Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Discovering Generalizable Skills via Automated Generation of Diverse Tasks

Kuan Fang[1], Yuke Zhu[2,3], Silvio Savarese[1], Li Fei-Fei[1]
[1] Stanford University, [2] UT Austin, [3] Nvidia

***Abstract*—The learning efficiency and generalization ability of an intelligent agent can be greatly improved by utilizing a useful set of skills. However, the design of robot skills can often be intractable in real-world applications due to the prohibitive amount of effort and expertise that it requires. In this work, we introduce Skill Learning In Diversified Environments (SLIDE), a method to discover generalizable skills via automated generation of a diverse set of tasks. As opposed to prior work on unsupervised discovery of skills which incentivizes the skills to produce different outcomes in the same environment, our method pairs each skill with a unique task produced by a trainable task generator. To encourage generalizable skills to emerge, our method trains each skill to specialize in the paired task and maximizes the diversity of the generated tasks. A task discriminator defined on the robot behaviors in the generated tasks is jointly trained to estimate the evidence lower bound of the diversity objective. The learned skills can then be composed in a hierarchical reinforcement learning algorithm to solve unseen target tasks. We demonstrate that the proposed method can effectively learn a variety of robot skills in two tabletop manipulation domains. Our results suggest that the learned skills can effectively improve the robot's performance in various unseen target tasks compared to existing reinforcement learning and skill learning methods.**

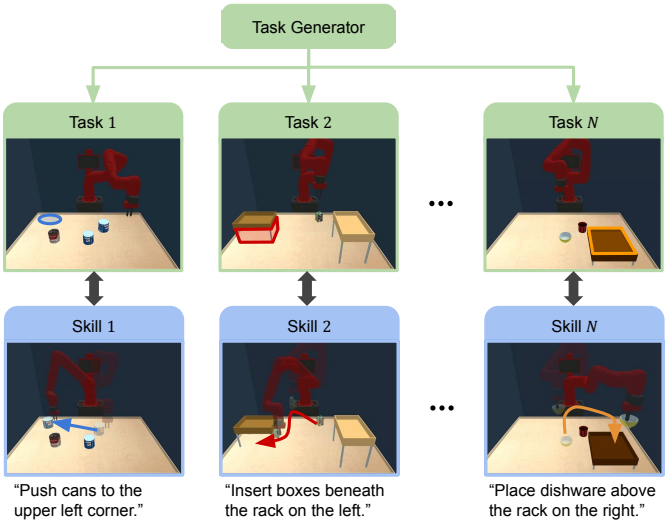

Fig. 1. The robot skills are learned through automated generation of tasks in our method. Each skill is paired with a unique task produced by the trainable task generator. The skills are trained to specialize in the paired tasks. We propose to discover a diverse set of skills by diversifying the generated tasks.

I. Introduction

The ability to acquire diverse and reusable skills is essential for intelligent agents to achieve generalizable autonomy. In well designed tasks, a variety of skills such as grasping [7, 43, 56], pushing [57, 30], and assembly [64, 89] can be crafted or learned by a robot. Given a repertoire of skills, previous work has enabled robots to strategically compose these skills to solve novel tasks by performing compositional planning [42, 73, 85, 80] and hierarchical reinforcement learning [4, 74]. However, the manual design of robot skills often involves a significant amount of time and human expertise. It renders handcrafting the sufficient set of skills intractable for the variability and complexity of real-world tasks.

To reduce the engineering burdens, a plethora of learning-based approaches have aimed to acquire generalizable robot skills from various sources of supervision. Given expert demonstrations or rewards in the target task as the supervision, these skills can be learned through hierarchical reinforcement learning [77, 18] or variational inference [63, 21]. Despite their successes, such supervision can be expensive to obtain and the skills trained to specialize in the target tasks might fall short in unseen tasks.

As an alternative, recent works [19, 32, 72] have proposed to learn skill-conditioned policies without supervision by incentivizing each skill to choose a sequence of actions that will lead to different outcomes. These methods have opened up a promising direction for unsupervised skill discovery. Nonetheless, most of these approaches gauge the diversity of the skills based on the next states they can reach in the environment, which does not capture the long-term semantics of the robot behaviors. Furthermore, while a set of skills are expected to serve diverse purposes in different scenarios, these methods focus on discovering the skills in a fixed environment. As a result, the learned skills are usually of limited versatility.

In this paper, we address the problem of learning generalizable skills by automatically generating a variety of tasks. Our key insight is: *a more effective way to acquire a diverse set of skills is to find a diverse set of tasks for training the skills*. In order to create tasks of rich variations, we resort to procedural content generation (PCG), which has been widely used for the automated creation of environments in physics simulations and video games [10, 68, 75]. Recent works have utilized PCG tools to create benchmarks for robot learning [75, 68, 10] and automated curricula for challenging tasks [65, 22]. To encourage generalizable skills to emerge, we would like to automate the generation of a diverse set of tasks.

