Proceedings of the Twentieth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2024)

SCORE: Skill-Conditioned Online Reinforcement Learning

Sara Karimi[1,2], **Sahar Asadi**[2], **Amir H. Payberah** [1]

[1] KTH Royal Institute of Technology, Sweden
[2] King.com Ltd., Sweden
sara.karimi@king.com, sahar.asadi@king.com, payberah@kth.se

Abstract

Solving complex long-horizon tasks through Reinforcement Learning (RL) from scratch presents challenges related to efficient exploration. Two common approaches to reduce complexity and enhance exploration efficiency are (i) integrating learning-from-demonstration techniques with online RL, where the prior knowledge acquired from demonstrations is used to guide exploration, refine representations, or tailor reward functions, and (ii) using representation learning to facilitate state abstraction. In this study, we present **S**kill-**C**onditioned **O**nline **RE**inforcement Learning (SCORE), a novel approach that leverages these two strategies and utilizes skills acquired from an unstructured demonstrations dataset in a policy gradient RL algorithm. This integration enriches the algorithm with informative input representations, improving downstream task learning and exploration efficiency. We evaluate our method on long-horizon robotic and navigation tasks and game environments, demonstrating enhancements in online RL performance compared to the baselines. Furthermore, we show our approach's generalization capabilities and analyze its effectiveness through an ablation study.

Introduction

Reinforcement Learning (RL) systems have demonstrated remarkable success in tackling tasks across various domains, such as games, robotics, and autonomous driving (Souchleris, Sidiropoulos, and Papakostas 2023; Singh, Kumar, and Singh 2022; Elallid et al. 2022). However, solving complex tasks using RL faces several challenges, such as (i) the need to collect large amounts of samples from the environment by exploring the task space, which is costly or impractical in some domains, and (ii) the lack of expressive state representations, which is essential for learning an effective policy. These challenges are especially relevant to the gaming industry, where RL is applied for gameplay testing in games with complex state representations.

One approach to mitigate the first obstacle (i.e., collecting large amounts of samples) is to implement a guided exploration strategy for RL. Such strategies enhance sample efficiency, accelerate learning, and improve overall performance (Ladosz et al. 2022). Previous studies have employed unstructured datasets (i.e., consisting of sequences of states and actions excluding the reward corresponding to each pair) to acquire *skills* applicable in solving various downstream tasks (Ajay et al. 2021; Hausman et al. 2018; Lynch et al. 2020; Merel et al. 2018, 2020; Pertsch, Lee, and Lim 2021; Shankar et al. 2019; Sharma et al. 2019). These skills are temporally-extended actions representing a specific behavior in that environment. These approaches have used pretrained skill priors to guide an RL policy trained in the space of these skills. Moreover, they have demonstrated leveraging these prior experiences in RL can enhance robustness in generalization to new downstream tasks.

To tackle the second challenge (i.e., lack of expressive state representations), recent studies inspired by the human brain have learned compact representations of tasks, facilitating accelerated learning in subsequent experiences. This state representation learning process focuses on identifying the most important aspects of experiences (Radulescu, Shin, and Niv 2021). In the realm of RL applied to complex and large-scale tasks with high-dimensional state spaces, mastering effective representations enhances both the sample efficiency and computational effectiveness of the RL process (Stooke et al. 2021; Uehara, Zhang, and Sun 2022).

This work aims to tackle the above challenges within environments of varying complexity by offering robust representations for effectively guiding exploration. We present **S**kill-**C**onditioned **O**nline **RE**inforcement Learning (SCORE), a novel method that integrates skill priors extracted from unstructured data into policy gradient online RL, enriching it with informative representations. These skills provide an input representation that guides exploration and facilitates effective policy learning. Our approach provides a fresh perspective on the concept of *skill atoms* (Deterding 2015) in game design, which are the fundamental building blocks of player skills that can be combined to execute complex tasks within a game. Our work proposes a novel way to utilize these learned skills, offering a new direction for enhancing gameplay through skill-informed RL.

To implement SCORE, we develop a variational autoencoder model, inspired by OPAL (Ajay et al. 2021), that captures a continuous representation of skills and their underlying distribution from unstructured data. To evaluate our method, we conduct experiments on long-horizon robotic and navigation tasks and game environments, and our results show that policy learning conditioned on learned skills en-

ables more efficient exploration, sample efficiency, and enhanced performance [1].
In summary, our contributions are three-fold:

1. Extension of policy gradient RL to incorporate learned skills as input representations for downstream tasks. We conduct an ablation study to analyze the impact of various components in our proposed solution, thus demonstrating its effectiveness.
2. Comparison of SCORE with Proximal Policy Optimization (PPO) (Schulman et al. 2017), a policy gradient method, and Soft Actor-Critic (SAC) (Haarnoja et al. 2018), an off-policy method, showcasing the superior performance of our method on environments of varying complexity.
3. Providing insights into the generalization capability of our approach, showcasing its adaptability to new environments.

Preliminary

In this section, we first provide details of the concepts essential for defining our problem. We then explain the construction of continuous skill embedding and skill prior model required for our method, SCORE.

Reinforcement Learning

To address the problem at hand, we employ a Markov Decision Process (MDP) framework, characterized by the 5-tuple $\{S, A, P, R, \gamma\}$, where S represents the set of possible states within the environment, A denotes the action space, P is the transition probabilities that govern the dynamics of moving from one state to another following an action, R is the reward function that assigns immediate rewards for actions taken in states, and γ represents the discount factor, influencing the significance of future rewards. The objective is to discover a policy $\pi_\theta(a|s)$, parameterized by θ, where $a \in A$ and $s \in S$, that maximizes the expected sum of discounted rewards, defined as $J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t]$, where T represents the finite horizon of an episode and $r \in R$. This formulation enables a principled approach to learning the optimal policies in complex environments.

