IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Skill-Critic: Refining Learned Skills for Hierarchical Reinforcement Learning

Ce Hao[1*†], Catherine Weaver[1*], Chen Tang[1], Kenta Kawamoto[2],
Masayoshi Tomizuka[1], Wei Zhan[1]

***Abstract*— Hierarchical reinforcement learning (RL) can accelerate long-horizon decision-making by temporally abstracting a policy into multiple levels. Promising results in sparse reward environments have been seen with *skills*, i.e. sequences of primitive actions. Typically, a skill latent space and policy are discovered from offline data. However, the resulting low-level policy can be unreliable due to low-coverage demonstrations or distribution shifts. As a solution, we propose the *Skill-Critic* algorithm to fine-tune the low-level policy in conjunction with high-level skill selection. Our Skill-Critic algorithm optimizes both the low-level and high-level policies; these policies are initialized and regularized by the latent space learned from offline demonstrations to guide the parallel policy optimization. We validate Skill-Critic in multiple sparse-reward RL environments, including a new sparse-reward autonomous racing task in Gran Turismo Sport. The experiments show that Skill-Critic's low-level policy fine-tuning and demonstration-guided regularization are essential for good performance. Code and videos are available at our website: https://sites.google.com/view/skill-critic.**

I. Introduction

Reinforcement learning (RL) has demonstrated remarkable success in various domains [1], [2]. However, standard RL algorithms often lack the ability to incorporate prior structure, knowledge, or experience, which may be necessary for complex tasks [3]–[5]. Incorporating prior experience by learning from demonstrations can facilitate efficient exploration [6]. For example, statistical methods can infer the hidden structure of offline data and inform the decision-making process [4], [5]. However, offline data alone may not suffice for determining an optimal policy, particularly when the data originates from simpler environments or pertains to intricate or stochastic tasks. In such cases, online policy optimization (referred to as *fine-tuning*) is required to refine suboptimal policies [7], [8]. We present a hierarchical RL framework that leverages offline data to accelerate RL training without limiting its performance by the quality of offline data.

Our framework employs *skills*, temporally extended sequences of primitive actions [9]. Previous works extract skills from unstructured data and transfer them to downstream RL tasks with a skill selection policy whose action space is the skill itself [10]. Skill-Prior RL (SPiRL) [4] found learning a set of skills may not be adequate to guide skill selection; rather, exploration is improved when high-level skill selection is regularized by a data-informed prior distribution known as the skill prior. The skill prior informs the high-level policy, but the low-level policy, i.e. the skill, is *stationary*. However, with low-coverage or low-quality offline data, stationary skills may not suffice in complex downstream tasks.

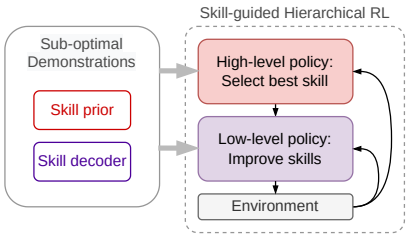


Fig. 1. Skill-Critic leverages low-coverage demonstrations to facilitate hierarchical reinforcement learning by (1) acquiring a basic skill-set from demonstrations that (2) guides online skill selection and skill improvement.

Our *Skill-Critic* approach shown in Fig. I aims to leverage parallel high-level and low-level policy optimization to refine the skills themselves during skill selection. Intuitively, agents can use their experience to *improve* their skill set, rather than being constrained to select skills from a stationary, offline library. We show this problem can be formulated as the parallel optimization of a high-level (HL) policy to select a skill and a low-level (LL) policy to select an action. We guide HL skill selection with a data-informed skill prior [4], and we extend this notion to initialize and regularize the LL skills using an action prior informed by offline data. Skill-Critic is reminiscent of discrete *options* [11]; however, the offline data-informed, continuous skill space adds a unique structure for guiding and stabilizing the parallel policy optimization.

Our contributions are: (1) We formulate parallel optimization of the HL and LL policies to simultaneously select skills and improve the skill set, (2) We use an action prior to guide LL policy fine-tuning to improve the offline data-driven skill set, and (3) Our method *improves* the skill set and performance in simulated navigation and robotic manipulation tasks and solves a new, sparse reward autonomous racing task in the complex Gran Turismo Sport environment.

II. Related Works

1) Skill-transfer RL: Skill-transfer RL reuses pre-trained skills, i.e. sequences of actions, to accelerate RL training for downstream tasks [4], [5], [12], [13]. One commonly used approach is to learn a skill latent space from offline data using

[1]Department of Mechanical Engineering, University of California Berkeley, CA, USA. Catherine Weaver is supported by NSF GFRP Grant No. DGE 1752814. {cehao, catherine22, chen_tang, tomizuka, wzhan}@berkeley.edu

[2]Sony Research Inc. Tokyo, Japan, kenta.kawamoto@sony.com. This work was supported by Sony R&D Center Tokyo and Sony AI. We are also very grateful to Polyphony Digital Inc. for enabling this research and providing GT Sport framework.

* Authors contributed equally.
† Correspondence to cehao@berkeley.edu

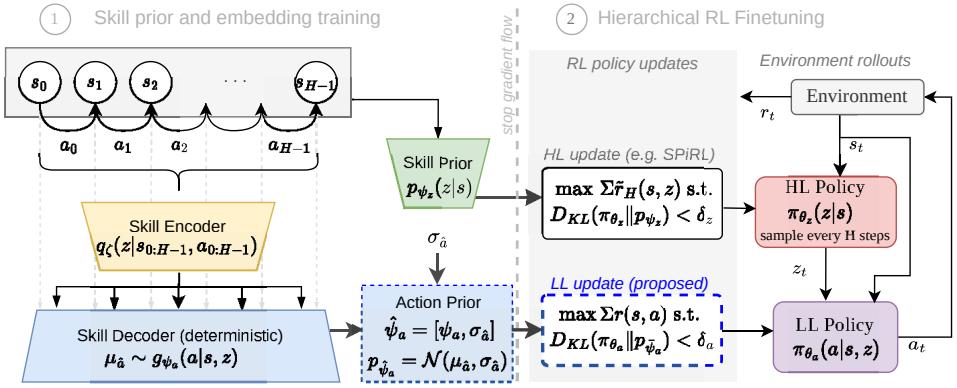


Fig. 2. Hierarchical RL from a demonstration-guided latent space. **Left:** Offline data informs the skill embedding model with skill encoder (**yellow**), skill prior (**green**), and skill decoder (**blue**). Hyperparameter $\sigma_{\hat{a}}$ is augmented to the decoder to define the action prior. **Right:** HL (**red**) and LL (**purple**) policies are fine-tuned on downstream tasks via our Skill-Critic algorithm. During fine-tuning, the HL and LL policies are regularized with the skill and action priors.

variational autoencoder (VAE) [9]. Then in downstream RL, an HL skill-selection policy is trained to select the optimal skill from the learned skill space. Thus, RL only needs to explore how to stitch temporally-extended action sequences, instead of searching for the optimal action at every time step. Extensions have learned priors for the HL policy from VAE training to guide and further accelerate RL training [4], [14], [15] and learned these skill priors from multiple datasets [16], [17]. However, prior works often consider a *stationary* skill space, constraining performance to skills learned from offline data.