To this end, we introduce Skill Learning In Diversified Environments (SLIDE), a method that discovers skills by utilizing procedurally generated tasks. In contrast to prior work which learns skills in a fixed environment [19, 32, 72], our method pairs each skill with a unique task. As shown

in Fig. 1, a skill-conditioned *task generator* is trained to create customized tasks for each skill. Instead of directly incentivizing the skills to produce different action distributions or reach distinguishable states, we propose to maximize the diversity of the generated tasks defined by an information-theoretic objective. A *task discriminator* defined on the agent's trajectories is designed to derive evidence lower bound of the objective. By jointly training the task generator, the task discriminator, and the skill policy, our method is able to learn skills of high inter-skill and intra-skill diversities. We further design a hierarchical reinforcement learning (HRL) policy that can efficiently learn to solve unseen target tasks by composing the discovered skills. At each time step, the HRL policy selects the suitable skill to perform various robot behaviors for completing the task goal.

Our experiments are conducted in two tabletop robotic manipulation domains in realistic simulation environments. Our method is able to learn diverse sets of tasks and skills without the notion of target tasks. Although no predefined task semantics are exerted, we found that the learned skills can often be interpreted as semantically meaningful interactions with different types of objects such as pushing, picking, placing, and inserting. Using the learned skills, the hierarchical policy in our method substantially outperforms reinforcement learning and skill learning baselines on unseen target tasks. Videos of generated tasks and learned skills are available at `kuanfang.github.io/slide/`

II. Related Work

A. Skill Learning

There is a large body of work on learning composable robot skills. In hierarchical reinforcement learning (HRL) approaches [15, 18, 76, 41, 4, 79, 28], the target tasks can be jointly solved by low-level policies that handle subroutines and high-level policies that learn to select the suitable low-level policies during different stages of the task execution. In these approaches, the skills can be represented by goal-conditioned policies [49, 16], options [77, 66, 2], and dynamic motion primitives [70, 38] which can be jointly trained with the high-level policies to solve the target task. In addition, pre-defined sub-goals [35], hand-designed features [26], or expert trajectories [12, 71] can be also used for learning the skills. [48, 3, 50] discovers skills by chaining and scheduling a sequence of policies. Given expert demonstrations, variational inference [46] can also be applied for learning skill-conditioned and goal-conditioned policies by encoding the trajectories to latent spaces [63, 21]. As opposed to these approaches, our method does not utilize any supervision from target tasks or expert demonstrations for learning the skills. Our method resonates with recent works on the unsupervised discovery of diverse behaviors which aim to learn diverse behaviors without the notion of target tasks. [32, 19] trains skill-conditioned policies to reach distinguishable states in the environment through mutual information maximization. Similarly, [72] simultaneously discovers predictable behaviors and learns their dynamics. [9, 29, 13, 78, 83, 53] encourage diverse behaviors to emerge by training the agent to maximize the entropy of the action distribution or the coverage of states in the environment. Inspired by the prior work, our method also defines the diversity of skills based on information theory. However, instead of diversifying the robot's behaviors in a fixed environment, our method learns to provide diverse tasks to encourage generalizable skills to emerge.

B. Skill Composition

Skills can be composed to enable the robot to perform long-term sophisticated interactions with the environment. Conventionally, hierarchical planning [24, 42, 73] integrates high-level planning of the sub-task order and the low-level planning of the motions. [61, 80] enable robots to compose multiple action modes and plan for motion trajectories that can solve sophisticated sequential problems using differentiable physics. Our method does not assume the physical dynamics of the environment to be known beforehand and relies on no predefined action modes. Recently, an increasing number of learning-based methods have been proposed to compose robot skills. In the robotic manipulation domain, several works enable robots to perform tool-use by learning the synergy between grasping and the subsequent manipulation actions [90, 20, 91] in a fixed order of execution. [6, 1, 17, 87, 14, 37, 52] composes predefined parameterized motion primitives by exploiting the compositional structure of long-horizon tasks and trains the high-level policy to decide the order and arguments of the primitives. [23, 55, 54, 51] propose to select or combine a set of sub-optimal policies to effectively learn to solve the target task. As opposed to these methods, our approach does not assume the skills are specified beforehand.

C. Procedural Task Generation

Procedural content generation (PCG) has been widely used for the automated creation of environments in physics simulations and video games [75, 68, 10]. Recently, PCG tools have been used to create benchmarks for robot learning and reinforcement learning [47, 86, 69, 88, 40]. Deign of these task environments can be labor intensive and heavily relies on human expertise. Without a carefully designed generation procedure, randomly sampled environments can often be infeasible to solve or trivially easy. Learning-based methods have been recently proposed to generate tasks for training robots and autonomous agents. In [31, 44, 8], diverse games and task environments are automatically created for training RL agents. The generated tasks can be used as automated curricula through parameterization of goals [27, 36, 67], initial states [84], and reward functions [33, 39]. [62] and [59] propose to actively adjust the hyperparameters in physical simulators to alleviate the domain shift by increasingly adding randomization to the physics of the environment. In [82, 81], an evolution strategy has been proposed to discover a set of environments by randomly mutating the existing environments in a simulated 2D game. [22] proposes a general framework that learns to generate new tasks of rich variations with configurable initial state probability, transition probability, and

reward function. Our method is similar in spirit to the prior work which aims to learn to procedurally generate tasks for training the robots. However, our method focuses on generating diverse tasks for skill learning. Instead of randomly mutating the tasks or maximizing an indicator of the learning progress of curriculum learning, we train the task generator by defining a diversity objective of the generated tasks.