In this work, we use the policy gradient algorithms (Sutton and Barto 2018), a family of RL algorithms that collect samples by interacting with the environment using the latest policy (known as *policy rollout* phase) and then directly optimize policies by ascending gradients of expected rewards (known as *policy update* phase). More specifically, we employ PPO (Schulman et al. 2017) as a policy gradient algorithm with the following objective function:

$$J(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\mathcal{A}_t(s,a), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon).\right.\right. \left.\left.\mathcal{A}_t(s,a)\right)\right] \quad (1)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio of the policy after gradient updates to the policy before gradient updates, $\mathcal{A}_t(s,a)$ is the advantage function at time t estimating how good an action a is compared to all possible actions in state s, $\text{clip}(x,y,z)$ is a function that clips the value x to the range $[y,z]$, and ϵ is a hyperparameter that controls the clipping threshold.

[1]Our implementations are publicly available on Github: `https://github.com/sarakarimi/SCORE`

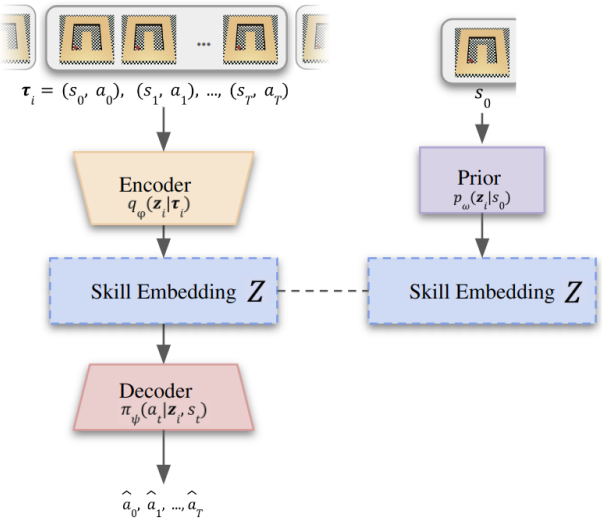

Figure 1: Overview of the autoencoder model designed for the simultaneous learning of skill embeddings and skill priors.

Skill Embedding With Learnable Skill Prior

This section explains the method for learning *skills*, temporally-extended actions representing a specific behavior in the environment. To build a skill embedding model from offline data, we need a dataset $D := \{\tau_i\}_{i=1}^N$ composed of N pre-recorded state-action trajectories $\tau_i = \{(s_0, a_0), \ldots, (s_T, a_T)\}$ of fixed length T. Such offline data can be collected from various sources, including other agents trained on diverse tasks, autonomous exploration by agents in real-world environments, and human teleoperation. The key aspect of this data collection approach is its reliance on unstructured data, which is sequences of state and action pairs without any reward information. This approach fosters adaptability in real-world settings. Notably, this method does not assume the presence of complete task solutions within the training data. Instead, it concentrates on extracting generalizable skills that improve the adaptability and efficiency of the learning process.

To train a model for extracting skills z from dataset D, we adopt the approach introduced in (Ajay et al. 2021), which employs a simple autoencoder with three learnable components: (i) skill encoder, (ii) skill decoder, and (iii) skill prior, as illustrated in Fig. 1. The skill encoder with parameters ϕ, denoted as $q_\phi(z|\tau)$, gets a trajectory τ from the dataset D as input and converts its sequence of states-actions into a posterior distribution on latent skill embeddings. Subsequently, the skill decoder with parameters ψ, denoted as $\pi_\psi(a_i|z, s_i)$, reconstructs the sequence of actions conditioned on the states and a sample z drawn from the skill

distribution. Concurrently, the skill prior with parameters ω, denoted as $p_\omega(z|s_0)$, learned alongside the autoencoder, establishes a prior distribution over the skills based on the initial state s_0 of the trajectory τ. So, when trained, the skill prior provides a skill based on the current state of the environment, and that skill is unrolled using the decoder for T steps before extracting another skill from the prior.

To train the skill prior to mimic the distribution learned by the skill encoder while only taking s_0 as input, the Kullback-Leibler (KL) divergence (Csiszar 1975) between the skill prior and skill encoder is calculated, and added as regularization to the autoencoding loss as below:

$$J(\psi, \phi, \omega) = \hat{\mathbb{E}}_{\tau \sim D, z \sim q_\phi(z|\tau)} \left[-\sum_{t=0}^{T} \log \pi_\psi(a_t|z, s_t) \right] - \beta D_{KL}(q_\phi(z|\tau)||p_\omega(z|s_0)) \quad (2)$$

where $D_{KL}(q||p)$ measures the KL divergence between distributions q and p, and β is a hyperparameter adjusting the weight of the KL regularization term.

Method

This section describes our method for combining the skill embedding model with RL. This integration aims to improve the policy gradient RL algorithm by providing richer input representations. This, in turn, helps make better use of available samples and learn more effective exploration policies.

In our approach, named **S**kill-**C**onditioned **O**nline **RE**inforcement Learning (SCORE), we integrate a pre-trained skill embedding model (described in the previous section) into a policy-gradient RL algorithm. More specifically, in the policy rollout phase of the policy-gradient algorithm, where the agent collects samples by interacting with the environment using the latest policy, we augment the state observations from the environment with the corresponding skill embedding obtained from the pre-trained skill prior. In other words, alongside the state observations from the environment, we provide the RL model with the representation of skills applicable in each state.

To successfully integrate these skills into online RL, we apply the following three steps: (i) skill conditioning and multi-step rollouts, (ii) using KL penalty between the decoder distribution π_ψ and policy distribution π_θ (Fig. 2) to provide stability in learning, (iii) stabilizing training through policy initialization and dynamic weighting of the KL regularization term. Below, we explain each of these steps.

Step 1 - Skill Conditioning: In the first step, as described in the previous section, the skill prior p_ω from the skill embedding model is trained to provide a skill embedding based on the first state s_0 in the trajectory sequence τ of size T. Consequently, when integrating the prior with policy gradient RL, we take a skill embedding sample from the skill prior model every T step. This means that the extracted skill embedding sample z_i is concatenated to the environment observations $(s_{i+t})_{i=0}^{T}$ for T consecutive steps before sampling the next skill embedding sample. This process is shown in Fig. 2 under policy rollout. Intuitively, the policy network in our RL agent learns to decode a skill into primitive actions given the state of the environment.