2) Hierarchical RL: Hierarchical RL (HRL) decomposes long-horizon tasks into simpler sub-tasks, encouraging exploration during training [18]. Algorithms often employ intermediate variables, such as languages [19], goals, options, or skills, to define subdomains that bridge high and low levels. Discrete options [11], [20], [21] may not be sufficiently descriptive for complex tasks. Goal-conditioned HRL [22]–[26] leverages automatic goal sampling methods to train goal-conditioned policies; however, goals must be available from the state space. In Skill-transfer RL [12], [13], hierarchical policies use a data-informed, continuous latent space, potentially representing a wider range of behaviors. Recent works use residual policies to augment an LL data-driven skill decoder [27], [28]. Skill-Critic provides an alternate mechanism for parallel HL and LL policy optimization: the decoder is an *action prior* to guide the LL policy, and the skill prior guides the HL policy.

III. Approach

We employ demonstration-informed, temporally abstracted skills in hierarchical RL to facilitate learning complex long-horizon tasks [4]. The hierarchical policy consists of 1) a high-level (HL) policy, $\pi_z(z_t|s_t)$, that selects the best skill, z_t, given the current state, s_t; and 2) a low-level (LL) policy, $\pi_a(a_t|s_t, z_t)$, that selects the optimal action, a_t, given the state and selected skill. The HL policy selects skills from a learned continuous skill set $\mathcal{Z}$, i.e., $z_t \in \mathcal{Z}$. As in SPiRL [4], we consider a task in which we can extract an initial skill set from offline demonstrations to accelerate downstream RL training. We further consider cases where the extracted skill set is *insufficient* for the downstream tasks, motivating our *Skill-Critic* framework in Figure 2. In Stage 1, we leverage demonstrations to learn a skill decoder and skill prior that can accelerate RL training. In Stage 2, we leverage hierarchical RL to fine-tune *both* the HL and LL policies, thus further improving the inadequate offline-learned skill set.

A. Offline Skill Prior and Embedding Pre-Training (Stage 1)

We assume access to demonstrations consisting of trajectories $\mathcal{D} = (\tau_0, \tau_1, ..., \tau_N)$, which each include states and actions at each time step $\tau = (s_t, a_t, ...)$. The demonstrations may not contain complete solutions for the downstream task; however, there are skills that can be transferred from the offline data by training an HL policy to select the best skills for a new task. Furthermore, the demonstrations include only a subset of potential skills or suboptimal skills, motivating the improvement of skills with RL fine-tuning of the LL policy.

We directly follow SkilD [5], to embed a sequence of H consecutive actions $a_{0:H-1}$, known as a *skill*, into a latent space using a variational autoencoder (VAE) [9]. The VAE objective contains three parts: 1) a reconstruction loss to minimize the difference between demonstration actions a_k and those predicted by the decoder $\hat{a}_k = g_{\psi_a}(a|s, z)$; 2) regularization on the encoder $q_\zeta(z|s_{0:H-1}, a_{0:H-1})$ to align the latent distribution with a standard normal distribution; and 3) a KL-divergence term to train the skill prior $p_{\psi_z}(z|s_0)$ to match the posterior distribution inferred from the encoder. The first two terms are standard components of a VAE [9], and the third term trains the skill prior that can accelerate downstream RL [4]. We use a state-dependent decoder, $g_{\psi_a}(a|s, z)$ [5], and augment the state with a one-hot vector corresponding to the time since skill selection for a more informative policy.

The skill prior parameterizes a Gaussian distribution and can be used directly to initialize and regularize a downstream HL policy [4], [5]. In the next section, we extend this notion to an *action prior* that can initialize and regularize a downstream LL policy. The pre-trained (deterministic) skill decoder is an obvious choice, as previous works directly use the decoder as the LL policy [4], [5]. Thus, we define a Gaussian action prior, denoted as $p_{\bar{\psi}_a}(a|z, s) = \mathcal{N}(\mu_{\hat{a}}, \sigma_{\hat{a}})$, with mean given by the skill decoder: $\mu_{\hat{a}} = g_{\psi_a}$. While the variance, $\sigma_{\hat{a}}$, is a

hyperparameter, this action prior provides a more informative prior than SAC's entropy (unit Gaussian prior) [4].

B. Hierarchical Skill-Prior and Action-Prior Regularized RL Fine-tuning (Stage 2)

We present Skill-Critic (Alg. 1), which uses a parallel MDP structure to optimize the HL and LL policy with guidance from the pre-trained skill prior and action prior. To derive the parallel optimization of π_a and π_z, we first note that the learned skill space forms a semi-MDP endowed with skills in Section III-B.1. This semi-MDP formulation can be written as a parallel MDP formulation (Section III-B.2), so that we optimize the HL policy π_z on a "high-MDP" $M^{\mathcal{H}}$ and the LL policy π_a on a "low-MDP" $M^{\mathcal{L}}$. Finally, in Sections III-B.3 and III-B.4, the HL and LL policy optimizations are guided by the pre-trained skill prior, $p_{\psi_z}(z|s)$, and action prior, $p_{\bar{\psi}_a}(a|s,z)$. For each policy, we initialize the trained policy with the corresponding pre-trained prior policy and augment the objective function with the KL divergence between the trained policy and its prior. To stabilize hierarchical training, we train the HL policy for $N_{\text{HL-warm-up}}$ steps prior to training the LL policy.

Our formulation makes three notable improvements on prior works: 1.) we employ a *skill-based* parallel MDP formulation to update the HL and LL policies in parallel, 2.) we introduce a LL Q-function estimate using known relationships between state-action values on the MDPs to stabilize optimization, and 3.) we extend soft actor-critic (SAC) [29] with non-uniform priors [4] to guide the LL policy update with the action prior.

1) Semi-MDP endowed with skills: The hierarchical policies π_a and π_z form an MDP endowed with skills. We argue this is a semi-MDP similar to the one defined in the options framework [30], as skills are continuous, fixed-duration options. The state space $\mathcal{S}$ consists of the environment states that are augmented with a one-hot encoding of the time index since the beginning of the active skill, $k_t \doteq (t \mod H) \in \mathcal{K} \doteq \{0, 1, ...H-1\}$. Following options notation [11], the skill is a triple $(\mathcal{I}, \pi_a, \beta)$, with initiation set $\mathcal{I}$, intra-skill policy $\pi_a : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, and termination function $\beta : \mathcal{S} \to [0, 1]$. We set $\mathcal{I}$ to the subset of states in $\mathcal{S}$ where $k = 0$, meaning skills are only initiated after a fixed horizon H since the previous skill was initiated. The termination function is $\beta(s_t) \doteq \beta_t \doteq \mathbb{I}_{k_t=0}$, which takes value $\beta_t = 1$ when $k_t = 0$ and $\beta_t = 0$ otherwise. The semi-MDP consists of states, actions, skills, reward, transition probability, initial state distribution, and discount as listed in Table I.

