-dimensional observation space, composed of 10 state factors that capture the states of each agent and its corresponding landmark. The action space is 50-dimensional, with 5 dimensions per agent controlling their movements and interactions with the landmarks.

B.0.3 Gunner

As shown in Figure 5(c), the blue star marks the position of the agent, the blue line marks its shooting direction, the red diamond marks ammo location, and the orange cross marks the target position. The agent has a 18-dimensional observation space, consisting of 3 state factors: Agent Position, Ammo State, Target State. The action is 6-dimensional, 2 for agent movement, 3 for ammo pickup, and 1 for shooting direction.

C Downstream Tasks

C.1 Ant

Multi-goal Ant: The task requires the agent to reach four target goals, each within a radius of 3. Each goal is randomly selected from the region $[s_x - 7.5, , s_x + 7.5] \times [s_y - 7.5, , s_y + 7.5]$, where (s_x, s_y) denotes the agent's current position in the x-y plane. The agent is awarded 2.5 upon reaching a goal. A new goal is generated either when the current goal is reached or if the agent fails to reach it within 50 steps.

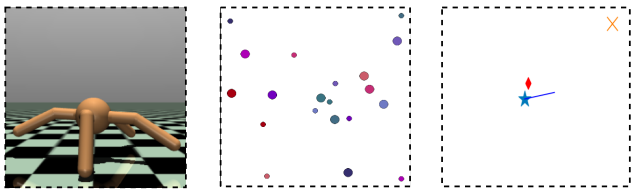


Figure 5: **Benchmark Environments**

C.2 Gunner

• Unlimited Ammo (unlim): In this downstream task, targets appear at random locations, and the agent must approach and shoot each target to score. Since ammunition is unlimited, the agent does not need to collect any.

• Limited Ammo (lim): This downstream task differs from the "Unlimited Ammo" task in that the agent begins without ammunition and must collect it before shooting, while all other aspects remain unchanged.

C.3 Multi-Particle (MP)

• Sequential interaction (seq) (easy, medium, hard): In this task, agents are required to interact with their assigned stations following the sequence given by an instruction at the start of each episode. Interacting with stations out of sequence incurs a penalty. In the easy version, the sequence has length 2; in the medium version, length 3; and in the hard version, length 4.

• Food-poison (fp) (easy, medium, hard, difficult): In this downstream task, each station delivers either food or poison to its corresponding agent. Agents must decide whether to interact with their station based on a sequence of binary indicators provided at the beginning of each episode. The easy version uses a sequence of length 2, the medium version length 5, the hard version length 8, and the difficult version length 10.

D Ablation Study

In this study, we isolate and evaluate the impact of the curiosity-based weighting module by removing it. This allows us to measure the performance drop when only factorization is used. As shown in Figure. 6, our method without the weighting module still outperforms CSD Park et al. [2023], but its performance declines compared to our full method that incorporates both factorization and curiosity-based weighting.

E Implementation Details

Unsupervised skill discovery methods. For the Ant environment, which has only one factor, we use a 2-D continuous skill vector. For the Multi-Particle environment, which consists of $N = 10$ factors, we use a 20-D continuous skill vector. Finally, for 2D-Gunner, with $N = 3$ factors, we use a 6-D continuous skill latent vector. Continuous skills are sampled from the standard Gaussian distribution. All baselines use a 2-D continuous skill vector, as this configuration yields their best performance. We present the full list of hyperparameters used for skill discovery methods in Table 1.

High-level controllers for downstream tasks. In Figure 1, we evaluate the learned skills on downstream tasks by training a high-level controller $\pi^h(\mathbf{z} \mid \mathbf{s}, \mathbf{s}_{\text{info}})$, which selects a skill every $K = 5$ environment steps for both MP and 2D-Gunner. At each selection step, the high-level policy chooses a skill $\mathbf{z}$, and the pre-trained low-level skill policy $\pi^l(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$ executes this skill for the next K steps. High-level controllers are trained using SAC [Haarnoja et al., 2018] for continuous skills, with hyperparameters identical to those used in the unsupervised skill discovery methods (Table 2).