Step 2 - Using KL Penalty Between π_ψ And π_θ: Using a pre-trained fixed prior with the RL policy could lead to unstable learning due to policy distribution changes. Therefore, to mitigate this problem and maintain policy alignment with the data distribution, in the second step, we introduce a KL penalty into the policy loss in the following form:

$$\hat{J}_\theta = J_\theta - \alpha D_{KL}(\pi_\psi(a|z, s)||\pi_\theta(a|z, s) \quad (3)$$

where J_θ is the policy loss of the RL algorithm (in our case, policy loss of PPO shown in Equ. 1), and α is a variable coefficient used for adjusting the weight of the KL regularization term in the loss. This KL regularization step is depicted in Fig. 2 in the policy update phase (where the policy receives gradient updates based on the collected samples).

Step 3 - Policy Initialization And Dynamic Weighting: During the early training steps, the KL divergence between the initially random policy and the decoder's distribution might be too large, potentially leading to an unstable learning process. Moreover, optimizing the divergence between the decoder distribution π_ψ and policy distribution π_θ may not be effective in states that the decoder has not encountered during training. To address these issues, in the third step, we adopt two key strategies: (i) initializing the policy π_θ weights with those from the pre-trained decoder π_ψ and (ii) dynamically adjusting the regularization weights based on the confidence of the decoder in its predictions for new states. Specifically, we set the KL-regularization weight α (Equ. 3) based on the decoder's confidence. The decoder provides a probability distribution over possible actions given a state. We sample actions from this distribution and use the sample probabilities as a measure of the decoder's confidence, represented by α. This approach encourages the policy π_θ to closely align with the decoder's distribution π_ψ for states that are represented in the offline dataset. Our empirical findings demonstrate that these methods—skill conditioning, strategic initialization, targeted regularization, and uncertainty-adjusted regularization—are crucial to the success of our method in learning downstream tasks.

Evaluation

In this section, we provide the details of the experiments conducted on the selected environments, present the results, and analyze the findings. We have chosen two robotic and navigation tasks in maze-like environments, AntMaze and Maze2D, from the D4RL benchmark (Fu et al. 2021). Additionally, we test our proposed method on a platformer game environment, CoinRun, from the Procgen benchmark (Cobbe et al. 2020). We organize the findings into three main categories: (i) a comparison of the training performance of our approach with the baselines, (ii) an ablation study investigating the impact of specific components,

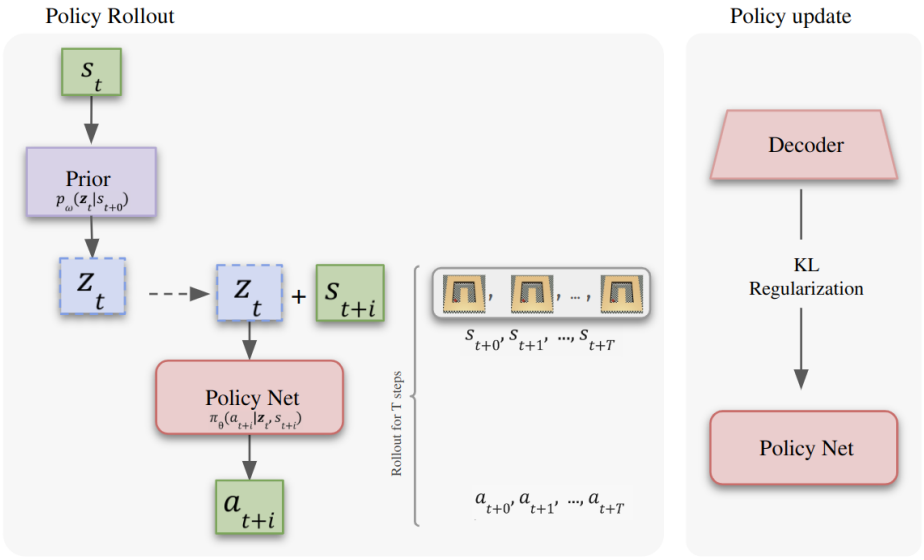

Figure 2: Overview of the training process of SCORE. In the policy rollout phase, the agent performs rollouts on the policy net for T steps using the same skill Z_i before extracting a new skill from the prior model. In the policy update phase, the policy net is updated and regularized by the KL divergence between the policy and decoder distribution. Both the prior and decoder are pre-trained and kept frozen in this process.

including the initialization of decoder weights, decoder regularization, and the application of uncertainty-based adjustments to the regularization term, and (iii) an evaluation of our models' capacity to adapt to a new, larger environment when trained on smaller environment settings.

Experiment Settings

Environments: In this study, we leverage three environments and their associated datasets. We select two environments from the D4RL benchmark (Fu et al. 2021), specifically the Maze2D and AntMaze environments, and CoinRun, a platformer game from the Procgen benchmark (Cobbe et al. 2020). In the Maze2D environment, the challenge involves navigating a ball toward a target location along the X and Y axes. The observation space consists of 4-dimensional arrays capturing the ball's coordinates (x, y) and velocities, and the action space is of size two, representing the linear force exerted on the ball in the X and Y directions. The AntMaze environment simulates an ant quadruped robot (from the OpenAI Gym MuJoCo benchmark (Todorov, Erez, and Tassa 2012)) tasked with maze navigation. Here, goal-oriented observations comprise 29 dimensions detailing the positional and velocity metrics of the ant's body segments, and the action space is of size eight, representing the torques applied at the hinge joints. The core aim is to tackle a navigation problem, guiding the agent from one maze corner to another.