To solve the RL objective, we adapt the value functions from option-critic [20] to continuous, fixed horizon skills:

$$\begin{aligned} V_z^{\Omega}(s_{t+1}) &= \mathbb{E}_{z_{t+1}\sim\pi_{\theta_z}}\left[Q_z^{\Omega}(s_{t+1}, z_{t+1})\right] \\ Q_z^{\Omega}(s_t, z_t) &= \mathbb{E}_{a_t\sim\pi_{\theta_a}}\left[Q_a^{\Omega}(s_t, z_t, a_t)\right] \\ Q_a^{\Omega}(s_t, z_t, a_t) &= r(s_t, a_t) + \gamma\mathbb{E}_{a_{t+1}\sim\pi_{\theta_a}, \pi_{\theta_z}}\left[U(z_t, s_{t+1})\right] \\ U(z_t, s_{t+1}) &= [1-\beta_{t+1}]\, Q_z^{\Omega}(s_{t+1}, z_t) + \beta_{t+1} V_z^{\Omega}(s_{t+1}). \end{aligned} \tag{1}$$

Here V_z^{Ω} is the value of the state s_t, Q_z^{Ω} is the value of selecting skill z_t from s_t, and Q_a^{Ω} is the value of selecting action a_t from state s_t and skill z_t. Simplifying for fixed horizon skills, when $\beta(s_{t+1}) = 0$ during rollout of a skill,

$$Q^{\Omega}_{a,\beta_{t+1}=0} = r(s_t, a) + \mathbb{E}_{\pi_a,\pi_z}[Q_a^{\Omega}(s_{t+1}, z_{t+1}, a_{t+1})]. \tag{2}$$

and when $\beta(s_{t+1}) = 1$ at selection of the next skill

$$Q^{\Omega}_{a,\beta_{t+1}=1} = r(s_t, a) + \mathbb{E}_{\pi_a,\pi_z}[Q_z^{\Omega}(s_{t+1}, z_{t+1})]. \tag{3}$$

2) Formulation as two augmented MDPs: The semi-MDP requires special algorithms [20], which are difficult to augment with skill and action prior regularization. Rather, we re-formulate the semi-MDP as two parallel augmented MDPs by adapting an options-based parallel MDP framework, Double Actor Critic (DAC) [11], to our continuous, fixed-horizon skills. Thus, we can use standard RL algorithms for each policy [11]. Table I derives the formulation with a similar notation to [11, Sec. 3] by replacing discrete options, $O \sim \mathcal{O}$, with skills, $z \sim \mathcal{Z}$. The HL policy π_z selects skills in the high-MDP $M^{\mathcal{H}}$, and the LL policy π_a selects actions in the low-MDP $M^{\mathcal{L}}$.

To form the high-MDP $M^{\mathcal{H}}$, the state is composed of the current state and skill $\mathbf{s}_t^{\mathcal{H}} \doteq (z_{t-1}, s_t)$, and the action is the next skill $\mathbf{a}_t^{\mathcal{H}} \doteq z_t$. We define p_z as the transition probability function from the current state and skill to the next state:

$$p_z(s_{t+1}|s_t, z_t) \doteq \mathbb{E}_{a\sim\pi_a(a|s_t,z_t)}\left[p(s_{t+1}|s_t, a)\right]. \tag{4}$$

Eq. 4 is analogous to the first equation in Section 2 of DAC; however, we define p_z by taking the expectation over actions instead of using a discrete probabilistic estimate. The transition probability, $p^{\mathcal{H}}$, on $M^{\mathcal{H}}$ is defined with p_z in Table I, where $\mathbb{I}$ is the indicator function. In the initial distribution $p_0^{\mathcal{H}}$, it is not necessary to define a dummy skill [11], since we always start with skill selection from $\pi_z(z_t|s_t)$ at $t = 0$.

Since $\pi^{\mathcal{H}}$ executes a skill for H time steps, at which point π_{θ_z} selects a new skill, we define an H-step reward:

$$\tilde{r}_H(s_t, z_t) \doteq \mathbb{E}_{a_t\sim\pi_a(a|s_t,z_t)}\left[\Sigma_{\tau=t}^{t+H-1} r(s_\tau, a_\tau)\right], \tag{5}$$

where $\tilde{r}_H(s_t, z_t)$ is the sum of rewards when executing z_t from s_t for H steps. The corresponding RL objective on $M^{\mathcal{H}}$ (Table 1) maximizes the sum of H-step rewards $\tilde{r}_H$ with discount factor γ_z, which is valid when $\tilde{r}_H$ is evaluated only at instances when skills change ($\beta(s) = 1$) [4]. This formulation

Algorithm 1 The Skill-Critic RL Algorithm

Inputs: Skill prior $p_{\psi_z}(z|s)$ and action prior $p_{\bar{\psi}_a}(a|s,z)$, which are pre-trained via Section III-A [4] on the offline dataset $\mathcal{D}$.
for each iteration $i = 0, 1, 2, ...$ **do**
  for each environment step **do**
    if $t \mod H == 0$ **then**
      Sample skill $z \sim \pi_{\theta_z}(\cdot|s)$
    Sample action $a_t \sim \pi_{\theta_a}(\cdot|s_t, z)$
    Perform action; add $\{s, z, a, r, s'\}_t$ to replay buffer
  for $t = 0, H, 2H, ...$ and $t^* \doteq t + H$ **do**
    Update HL policy, critic, and temperature towards Eqn. (8) (i.e SPiRL [4])
  for $t = 0, 1, 2, ...$ and $t' \doteq t + 1$ **do**
    Update LL policy, critic, and temperature towards Eqn. (10) via Algorithm 2 **if** $i \geq N_{\text{HL-warm-up}}$
Return trained HL policy π_{θ_z} and LL policy π_{θ_a}