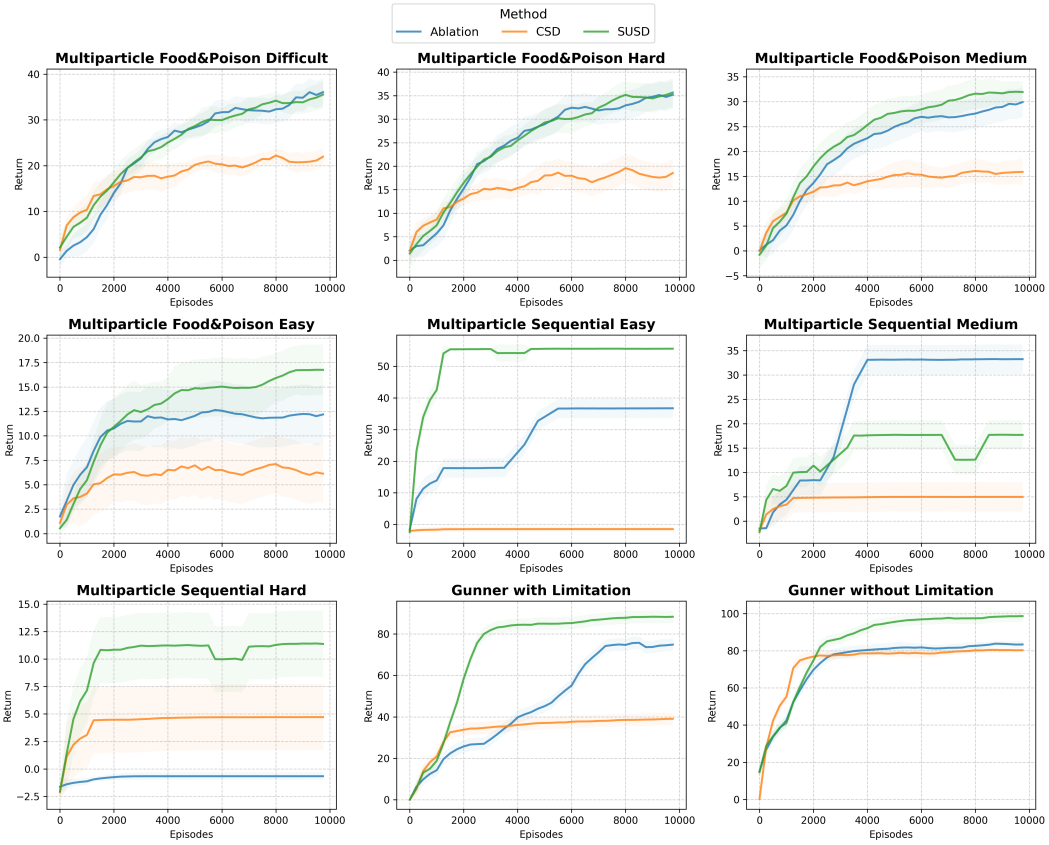


Figure 6: Effect of the curiosity-based weighting module. Removing it still yields better results than CSD, but performs worse than our full method.

Zero-shot goal-conditioned RL. In Figure 4(b),We evaluate the zero-shot performance of our method against other DSD baselines on goal-conditioned downstream tasks in the Ant environment. The skill vector $\mathbf{z}$ is recomputed at every step. For DIAYN, $\mathbf{z}$ is chosen based on the output of the skill discriminator at the goal state, i.e., $q(\mathbf{z} \mid \mathbf{g})$, where q denotes DIAYN's skill discriminator.

Table 1: Hyperparameters for unsupervised skill discovery methods.

Hyperparameter	**Value**
Learning rate	0.0001
Optimizer	Adam
# episodes per epoch	8
# gradient steps per epoch	50 (MP, Gunner, Ant)
Minibatch size	256
Discount factor γ	0.99
Replay buffer size	10^6
# hidden layers	2
# hidden units per layer	1024
Target network smoothing coefficient	0.995
Entropy coefficient	0.1 (Adaptive)
SUSD ϵ	10^{-6}
SUSD initial λ for each factor	3000
# Number of factors N	10 (MP), 3 (Gunner), 1 (Ant)
# Dimensions of each factor in $\mathbf{Z}$	2

Table 2: Hyperparameters for downstream policies.

Hyperparameter	**Value**
# training epochs	10^4 (MP, Gunner, Ant)
# episodes per epoch	1
# gradient steps per epoch	50 (MP, Gunner, Ant)
# skill sample frequency R	10
Replay buffer size	10^6
Target network smoothing coefficient	0.995
Entropy coefficient	0.1 (Adaptive)
Skill range	$[-1.5, 1.5]^{ND}$