In both D4RL environments, success is originally measured by a binary completion reward (0 and 1) upon reaching the goal. However, our experiments employ a dense reward strategy, computing rewards in AntMaze as the negative Euclidean distance of the agent's current location to the goal,

Figure 3: Example of an easy and a hard level in CoinRun (Cobbe et al. 2019).

and in Maze2D, the exponentiated value of this reward is used. Both environments provide two different maps of varying sizes, i.e., Medium and Large. On AntMaze, we have created one extra custom map called AntMaze XL, which expands the configuration of the Large map to four times its original size, offering a significantly more complex challenge to solve. (see Fig. 4 for Medium, Large, and XL maps).

The CoinRun environment (Cobbe et al. 2020) is a side-scrolling platformer game where an agent navigates through a platform filled with obstacles, such as gaps, enemies, and traps, to collect a coin at the far right of the platform. The state space is represented by Red-Green-Blue (RGB) images of size $(64 \times 64 \times 3)$ capturing the game screen, showing the agent's position, obstacles, enemies, and goal. The action space consists of discrete actions: move left, move right, and jump. CoinRun levels are procedurally generated, providing a near-infinite set of randomized levels with varying features such as platform layouts, game assets and backgrounds, and positions of in-game objects (Fig. 3). CoinRun levels can be generated with two difficulty levels:

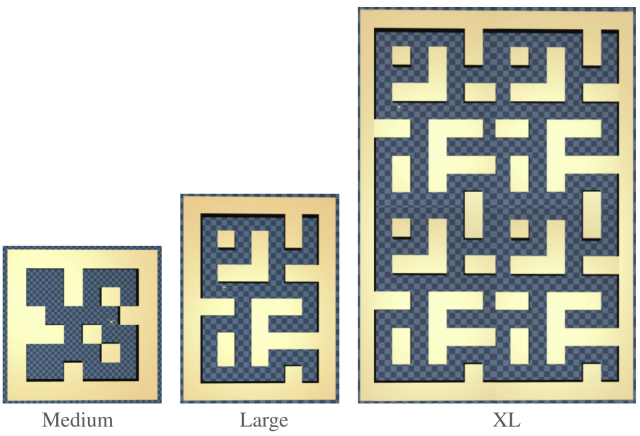

Figure 4: Medium, Large, and XL maze maps from AntMaze environment. Maze2D Medium and Large environments use the same maze maps.

easy and hard. In CoinRun, the agent receives a completion reward of 10 upon collecting the coin at the end of the level, and zero if the agent dies due to collision with obstacles or if it fails to collect the coin within the defined timestep, which is 1000 by default. In this work, we selected the first 64 difficult levels as the training levels, and to evaluate the trained models, we selected a distinct set of levels not being used during the training.

Datasets: We use diverse datasets from the D4RL environments to train the skill embedding model, including trajectories generated by navigating the point or ant to random goal locations in the maze. To collect trajectory datasets for the Maze2D environments, we use a waypoint planner that outputs a trajectory of points that the agent can follow to reach the goal. For this purpose, Q-Value Iteration (Watkins and Dayan 1992) is used. However, for the AntMaze environments, the ant agent utilizes a goal-reaching expert policy, trained using the SAC algorithm, which then follows a set of waypoints generated by the planner. For the CoinRun, we use a PPO model trained on 500 easy levels to collect trajectories. To simulate the mixed quality of data often encountered in real-world applications, we introduce random, suboptimal actions with a ratio of 0.2. This creates a dataset of trajectories with varying quality. Typically, when applied to real-world applications, the generated datasets in this work could be replaced with datasets collected from human teleoperations.

Baselines: As baselines, apart from vanilla PPO, we also compare to SAC (Haarnoja et al. 2018), which is an off-policy, actor-critic RL algorithm based in the maximum entropy framework, where the actor aims to maximize expected reward alongside entropy, aiming to perform well while maintaining as much randomness in actions as possible. We selected SAC as a baseline due to its reported superior performance over PPO in some continuous control tasks, as detailed in the original SAC paper (Haarnoja et al. 2018).

Metrics: We evaluate the following metrics in our experiments:

- *Episodic Return*: This is our primary metric for comparing the performance of different agents. It represents the sum of rewards accumulated over an episode. In the AntMaze environments, the reward is based on the negative Euclidean distance from the goal, meaning a shorter distance to the goal results in a higher (less negative) reward. For the Maze2D environments, the reward is exponentiated, with higher values indicating better performance. In the CoinRun environments, the reward is binary, with either +10 or 0 awarded, making +10 the highest possible episodic return.
- *Success Rate*: To complement the episodic return, we consider the success rate, which indicates the episodic frequency at which the agents reach their goals.
- *Episodic Length*: We also assess the episodic length, measuring the duration of episodes throughout the training as an indicator of exploration efficiency. All environments have a maximum episode length of 1000 steps. If the agent does not reach the goal within this limit, the episode terminates. A shorter episode length indicates that the agent has reached the goal more efficiently.

Implementations: In this work, we use PPO as the RL method in SCORE, such that the policy update and policy roll-out in Fig. 2 are performed by a modified PPO agent. However, any off-the-shelf continuous action RL method could be employed. So, in the rest of the paper, we will refer to our method as SCORE-PPO, indicating the use of PPO in SCORE. In our implementation of PPO for the D4RL environments, we adopt the details and hyperparameters outlined in (Huang et al. 2022a), and the SAC agent for the D4RL environments and PPO agent for the CoinRun environments adopt their implementation and hyperparameters from (Huang et al. 2022b).

The configuration and hyperparameters for the skill embedding and prior model remain consistent with those specified by (Ajay et al. 2021) with trajectory length set to 10 for D4RL and five for CoinRun, and skill length T set to eight. Regarding downstream tasks, models underwent training for 3.5 million steps in Medium-sized environments, 17.5 million steps in Larger-size environments on D4RL, and 7.5 million steps on CoinRun (details reported in Table 4 in Appendix). Despite the embedding model being trained on trajectories featuring a wide range of starting and goal positions, we maintain fixed start and goal locations for the agent during downstream task learning, ensuring a controlled learning environment. Similarly, in CoinRun, the embedding model trained on various easy levels is used to train a SCORE model on 64 difficult levels. All experiments are repeated with five seeds; the average and standard deviation values are reported in this paper.