TABLE I

MDP FORMULATIONS FOR SOLVING SKILL-BASED HRL

	Original semi-MDP, M^{Ω}	**High-MDP**, $M^{\mathcal{H}}$	**Low-MDP**, $M^{\mathcal{L}}$
MDP Tuple	$M^{\Omega} \doteq \{\mathcal{S}, \mathcal{A}, \mathcal{Z}, p, p_0, r, \gamma\}$	$M^{\mathcal{H}} \doteq \{\mathcal{S}^{\mathcal{H}}, \mathcal{A}^{\mathcal{H}}, p^{\mathcal{H}}, p_0^{\mathcal{H}}, r^{\mathcal{H}}, \gamma_z\}$	$M^{\mathcal{L}} \doteq \{\mathcal{S}^{\mathcal{L}}, \mathcal{A}^{\mathcal{L}}, p^{\mathcal{L}}, p_0^{\mathcal{L}}, r^{\mathcal{L}}, \gamma\}$
State Space	$s_t \in \mathcal{S}$	$\mathbf{s}_t^{\mathcal{H}} \doteq (z_{t-1}, s_t) \in \mathcal{S}^{\mathcal{H}}$	$\mathbf{s}_t^{\mathcal{L}} \doteq (s_t, z_t) \in \mathcal{S}^{\mathcal{L}}$
Action Space	$a_t \in \mathcal{A}$	$\mathbf{a}_t^{\mathcal{H}} \doteq z_t \in \mathcal{A}^{\mathcal{H}}$	$\mathbf{a}_t^{\mathcal{L}} \doteq a_t \in \mathcal{A}^{\mathcal{L}}$
Latent Space	$z_t \in \mathcal{Z}$	-	-
Transition Probability	$p(s_{t+1}\|s_t, a_t)$	$p^{\mathcal{H}}\left(\mathbf{s}_{t+1}^{\mathcal{H}} \mid \mathbf{s}_t^{\mathcal{H}}, \mathbf{a}_t^{\mathcal{H}}\right)$ $\doteq p^{\mathcal{H}}\left((z_t, s_{t+1}) \mid (z_{t-1}, s_t), a_t^{\mathcal{H}}\right)$ $\doteq \mathbb{I}_{\mathbf{a}_t^{\mathcal{H}}=z_t} p_z\left(s_{t+1} \mid s_t, z_t\right)$	$p^{\mathcal{L}}\left(\mathbf{s}_{t+1}^{\mathcal{L}} \mid \mathbf{s}_t^{\mathcal{L}}, \mathbf{a}_t^{\mathcal{L}}\right)$ $\doteq p^{\mathcal{L}}\left((s_{t+1}, z_{t+1}) \mid (s_t, z_t), a_t\right)$ $\doteq p\left(s_{t+1} \mid s_t, a_t\right) p\left(z_{t+1} \mid s_{t+1}, z_t\right)$
Init. Distribution	$p_0(s_0)$	$p_0^{\mathcal{H}}(\mathbf{s}_0^{\mathcal{H}}) \doteq p_0^{\mathcal{H}}((z_{-1}, s_0)) \doteq p_0(s_0)$	$p_0^{\mathcal{L}}(\mathbf{s}_0^{\mathcal{L}}) \doteq p^{\mathcal{L}}((s_0, z_0)) \doteq \pi_z(z_0\|s_0) p_0(s_0)$
Reward	$r(s_t, a_t)$	$r^{\mathcal{H}}(\mathbf{s}_t^{\mathcal{H}}, \mathbf{a}_t^{\mathcal{H}}) \doteq r^{\mathcal{H}}((z_{t-1}, s_t), z_t)$ $\doteq \tilde{r}_H(s_t, z_t)$	$r^{\mathcal{L}}(\mathbf{s}_t^{\mathcal{L}}, \mathbf{a}_t^{\mathcal{L}}) \doteq r^{\mathcal{L}}((s_t, z_t), a_t)$ $\doteq r(s_t, a_t)$
RL Objective	$\mathbb{E}_{\pi_a, \pi_z}[\Sigma_{t=0}^{T-1} \gamma^t r_t]$, $\gamma = .99$	$\mathbb{E}_{\pi_a, \pi_z}[\Sigma_{t=0}^{T-1} \gamma_z^t \tilde{r}_H(s_t, z_t)]$, $\gamma_z = .99$	$\mathbb{E}_{\pi_a, \pi_z}[\Sigma_{t=0}^{T-1} \gamma^t r(s_t, a_t)]$, $\gamma = .99$
Q & Value Functions	$V_z^{\Omega}(s_{t+1})$: state value $Q_z^{\Omega}(s_t, z_t)$: value of (s_t, z_t) $Q_a^{\Omega}(s_t, z_t, a_t)$: value of a_t in (s_t, z_t)	$Q^{\mathcal{H}}(\mathbf{s}_t^{\mathcal{H}}, \mathbf{a}_t^{\mathcal{H}}) \doteq Q^z(s_t, z_t)$ $V^{\mathcal{H}}(\mathbf{s}_t^{\mathcal{H}}) \doteq V^{\mathcal{H}}(s_t)$ applied every H steps	$Q^{\mathcal{L}}(\mathbf{s}^{\mathcal{L}}, \mathbf{a}^{\mathcal{L}}) \doteq Q^a(s, z, a)$ $V^{\mathcal{L}}(\mathbf{s}_t^{\mathcal{L}}) \doteq V^{\mathcal{L}}((s_t, z_t))$ applied every 1 step

is a slight deviation from DAC's single-step reward, but it improves performance on long-horizon tasks [4].

We define the Markov policy $\pi^{\mathcal{H}}$ on $M^{\mathcal{H}}$ as

$$\begin{aligned}\pi^{\mathcal{H}}\left(\mathbf{a}_t^{\mathcal{H}} \mid \mathbf{s}_t^{\mathcal{H}}\right) &\doteq \pi^{\mathcal{H}}\left(z_t \mid (z_{t-1}, s_t)\right) \\ &\doteq (1-\beta(s_t))\mathbb{I}_{z_{t-1}=z_t} + \beta(s_t)\pi_z\left(z_t \mid s_t\right).\end{aligned} \tag{6}$$

Eq. (6) shows that the previous skill is used until $\beta(s_t) = 1$; then a new skill is selected via π_z. Unlike DAC, we use β's definition to simplify (6) to only be a function of π_z and β.

In the low-MDP, $M^{\mathcal{L}}$, the state is composed of the current state and next skill, $\mathbf{s}_t^{\mathcal{L}} \doteq (s_t, z_t)$, and the action is the action $\mathbf{a}^{\mathcal{L}} \doteq a_t$. The transition probability, initial distribution, and reward directly follow using our definition of β and DAC.

We define a Markov policy $\pi^{\mathcal{L}}$ on $M^{\mathcal{L}}$ as the LL policy π_a:

$$\pi^{\mathcal{L}}\left(\mathbf{a}_t^{\mathcal{L}} \mid \mathbf{s}_t^{\mathcal{L}}\right) \doteq \pi^{\mathcal{L}}\left(a_t \mid (s_t, z_t)\right) \doteq \pi_a\left(a_t \mid s_t, z_t\right). \tag{7}$$

It follows that when we fix π_a and optimize $\pi^{\mathcal{H}}$, we are optimizing π_z. Likewise, when we fix π_z and optimize $\pi^{\mathcal{L}}$, we are optimizing π_a [11]. This implies that any policy optimization algorithm can be used to optimize $\pi^{\mathcal{H}}$ and $\pi^{\mathcal{L}}$.

3) High-MDP Policy Optimization: To optimize $\pi^{\mathcal{H}}$ on $M^{\mathcal{H}}$, policy π_z is parameterized by θ_z (denoted π_{θ_z}), and we fix $\pi^{\mathcal{L}}$. The skill prior, p_{ψ_z}, is a prior distribution for π_{θ_z}, and θ_z is initialized with ψ_z. The *HL update* in Fig. 2 solves

$$\begin{aligned}\underset{\theta_z}{\operatorname{argmax}}\, \mathbb{E}_{\pi_{\theta_z}, \pi^{\mathcal{L}}} \Bigg[\sum_{t=\{0,H,2H,\ldots\}}^{\infty} \gamma_z^t \Big(\tilde{r}_H(s_t, z_t) \\ -\alpha_z D_{KL}[\pi_{\theta_z}(z_t|s_t) \| p_{\psi_z}(z_t|s_t)] \Big) \Bigg].\end{aligned} \tag{8}$$

where the regularization term is weighted with temperature α_z. We solve (8) using the Bellman operator on $M^{\mathcal{H}}$:

$$\begin{aligned}\mathcal{T}^{\pi_z} Q^{\mathcal{H}}\left(s_t, z_t\right) &= \tilde{r}_H\left(s_t, z_t\right) + \gamma_z \mathbb{E}_{p^{\mathcal{H}}}\left[V^{\mathcal{H}}\left(s_{t+1}\right)\right] \\ V^{\mathcal{H}}\left(s_t\right) &= \mathbb{E}_{z_t \sim \pi_z}\big[Q^{\mathcal{H}}\left(s_t, z_t\right) \\ &\quad - D_{\mathrm{KL}}\left[\pi_z\left(z_t|s_t\right) \| p_{\psi_z}\left(z_t|s_t\right)\right]\big].\end{aligned} \tag{9}$$

The value function $V^{\mathcal{H}}$ and Q-function $Q^{\mathcal{H}}$ are defined on $M^{\mathcal{H}}$ using the Bellman operator in SPiRL [4], which is proven to solve (8) by adapting SAC [29]. The H-step reward, $\tilde{r}_H$, implies $Q^{\mathcal{H}}(\mathbf{s}_t^{\mathcal{H}}, \mathbf{a}_t^{\mathcal{H}}) \doteq Q^z(s_t, z_t)$, and the state-skill value estimation is discounted every H steps, which can improve performance by increasing the effect of sparse rewards [4].