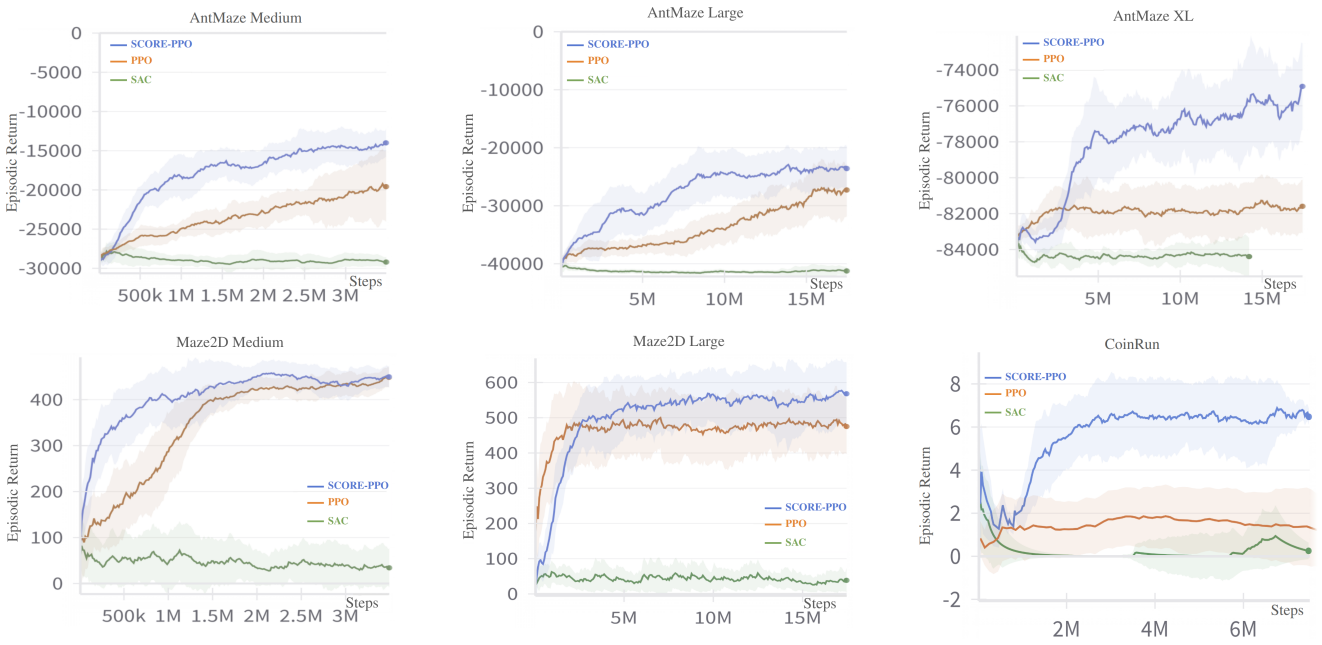

Figure 5: Average episodic return of SCORE-PPO, PPO, and SAC trained on AntMaze and Maze2D environments and CoinRun (the first 64 hard levels).

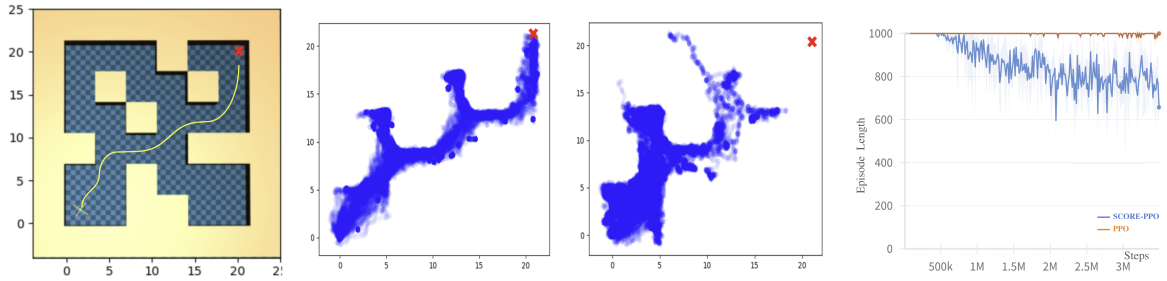

Figure 6: From left to right: (first) map of AntMaze Medium, visualization of state visitation for policies learned using (second) SCORE-PPO and (third) PPO, and (fourth) episodic length of the policy rollouts during training.

Environment	SCORE-PPO	SAC	PPO
AntMaze Medium	**81.5** $\pm$ **4.8**	0.0 $\pm$ 0.0	13.3 $\pm$ 5.2
AntMaze Large	**70.6** $\pm$ **3.5**	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
AntMaze XL	**14.0** $\pm$ **5.6**	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0

Table 1: Average success rate (%) of SCORE-PPO, PPO, and SAC on AntMaze Medium, Large, and XL.

Comparison with Baselines

In this section, we provide empirical evidence showing the enhanced performance of SCORE-PPO compared to vanilla PPO and SAC. The results illustrated in Fig. 5 show improved episodic return using SCORE-PPO, highlighting its strength in learning better exploration policies than the baselines. Also, looking at the state visitations of the agents on AntMaze at Fig. 6, SCORE-PPO demonstrates a more efficient policy than PPO. Furthermore, the plots of episodic length, as illustrated in Fig. 6, demonstrate SCORE-PPO's superior sample efficiency relative to PPO.