4) Low-MDP Policy Optimization: To optimize $\pi^{\mathcal{L}}$ on $M^{\mathcal{L}}$, policy π_a is parameterized by θ_a (denoted π_{θ_a}), and we fix $\pi^{\mathcal{H}}$. The action prior learned from offline demonstrations, $p_{\bar{\psi}_a}$, is a prior distribution for π_{θ_a}, and θ_a is initialized with $\bar{\psi}_a$. The regularized objective with temperature α_a is

$$\begin{aligned}\underset{\theta_a}{\operatorname{argmax}}\, \mathbb{E}_{\pi_{\theta_a}, \pi^{\mathcal{H}}} \Bigg[\sum_{t=0}^{\infty} \gamma^t \Big(r\left(s_t, a_t\right) \\ -\alpha_a D_{KL}\left[\pi_{\theta_a}(a_t|s_t, z_t) \| p_{\bar{\psi}_a}(a_t|s_t, z_t,)\right] \Big) \Bigg],\end{aligned} \tag{10}$$

corresponding to the *LL update* in Fig. 2. Algorithm 2 updates the LL policy by adapting KL-divergence regularized SAC to solve (10). We define the Bellman operator on $M^{\mathcal{L}}$:

$$\begin{aligned}\mathcal{T}^{\pi_a} Q^{\mathcal{L}}\left((s_t, z_t), a_t\right) &= r\left(s_t, a_t\right) + \gamma \mathbb{E}_{p^{\mathcal{L}}}\left[V^{\mathcal{L}}\left((s_{t+1}, z_{t+1})\right)\right] \\ V^{\mathcal{L}}\left((s_t, z_t)\right) &= \mathbb{E}_{a_t \sim \pi_a}\big[Q^{\mathcal{L}}\left((s_t, z_t), a_t\right) \\ &\quad - D_{\mathrm{KL}}\left[\pi_a\left(a_t|(s_t, z_t)\right) \| p_{\bar{\psi}_a}\left(a_t|(s_t, z_t)\right)\right]\big].\end{aligned} \tag{11}$$

Similar to the HL update, the entropy regularization in SAC is replaced with the deviation of the policy π_{θ_a} from the prior $p_{\bar{\psi}_z}$. Dual gradient descent on the temperature α_a [4], [29] ensures that the expected divergence between LL policy and the action prior is equal to the chosen target divergence δ_a on Line 9. The LL Q-value, $Q^a(s, z, a)$, estimates the value of the state-skill pair and action with 1-step discounting (10).

When estimating Q^a with (11), far-horizon rewards have an exponentially diminishing effect. Practically, this led to poor performance with sparse rewards. Inspired by the relationship between Q-functions on the semi-MDP (1), we investigate if the longer-horizon HL Q-value can inform LL policy optimization of the value that the HL policy is assigning to state-skill pairs. Thus, sparse rewards propagate to earlier

Algorithm 2 Skill-Critic Low-MDP Update

1: **Inputs:** Current iteration's HL parameters: $\bar{\phi}_z, \theta_z$; priors p_z, p_ψ, hyperparameters
2: **for** each $t = 0, 1, 2, ...$ and $t' = t + 1$ in buffer **do**
3: **if** $k_t == H - 1$, (where $k_t = t \mod H$) **then**
4: $\bar{Q}^a = Q^a_{\beta(s')=1}(s, z, a)$ using $Q^z_{\bar{\phi}_z}$ in (12)
▷ Estimate LL Q-value *upon arrival* to new skill
5: **else**
6: $\bar{Q}^a = Q^a_{\beta(s')=0}(s, z, a)$ using $Q^z_{\bar{\phi}_a}$ in (13)
▷ Estimate LL Q-value within current skill
7: $\theta_a \leftarrow$ step on (10) using $Q^a_{\phi_a}$ ▷ update LL policy parameters
8: $\phi_a \leftarrow \phi_a - \lambda_{Q_a} \nabla_{\phi_a} [\frac{1}{2}(Q_{\phi_a}(s_t, a_t, z_t) - \bar{Q}_a)^2]$
▷ update LL critic weights
9: $\alpha_a \leftarrow \lambda_{\alpha_a} \left[\alpha_a (D_{KL}(\pi_{\theta_a}(a_t|s_t, z_t) \| p_{\bar{\psi}_a}(a_t|s_t, z_t)) - \delta_a)\right]$
▷ update LL alpha
10: $\bar{\phi}_a \leftarrow \tau \phi_a + (1 - \tau)\bar{\phi}_a$ ▷ update LL target network weights
11: **Return** trained low-level policy π_{θ_a}

states. We observe that V^Ω_z is analogous to the value of skills, meaning V^Ω_z is analogous to the value function on the high-MDP, $V^\mathcal{H}$. Likewise, Q^Ω_z is analogous to the value of state-skill pairs, meaning Q^Ω_z is analogous to the value function on the low-MDP, $V^\mathcal{L}$. The semi-MDP (1) does not include policy regularization. To incorporate our non-uniform prior into Q^a estimation, we note that the regularization terms in (9) and (11) appear in the definition of $V^\mathcal{H}$ and $V^\mathcal{L}$. We follow this precedent as we introduce regularization into (2) and (3). We define

$$Q^a_{\beta(s_{t+1})=1}(s_t, z_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{\pi_z}\big[Q^z(s_{t+1}, z_{t+1}) - \alpha_z D_{KL}[\pi_{\theta_z}(z_t|s_t) \| p_{\psi_z}(z_t|s_t)]\big], \tag{12}$$

to estimate Q^a at the end of a skill (Line 4, Alg. 2) and

$$Q^a(s_t, z_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{\pi_a, \pi_z}\big[Q^a(s_{t+1}, z_{t+1}, a_{t+1}) - \alpha_a D_{KL}\left[\pi_{\theta_a}(a_t|s_t, z_t) \| p_{\bar{\psi}_a}(a_t|s_t, z_t,)\right]\big], \tag{13}$$

otherwise. While the semi-MDP assumptions do not strictly hold for the regularized objective (10), we find that using Q^z to estimate Q^a leads to faster, stable convergence.

IV. Experiments

We assess Skill-Critic in three tasks: maze navigation, racing, and robotic manipulation (Fig. 3). Please refer to our websites for demo videos. In each task, Stage 1 consists of collecting an offline dataset to create an informative skill set; however, it may not encompass *all* of the skills necessary for downstream tasks. Stage 2 consists of a sparse-reward episodic RL task. A binary reward (+1) is received at each time step after the goal is reached. The objective is to maximize the sum of rewards, i.e. complete the task *as fast as possible*.

We compare Skill-Critic to several baselines. **SPiRL** [4] extracts temporally extended skills with a skill prior from offline data, but downstream RL training occurs only on the HL skill selection policy. A state-dependent policy is used to improve performance [5]. **Skill-Critic** is warm-started with SPiRL at $N_{\text{HL-warm-up}}$ steps to stabilize the HL policy prior to LL policy improvement. **ReSkill** [27] is a HRL method with a residual policy that augments the stationary LL policy. This contrasts Skill-Critic, which directly fine-tunes the nonstationary LL policy with regularization to the stationary action prior. ReSkill uses independent HL and LL policy updates, but Skill-Critic relates the LL Q-function to the HL Q-function upon arrival at the next skill (13). Baselines include soft actor-critic (**SAC**) [29] and SAC initialized with behavioral cloning (**BC+SAC**).

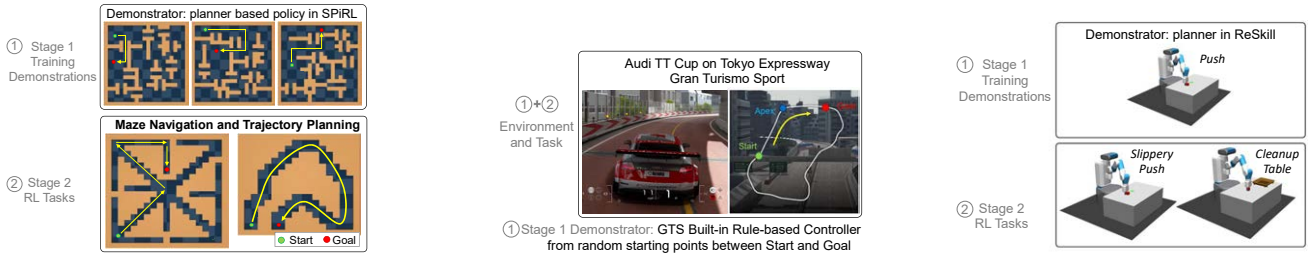
A. Maze Navigation and Trajectory Planning

The maze task tests if Skill-Critic can improve its LL policy and leverage the action prior to learn challenging, long-horizon tasks. The task uses the D4RL point maze [31] with a top-down agent-centric state and continuous 2D velocity as the action. Demonstrations consist of 85000 goal-reaching trajectories in randomly generated, small maze layouts (Fig. 3a). The demonstration planner [4] acts in right angles (i.e. goes up, down, left or right). For Stage 2 downstream learning, we introduce two new maze layouts: 1) a *Diagonal Maze* to test how well the agent can navigate a maze with unseen passages, and 2) a *Curvy Tunnel* with multiple options for position, heading, and velocity to test how well the agent can plan an optimal trajectory. In both layouts, the agent has 2000 step episodes and receives a $+1$ reward for each time step that the distance to the goal is below a threshold. Unlike SPiRL [4], in our maze layouts, the agent can improve its performance by moving diagonally and following a smooth path.

In both maze tasks in Fig. 4, SAC and BC+SAC fail to reach the goal, likely because the single-step policy cannot discover the sparse reward. Although ReSkill leverages the skill embedding to guide exploration, the LL residual policy update is independent of the HL update, and ReSkill eventually fails to reach the goal. Noting the maze tasks have a longer horizon than the robotic tasks [27], we hypothesize the distant goal's reward signal is too weak to guide the LL residual policy (see ablation in Section IV-D.1). In contrast, SPiRL and Skill-Critic use the offline demonstrations and H-step reward to reach the goal. Fig. 4 compares the trajectories of Skill-Critic and SPiRL. SPiRL plans slow, jagged trajectories because it cannot improve the offline-learned LL policy. Skill-Critic updates the LL policy to further optimize its path, resulting in planning a significantly faster trajectory. Interestingly, Skill-Critic *discovers* diagonal motion, but it still does not forget to solve the maze because LL policy exploration is guided by the action prior.

B. Autonomous Racing

The vehicle racing task tests if Skill-Critic can 1.) improve the LL policy when there is only access to low-coverage, low-quality demonstrations, and 2.) leverage the skill and action prior to accelerate learning a sparse reward. We employ the Gran Turismo Sport (GTS) high-fidelity racing simulator to solve a new, sparse-reward racing task. The low-dimensional state [32] includes pose, velocity, and track information. There are two continuous actions: steering angle and a combined



(a) *Diagonal Maze* & *Curvy Tunnel* Maze Tasks (b) Gran Turismo Sport (GTS) Racing Task (c) 7DoF Robotic Manipulation Tasks

Fig. 3. Demonstrations and experiments. (a) Maze Tasks: Stage 1 demonstration uses the planner in SPiRL [4]. Stage 2 tasks test the agent's navigation in a *Diagonal Maze* and path planning in a *Curvy Tunnel*. (b) GTS Racing on a single corner. The agent achieves +1 after the goal state is passed. Demonstrations start at random low-speed starting points on the course. (c) Robotic Manipulation: Stage 1 demonstrations use a hand-crafted controller [27] to push a block across a table. Stage 2 RL tasks are *Slippery Push*, which uses a more slippery surface, and *Cleanup Table*, which includes a tray as an obstacle.

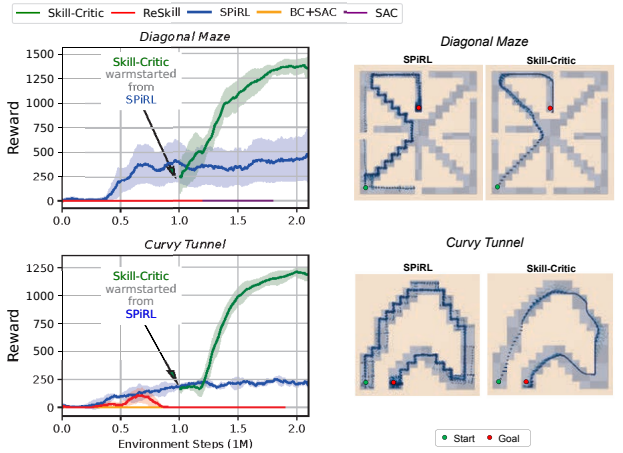
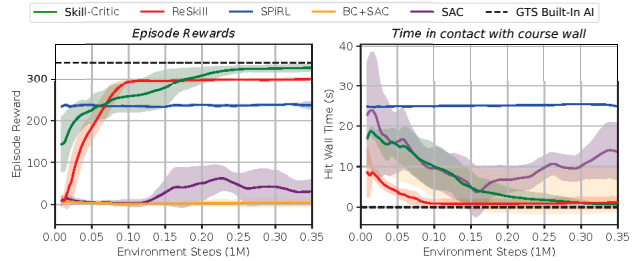


Fig. 4. Maze results. **Left:** Rewards. Skill-Critic starts training at $N_{\text{HL-warm-up}}$=1M steps. **Right:** Trajectories after policies converge. SPiRL reuses right-angle skills, but Skill-Critic plans diagonal and curved paths.



(a) Episode rewards and hit wall time during training

Algorithm	Finish Time (s)
Skill-Critic	27.2±0.6
ReSkill	29.9±0.2
SPiRL	36.4±0.9

Algorithm	Finish Time (s)
BC+SAC	59.6±0.7
SAC	56.8±2.9
Built-In AI	26.1

(b) Time to finish course at convergence

Fig. 5. GTS Racing Results. **Left:** mean (std) episode reward. **Right:** mean (std) of cumulative time in contact with track boundary per episode. SPiRL does not improve, so Skill-Critic does not use warm-up: $N_{\text{HL-warm-up}} = 0$.

throttle/brake command between $[-1, 1]$. The agent starts at a low speed in the center of the track and has 600-step (60-sec) episodes; the agent receives a binary +1 reward at each time step after it passes the goal. We use GTS's Built-in AI controller to collect 40000 *low-speed* demonstrations from random starting points on the course, each 200 steps in length. The agent can transfer skills such as speeding up and turning but must drive at higher speeds to rapidly navigate the course.