The analysis of performance improvements across various environments reveals that, although SCORE-PPO offers small gains over PPO in simpler settings such as Maze2D, its advantages become more pronounced in environments with larger state space, such as AntMaze and CoinRun. Additionally, the results of success rate (reported in Table 1) on the more complex D4RL tasks, AntMaze Large and XL show that while SAC and PPO policies fail to reach the goal during training, SCORE-PPO successfully learns a goal-reaching policy. Moreover, although SCORE-PPO is not significantly superior to PPO in terms of episodic return, the success rates indicate a significant distinction in its overall success in reaching the goal. Additionally, the training plots of Coin-

Environment	SCORE-M	SCORE-L	SCORE-XL	PPO-M	PPO-L
AntMaze large	**-29228 ± 522**	**-27668 ± 5794**	-	-34607 ± 4609	-
AntMaze XL	**-71193 ± 1634**	**-65072 ± 525**	-75573 ± 579	-	-80846 ± 295

Table 2: Average episodic return for generalization experiment. Evaluating the model trained on the Medium environment (SCORE-M and PPO-M), the Large environment (SCORE-L, PPO-L), and the XL environment (SCORE-XL) on the Large and XL environments. Models were evaluated for 1e5 steps and using five different seeds.

Environment	SCORE-PPO	PPO
64 unseen levels	5.32 ± 0.00	1.63 ± 0.03

Table 3: Average episodic return for generalization experiment. Evaluating the SCORE-PPO and PPO models trained on the 64 hard training levels on 64 hard new levels. Models were evaluated for 1e5 steps and using five different seeds.

Run reveal that while vanilla PPO struggles to make any significant progress on hard levels, SCORE-PPO demonstrates superior performance on those levels. These results support our initial proposition that offering online RL concise and informative representations is essential for enhanced learning in environments with large or complex state spaces.

Generalization

We assess the generalization of our approach in navigating new environments by evaluating the performance of SCORE-PPO, which is trained in a Medium environment when exposed to a larger environment. We compare these results against the generalization performance of SCORE-PPO trained directly on larger environments and a PPO model trained on a Medium environment. In Table 2, SCORE-M refers to the SCORE-PPO model trained on the Medium, SCORE-L refers to the model trained in a Large environment, SCORE-XL refers to the SCORE trained in an XL environment, and PPO-M and PPO-L refer to the PPO model trained in a Medium and Large environments respectively.

Table 2 shows that SCORE-M demonstrates generalization capabilities comparable with SCORE-L when applied to the AntMaze Large environment. On the AntMaze XL task, models trained on smaller environments (SCORE-M and SCORE-L) outperform the model trained directly on the XL environment. These results suggest that SCORE models trained in smaller (simpler) environments retain efficacy in bigger (more complex) settings without retraining.

In the CoinRun environment, we assess the generalization capabilities of the SCORE-PPO and PPO trained on the first 64 difficult levels in CoinRun, and we evaluate their performance on 64 new difficult levels that were not part of the training set. Table 3 shows that not only does SCORE-PPO demonstrate superior performance on training levels, but it can also retain the same performance level on unseen levels. This robustness in generalization is attributed to the skill representations' universality, underscoring the model's ability to adapt to varying environmental complexity levels and unseen scenarios.

Ablation Study

We conducted an ablation experiment on different SCORE-PPO components, showing their importance in the complete model. For this purpose, we performed experiments on the AntMaze Medium environment using the SCORE-PPO models with and without (i) weight initialization θ from the decoder weights ψ, (ii) decoder regularization (as shown in Fig. 2 policy update phase), and (iii) uncertainty-based weights α for the regularization term (as explained in the Method section). Through this ablation study, we aim to test the following hypotheses:

- Weight Initialization: During early training, the KL divergence between the random policy and decoder's distribution is large, causing instability. We hypothesize that initializing the policy weights with those of the decoder enhances initial stability.
- Decoder Regularization: Using a frozen prior model alongside the RL policy can cause instability due to changes in policy distribution. We hypothesize that regularizing the learned policy with the frozen decoder helps mitigate this distribution shift.
- Uncertainty-based Weights: Optimizing the divergence between the decoder and policy distributions may be less effective for unseen states. We hypothesize that dynamically adjusting the KL term will improve results by closely aligning the policy with the decoder's distribution for states present in the offline dataset while reducing alignment for new states.

The results presented in Fig. 7 show that ablating any of the three components causes a drop in the average episodic return collected during training. These results indicate that all three components are crucial to the final model's success. The most significant decline in performance occurs when the model is initialized with random weights rather than with weights derived from the decoder. This underscores the critical role of weight initialization in maintaining stability in KL divergence at the onset of training.

Related Work

In this paper, we aim to address long-horizon tasks with complex state representations by integrating concepts from skill priors in RL and skill-conditioned RL, thereby enriching the input representations provided to the model. Below, we review some of the related work.

State Abstraction in RL

State abstraction is particularly useful in addressing complex observation spaces. The primary goal is to identify a latent

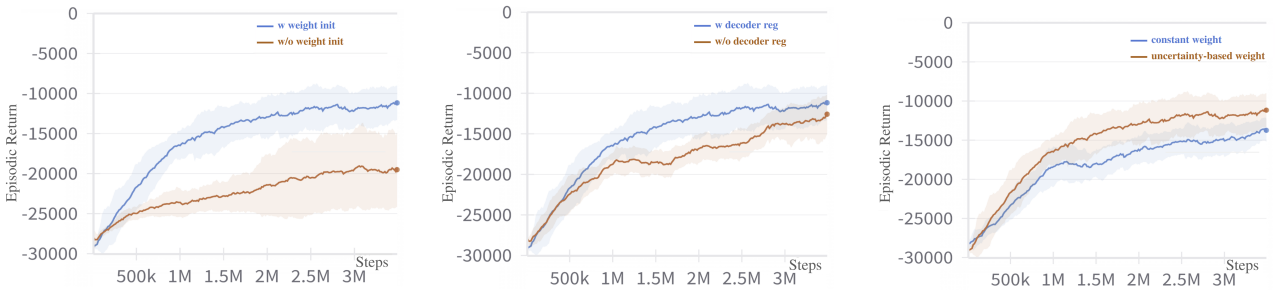

Figure 7: Ablation experiments showing the effect of different components of SCORE-PPO in AntMaze Medium environment. From left to right: (i) weight initialization ablation, (ii) decoder regularization ablation, and (iii) uncertainty-based KL weights vs constant weight.

representation of the state space that simplifies learning an optimal policy. Mawhorter and Smith (Mawhorter and Smith 2023) present a method for improving automated game testing by using a heuristic to make a simple abstraction of the game states that can help guide the exploration. One limitation of this approach is the need for game-specific formulation. In another work, Du et al. (Du et al. 2019) introduce a method for efficient exploration in RL by learning a decoding function that maps rich observations to latent states using regression and clustering techniques. One limitation of this approach is that the effectiveness of the method relies on certain assumptions on the state space, which can limit the application.