We compare performance to GTS's Built-in AI, which is a high-quality, rule-based controller deployed with the game as a competitor for players. Note that we deliberately give Skill-Critic access to low-speed demonstrations from Built-in AI to test Skill-Critic's ability to improve skills. Unlike previous works in GTS with dense rewards that must be uniquely designed for each car and track [2], [32], we use a generalizable sparse reward in a single corner (Fig. 3b). Given these factors, we consider Built-in AI a strong baseline.

In Fig. 5, we compare (a) rewards, which indicates how fast the car completes the course, and time in contact with the wall at the edge of the track, which indicates the car's dynamic stability, and (b) the converged policy's time to finish the corner. SAC reaches the goal in spite of its single-step policy, but it is slow to improve with the sparse reward. BC+SAC appears to hinder exploration, consistently crashing in the first straight-away. In contrast, SPiRL exploits the pre-trained skills to reach the goal. However, skills are learned from low-speed demonstrations, so the stationary LL policy may only be capable of low-speed maneuvers. Thus, SPiRL cannot plan high-speed trajectories and collides with the wall.

Both Skill-Critic and ReSkill address these issues to achieve high rewards and reduce contact time with course walls. Both methods exploit offline pre-training and temporally extended actions to guide exploration and maintain knowledge of the sparse reward. Online LL fine-tuning is critical to learn high-velocity maneuvers, such as collision avoidance and sharp cornering. However, ReSkill's LL residual policy update, which is independent of the value assigned by the HL update, does not improve the LL policy at states early in the rollout, resulting in a lower finish time. Conversely, Skill-Critic races close to the speed of the Built-in AI. We attribute this to the interrelated Q-function update that estimates the LL Q function using the H-step reward *upon arrival* to a skill. As shown in IV-D.1, sparse rewards propagate further into the LL policy update, yielding higher state values. In the videos, SPiRL is slow and collision-prone, and ReSkill's steering oscillates. In contrast, Skill-Critic races in a faster and more stable manner.

C. Robot Manipulation

Finally, we test a sparse-reward robot manipulation task with a 7-DoF Fetch robotic arm simulated in MuJoCO [33]. Handcrafted controllers [27] collect 40k demonstration

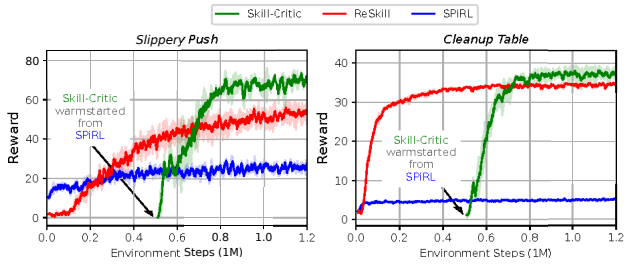


Fig. 6. Robotic Manipulation results. Mean episode reward (std). Skill-Critic employs $N_{\text{HL-warm-up}}$=500k steps. Left: Slippery Push, Right: Cleanup Table

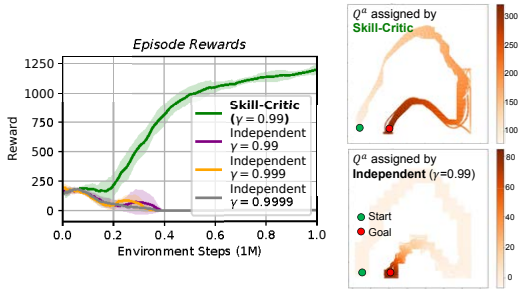


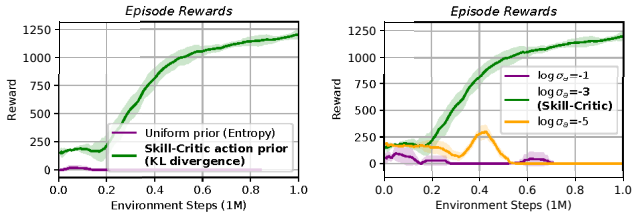
Fig. 7. Ablation of Q^a update in *Curvy Tunnel* ($N_{\text{HL-warm-up}}$=1M, 3 seeds). **Independent** update of $Q^a_{\beta(s_{t+1})=1}$ from current Q^a estimate versus **Skill-Critic** update of $Q^a_{\beta(s_{t+1})=1}$ from current Q^z estimate. **Left**: training episode rewards. **Right:** value distribution of trajectories at convergence.

trajectories, where the robot must *Push* a block along a table (Fig. 3c). For the Stage 2 RL tasks, we test *Slippery Push* and *Cleanup Table* tasks [27]. In *Slippery Push* the agent must push a block to a goal 100 step episodes, but the friction of the table surface is reduced from that seen in the demonstrations. The agent receives a reward of 1 once the block is at the goal location, otherwise the reward is 0; episodes are 100 steps. For *Cleanup Table* task, the agent must place a block on a rigid tray object, which was not present in the demonstrations. The agent receives a reward of 1 only when the block is placed on the tray, otherwise the reward is 0; episodes are 50 steps.

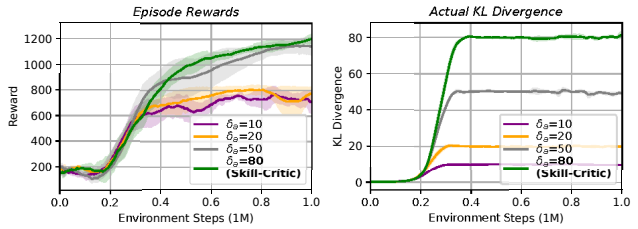
ReSkill outperforms other hierarchical methods like Hierarchical Actor Critic (HAC) [34] and PARROT [35], which do not learn anything meaningful (see results in [27]). In comparison to SPiRL and Skill-Critic (Fig. 6), ReSkill speeds up exploration with its alternative skill embedding that biases the HL policy towards relevant skills. However, Skill-Critic achieves a higher reward by completing the task even *faster*. As shown in the demo videos, ReSkill corrects SPiRL's pre-trained policy that aggressively pushes the block, but Skill-Critic is the fastest to push the block to the goal. Interesting future research could apply ReSkill's alternative skill embedding to Skill-Critic, but we believe Skill-Critic's LL policy update is crucial to converge to the highest reward.

D. Ablation Studies

1) LL Q Function Estimate: **Skill-Critic** uses the value *upon arrival* to a new skill to estimate Q_a by using $Q_{\bar{\phi}_z}$(Line 4 of Algorithm 2). Namely, when $\beta(s_{t+1}) = 0$, Q^a is estimated with single-step discounting (13), but when $\beta(s_{t+1}) = 1$, Q^a is estimated with H-step discounting (12).



(a) Ablation: Prior Distribution (b) Ablation: Action Prior Variance



(c) Ablation Study of KL Divergence

Fig. 8. Ablation studies of LL policy regularization in *Diagonal Maze* ($N_{\text{HL-warm-up}} = 1$M, 3 seeds). **(a):** LL policy prior distribution: uniform prior (entropy) or proposed nonuniform prior (KL divergence). **(b):** Variance of the action prior, $\sigma_{\hat{a}}$. **(c)**: LL policy target KL divergence, δ_a, training rewards (left) and actual KL divergence during training (right).