Temporally Extended Actions in RL

The formulation and application of temporal abstraction in RL have been essential for solving complex reinforcement learning challenges that span long durations. Sutton et al. (Sutton, Precup, and Singh 1999) put forward the idea of ”options”, which are temporally extended actions simultaneously trained using intrinsic reward functions. After these options are learned, a higher-level agent, which operates within a space of these options, evaluates them based on its own reward function. Following this research, Barreto et al. (Barreto et al. 2019) introduce a method that combines learned sequences of actions, known as skills, to create diverse behaviors. This method involves learning options associated with a set of pseudo-rewards and generating new options through any linear combination of these pseudo-rewards without requiring additional learning.

Skill Priors in RL

There are several works involving the use of unsupervised objectives to discover skills that are subsequently applied to various downstream tasks through both offline and online RL methods. Notably, Ajay et al. (Ajay et al. 2021) and Pertsch et al. (Pertsch, Lee, and Lim 2021) focus on training an RL agent within a skill-defined space, utilizing a fixed, previously learned prior for refining the RL objective. This approach facilitates more informed exploration within the RL paradigm. Similarly, in (Hakhamaneshi et al. 2022), Hakhamaneshi et al. enhance this methodology by integrating an inverse dynamic model, enabling the conditioning of priors on both the current and future states, further improving the performance.

To more accurately tailor extracted skills to human intentions, Wang et al. (Wang et al. 2022) present Skill Preferences (SkiP) modeling human preferences for skill extraction from offline data and employing human feedback as a reward signal for downstream task solving with RL. Nam et al. (Nam et al. 2022) propose a method that combines meta-learning with offline datasets to improve sample efficiency in tackling unseen tasks compared to previous meta-RL approaches. The approach involves extracting skills, mapping transitions to task embeddings via a task encoder, and then using meta-learning to condition a skills-based policy on these task embeddings, improving performance compared to traditional meta-learning approaches. Park et al. (Park et al. 2024) present HIQL, an algorithm for offline goal-conditioned RL that leverages hierarchical policies to learn from diverse, unlabeled datasets. HIQL efficiently utilizes offline data to train a high-level policy for generating subgoals and a low-level policy for executing these subgoals toward achieving complex navigation tasks.

Skill-Conditioned RL

Unlike human intelligence, which combines diverse skills to adapt to new challenges, artificial intelligence often focuses on mastering single tasks. By conditioning agent policies on specific skills, they can explore the state space more efficiently. Grillotti et al. (Grillotti et al. 2023) propose SCOPA, a skill-conditioned RL approach that efficiently utilizes successor features for learning a spectrum of expressive skills. SCOPA blends policy skill improvement strategies with universal function approximators, forming a unified algorithm that strives for performance maximization and ensures the execution of a broad array of predefined skills. In another work, Emukpere et al. (Emukpere et al. 2024) tackle skill discovery in robotic manipulation by employing multiple critics and corresponding reward functions. This multi-critic approach cultivates a skill-conditioned policy that not only masters diverse and safe manipulation skills but also facilitates their application to downstream tasks through hierar-

chical RL and planning frameworks. Laskin et al. (Laskin et al. 2022) introduces an unsupervised RL approach focusing on developing skill-conditioned policies that can apply learned skills to new, diverse tasks. This method encourages the discovery of a wide range of behaviors embedded within skill-conditioned policies, enhancing the agent's ability to navigate and solve complex environments without direct supervision.

While the listed papers demonstrate that skill use and composition in RL can enhance exploration and solve complex tasks, our work specifically targets the challenges posed by learning in environments with complex and large state spaces.

Conclusions and Future Work

This paper presented **S**kill-**C**onditioned **O**nline **RE**inforcement Learning (SCORE), an approach for integrating skill-conditioned policy learning with online Reinforcement Learning (RL). By leveraging unsupervised skill discovery from offline datasets, SCORE efficiently guides exploration and enhances policy learning for long-horizon navigation tasks and game environments. The performance improvements of SCORE over baseline methods (i.e., PPO and SAC) highlight the potential of using skill representations to facilitate adaptive and efficient learning in complex environments. These results highlight SCORE's superior performance compared to the conventional RL methods and its ability to generalize across diverse scenarios. Moreover, through an ablation study, we validated the efficacy of SCORE by analyzing the impact of its individual components in detail.

Our contributions, which include extending policy gradient RL to incorporate learned skills as input representations and demonstrating the generalization capabilities of our approach, lay the groundwork for future explorations into combining offline skill representations with online task learning. Further research may focus on showcasing the strength of skill-conditioned policies for online downstream task learning in more complex game environments where the state representations are large, sparse, and complex. Another direction is employing more sophisticated embedding models, such as transformers, to capture an even richer set of skills and enhance the generalization performance of RL agents.

Appendix

Hyperparameters Details

Here, we provide the details of the hyperparameters used in our implementations. For the D4RL environments, the skill embedding model utilizes three linear layers, each with a size of 200, for the decoder and prior. The encoder consists of 3 linear layers of size 200, followed by four layers of GRU units, each with 200 units. In the SAC model, both actor and critic networks have three linear layers of size 200. For the CoinRun environment, convolutional layers are used, and the network architecture used in SCORE, PPO, and SAC models is adopted from (Huang et al. 2022b).