In Fig. 7, we compare this to "independent" estimation of Q^a and Q^z in *Curvy Tunnel*. **Independent** refers to estimating Q^a with (13) regardless of the value of $\beta(s_{t+1})$. The high-MDP policy, π_z, and critic, Q^z, are warm-started for 1M steps of SPiRL, then π_a and π_z are trained (Algorithm 1) for an additional 1M steps. In Fig. 7, the **independent** LL Q-value hinders exploration, and eventually, the policy can no longer find the goal. We hypothesize the distant goal's reward signal is too weak to guide the policy at states early in the roll-out due to single-step exponential discounting (13) even for larger values of γ. **Skill-Critic** includes Q^z in the estimate of Q^a, with two benefits: 1) the H step discounting of Q^z is less prone to losing the sparse reward signal at early states, and 2) the LL policy update uses state-skill values assigned by the HL policy. The ablation also informs why $N_{\text{HL-warm-up}} > 0$ is necessary for success in the maze and robot tasks, as HL warm-up allows accurate Q^z estimates.

2) LL Policy Regularization: Fig. 8, provides an ablation on LL policy regularization in *Diagonal Maze*. All methods use $N_{\text{HL-warm-up}} = 1$M, then are trained via Skill-Critic with the specified hyperparameter. In Fig. 8(a), we replace Skill-Critic's non-uniform action prior divergence term with a LL policy update with a uniform prior [29], which is identical to entropy regularization [4]. A uniform prior leads to poor exploration, as entropy encourages random actions that are not guided to the sparse reward. Fig. 8(b) changes variance of the action prior, $\sigma_{\hat{a}}$, which determines policy variation from the pre-trained decoder (III-A). Small values (e.g. $\log \sigma_{\hat{a}} = -5$), over-constrain the LL policy. However, with large variance, e.g. $\log \sigma_{\hat{a}} = -1$, the agent forgets the pre-trained skills. A suitable value is $\log \sigma_{\hat{a}} = -3$, which promotes exploitation of the decoder and exploration to improve skills. In Fig. 8(c) we compare rewards for varying values of δ_a and the actual KL divergence of the LL policy from the action prior during

training. As explained in Algorithm 2 [Line 9], α_a is a dual descent parameter to constrain the LL policy's divergence to the target divergence δ_a [4], [29]. As δ_a increases, rewards likewise increase as the LL policy has freedom to deviate from the action prior. The LL policy divergence does converge to δ_a, but in early training (<.2M steps), KL divergence is relatively low for the initial α_a in [4], [29]. Thus, the initial α_a may also be an important hyperparameter for stable training.

V. Conclusion

We proposed Skill-Critic, a hierarchical skill-transfer RL algorithm, to perform two parallel policy optimization updates for skill selection and skill fine-tuning. We show that Skill-Critic can effectively leverage low-coverage and low-quality demonstrations to accelerate RL training, which is difficult with existing skill-transfer RL methods with stationary LL policies. In our experiments, our method solves maze navigation tasks that require exploring new skills online. Also, Skill-Critic outperforms existing methods on a challenging sparse-reward autonomous racing task and robotic manipulation task with the help of low-quality, non-expert demonstrations.

Limitations and Future Work: Skill-Critic reformulates hierarchical RL as two parallel MDPs. Alternating between HL and LL optimization does not guarantee an optimal joint policy for the original semi-MDP. In future work, we are interested in alternative theoretical frameworks to jointly optimize HL and LL policies for a single KL-regularized semi-MDP. Further, we plan to alleviate the restriction of fixed skill horizons with adaptive horizons and explore frameworks that differentiate between skill improvement and skill discovery.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[3] J. Li, C. Tang, M. Tomizuka, and W. Zhan, "Hierarchical planning through goal-conditioned offline reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 216–10 223, 2022.

[4] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conf. Robot Learning*. PMLR, 2021, pp. 188–204.

[5] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," in *Conference on Robot Learning*. PMLR, 2022, pp. 729–739.

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics Autonomous Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[7] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *preprint arXiv:2006.09359*, 2020.

[8] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," in *International Conference on Learning Representations*, 2019.

[11] S. Zhang and S. Whiteson, "Dac: The double actor-critic architecture for learning options," *Adv. Neural Inform. Process. Syst.*, vol. 32, 2019.

[12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*, 2018.

[13] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *Int. Conf. Learning Representations*, 2018.

[14] H. R. Walke, J. H. Yang, A. Yu, A. Kumar, J. Orbik, A. Singh, and S. Levine, "Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning," in *Conf. Robot Learning*. PMLR, 2023, pp. 1652–1662.

[15] R. Martin-Martin, A. Allshire, C. Lin, S. Mendes, S. Savarese, and A. Garg, "Laser: Learning a latent action space for efficient reinforcement learning," in *IEEE Int. Conf. Robotics Autom.*, 2021.

[16] M. Xu, M. Veloso, and S. Song, "ASPire: Adaptive skill priors for reinforcement learning," in *36th Conf. Neural Inform. Process. Syst.*, 2022.

[17] T. Nam, S.-H. Sun, K. Pertsch, S. J. Hwang, and J. J. Lim, "Skill-based meta-reinforcement learning," in *Int. Conf. Learning Representations*, 2021.

[18] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *Comput. Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[19] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," in *Conf. Robot Learning*. PMLR, 2023, pp. 302–325.

[20] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Pro. AAAI Conf. Artificial Intell.*, vol. 31, 2017.

[21] C. Li, X. Ma, C. Zhang, J. Yang, L. Xia, and Q. Zhao, "Soac: The soft option actor-critic architecture," *preprint arXiv:2006.14363*, 2020.

[22] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 5048–5058.

[23] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov, "Contrastive learning as goal-conditioned reinforcement learning," *Adv. Neural Inform. Process. Syst.*, vol. 35, pp. 35 603–35 620, 2022.

[24] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," *Adv. Neural Inform. Process. Syst.*, vol. 32, 2019.

[25] S. Pateria, B. Subagdja, A.-H. Tan, and C. Quek, "End-to-end hierarchical reinforcement learning with integrated subgoal discovery," *Trans. Neural Networks Learning Syst.*, 2021.

[26] C. Gao, Y. Jiang, and F. Chen, "Transferring hierarchical structures with dual meta imitation learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 762–773.

[27] K. Rana, M. Xu, B. Tidd, M. Milford, and N. Sünderhauf, "Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics," in *Conf. Robot Learning*. PMLR, 2023, pp. 2095–2104.

[28] J. Won, D. Gopinath, and J. Hodgins, "Physics-based character controllers using conditional vaes," *ACM Trans. Graphics*, vol. 41, no. 4, pp. 1–12, 2022.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Int. Conf. Machine Learning*. PMLR, 2018, pp. 1861–1870.

[30] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intell.*, vol. 112, no. 1-2, pp. 181–211, 1999.

[31] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *preprint arXiv:2004.07219*, 2020.

[32] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-human performance in gran turismo sport using deep reinforcement learning," *Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4257–4264, 2021.

[33] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *RSJ Int. Conf. Intell. Robots Syst.* IEEE, 2012, pp. 5026–5033.

[34] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," in *Int. Conf. Learning Representations*, 2018.

[35] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," in *Int. Conf. Learning Representations*, 2020.