Hyperparameter	**PPO**	**SAC**	**PPO**	**SAC**	**Skill**
learning rate	3e-4	3e-4	5e-4	3e-4	3e-4
batch size	64	256	256	64	50
horizon	2048	N.A	256	N.A	N.A
epochs per update	10	N.A	3	N.A	600
γ	0.99	0.99	0.999	0.99	N.A
gae-λ	0.95	N.A	0.95	N.A	N.A
value-loss coef.	0.5	N.A	0.5	N.A	N.A
buffer size	N.A	1e6	N.A	1e4	N.A
entropy coef.	0.0	0.2	0.01	0.2	N.A
target coef. τ	N.A	0.005	N.A	0.89	N.A
kL-term coef.	N.A	N.A	N.A	N.A	0.1

Table 4: Hyperparameters used in the D4RL and CoinRun implementations. The two columns provide the details of PPO and SAC for D4RL environments, and the second two columns provide the details of PPO and SAC for CoinRun environments. The final column contains the details of the skill embedding models.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Ajay, A.; Kumar, A.; Agrawal, P.; Levine, S.; and Nachum, O. 2021. OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning. arXiv:2010.13611.

Barreto, A.; Borsa, D.; Hou, S.; Comanici, G.; Aygun, E.; Hamel, P.; Toyama, D.; Mourad, S.; Silver, D.; Precup, D.; et al. 2019. The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Cobbe, K.; Hesse, C.; Hilton, J.; and Schulman, J. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. arXiv:1912.01588.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, 1282–1289. PMLR.

Csiszar, I. 1975. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, 146–158.

Deterding, S. 2015. The lens of intrinsic skill atoms: A method for gameful design. *Human–Computer Interaction*, 30(3-4): 294–335.

Du, S.; Krishnamurthy, A.; Jiang, N.; Agarwal, A.; Dudik, M.; and Langford, J. 2019. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, 1665–1674. PMLR.

Elallid, B. B.; Benamar, N.; Hafid, A. S.; Rachidi, T.; and Mrani, N. 2022. A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. *Journal of King Saud University-Computer and Information Sciences*, 34(9): 7366–7390.

Emukpere, D.; Wu, B.; Perez, J.; and Renders, J.-M. 2024. SLIM: Skill Learning with Multiple Critics. arXiv:2402.00823.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2021. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219.

Grillotti, L.; Faldor, M.; Leon, B. G.; and Cully, A. 2023. Skill-Conditioned Policy Optimization with Successor Features Representations. In *Second Agent Learning in Open-Endedness Workshop*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hakhamaneshi, K.; Zhao, R.; Zhan, A.; Abbeel, P.; and Laskin, M. 2022. Hierarchical Few-Shot Imitation with Skill Transition Models. arXiv:2107.08981.

Hausman, K.; Springenberg, J. T.; Wang, Z.; Heess, N.; and Riedmiller, M. 2018. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*.

Huang, S.; Dossa, R. F. J.; Raffin, A.; Kanervisto, A.; and Wang, W. 2022a. The 37 Implementation Details of Proximal Policy Optimization. In *ICLR Blog Track*. Https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Huang, S.; Dossa, R. F. J.; Ye, C.; Braga, J.; Chakraborty, D.; Mehta, K.; and Araújo, J. G. 2022b. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274): 1–18.

Ladosz, P.; Weng, L.; Kim, M.; and Oh, H. 2022. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85: 1–22.

Laskin, M.; Liu, H.; Peng, X. B.; Yarats, D.; NYU, M. A.; Rajeswaran, A.; and Abbeel, P. 2022. Contrastive intrinsic control for unsupervised reinforcement learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 34478–34491.

Lynch, C.; Khansari, M.; Xiao, T.; Kumar, V.; Tompson, J.; Levine, S.; and Sermanet, P. 2020. Learning latent plans from play. In *Conference on robot learning*, 1113–1132. PMLR.

Mawhorter, R.; and Smith, A. 2023. Automated Testing in Super Metroid with Abstraction-Guided Exploration. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–9.

Merel, J.; Hasenclever, L.; Galashov, A.; Ahuja, A.; Pham, V.; Wayne, G.; Teh, Y. W.; and Heess, N. 2018. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*.

Merel, J.; Tunyasuvunakool, S.; Ahuja, A.; Tassa, Y.; Hasenclever, L.; Pham, V.; Erez, T.; Wayne, G.; and Heess, N. 2020. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4): 39–1.

Nam, T.; Sun, S.-H.; Pertsch, K.; Hwang, S. J.; and Lim, J. J. 2022. Skill-based Meta-Reinforcement Learning. arXiv:2204.11828.

Park, S.; Ghosh, D.; Eysenbach, B.; and Levine, S. 2024. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36.

Pertsch, K.; Lee, Y.; and Lim, J. 2021. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, 188–204. PMLR.

Radulescu, A.; Shin, Y. S.; and Niv, Y. 2021. Human representation learning. *Annual Review of Neuroscience*, 44: 253–273.

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

Shankar, T.; Tulsiani, S.; Pinto, L.; and Gupta, A. 2019. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*.

Sharma et al., A. 2019. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*.

Singh, B.; Kumar, R.; and Singh, V. P. 2022. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2): 945–990.

Souchleris, K.; Sidiropoulos, G. K.; and Papakostas, G. A. 2023. Reinforcement learning in game industry—Review, prospects and challenges. *Applied Sciences*, 13(4): 2443.

Stooke, A.; Lee, K.; Abbeel, P.; and Laskin, M. 2021. Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, 9870–9879. PMLR.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Uehara, M.; Zhang, X.; and Sun, W. 2022. Representation Learning for Online and Offline RL in Low-rank MDPs. arXiv:2110.04652.

Wang, X.; Lee, K.; Hakhamaneshi, K.; Abbeel, P.; and Laskin, M. 2022. Skill preferences: Learning to extract and execute robotic skills from human feedback. In *Conference on Robot Learning*, 1259–1268. PMLR.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8: 279–292.