III. Background

We consider each task as a Markov Decision Process (MDP) denoted by a tuple $M = (\mathcal{S}, \mathcal{A}, \rho, P, R, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, initial state probability ρ, transition probability P, reward function R, and discount factor γ. At each time step t, the policy $\pi(a|s)$ receives the current state s_t and selects an action $a_t \in \mathcal{A}$ to interact with the environment. A reward $r_t = R(s_t, a_t, s_{t+1})$ is received by the agent at each step as the feedback. To solve the task, the policy is trained to maximize the average cumulative reward $\mathbb{E}[\Sigma_t \gamma^t r_t]$.

Our formulation of skill learning follows the notations from [19, 72]. A latent variable $z \in \mathcal{Z}$ is used to represent the skill index, where $\mathcal{Z}$ can be a discrete or continuous space. A uniform skill prior $p(z)$ is used to sample z during training. Following the practice of [32, 19], we focus on the case of using discrete z due to consideration of learning robustness, while the proposed algorithm can be extended to continuous z with minor modifications. The skill policy $\pi_l(a|s, z; \theta_l)$ is defined to compute the action distribution conditioned on z, where θ_l is the trainable model weight.

Given the learned skill policy $\pi_l(a|s, z; \theta_l)$, we apply hierarchical reinforcement learning (HRL) [4] to solve the target task $\overline{M}$ by training a high-level policy $\pi_h(z|s; \theta_h)$ with trainable parameters θ_h. The high-level policy is trained to select the suitable skill index z at each time step by maximizing the cumulative reward in the target task. Since we use a discrete Z space in this work, π_h can be trained with the off-the-shelf Q-learning algorithm [60]. We jointly train the high-level policy and finetune the skill policies for each target task.

To conduct procedural task generation for skill learning, we consider a task space $\mathcal{T}$ which defines a finite or infinite number of MDPs of similar designs and properties as in [20]. We use a multi-dimensional parameter space $\mathcal{W}$ to represent the inter-task variation of $\mathcal{T}$. Given $w \in \mathcal{W}$, a task $M(w)$ can be instantiated in the task space by a predefined mapping $M(\cdot)$. We assume that all tasks share the same $\mathcal{S}$, $\mathcal{A}$ and γ such that all policies share the same input and output dimensions. More specifically, each $M(w)$ is defined by a distinct set of ρ, P, R parameterized by w. In general, the target task $\overline{M}$ can be either an instance of unknown parameter $\overline{w} \in \mathcal{W}$ or a task outside of $\mathcal{T}$ but shares the same $\mathcal{S}$ and $\mathcal{A}$. In this work, we consider $\overline{M}$ to be chosen from the parameterized task space.

IV. Skill Learning in Diversified Environments

We present Skill Learning In Diversified Environments (SLIDE), a method that utilizes procedural task generation for the discovery of generalizable skills. In contrast to prior work which trains the skills in a fixed environment, our method learns to automatically create diversified environments to train the skills. In this section, we first introduce our problem formulation of skill learning through the automated generation of tasks. We then explain our training pipeline and objective functions for learning diverse tasks and skills.

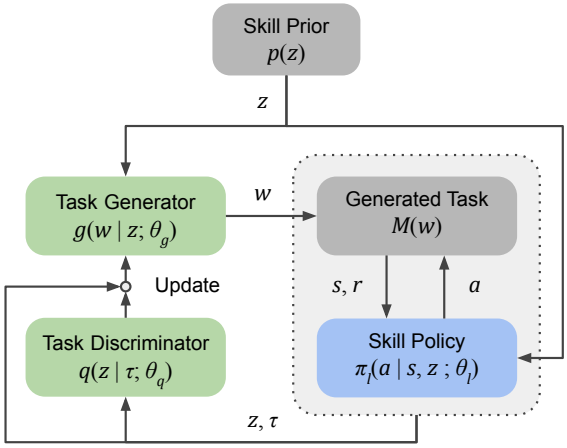

Fig. 2. Skill Learning In Diversified Environments (SLIDE). Conditioned on the sampled skill index z, the skill policy is trained in tasks procedurally created by the task generator. The task generator is trained to maximize an objective defined by the collected trajectory τ and the task discriminator. Task generators and discriminators are marked in green, the policy is marked in blue, and predefined modules are marked in grey.

A. Procedural Task Generation for Skill Learning

Our method integrates skill discovery and procedural task generation. Instead of learning all skills in the same environment, we pair each skill with a task for it to specialize. Our key insight is that a diverse set of skills will emerge by learning in a diverse set of tasks where each task requires unique robot behaviors to solve.

Given a parameterized task space $M(\cdot)$, a task can be represented by a task parameter w sampled from the task parameter space $\mathcal{W}$ as $M(w)$. However, each w often defines a specific configuration of the environment and the task goal. Training each skill for a single $M(w)$ can lead to limited generalization capability and a prohibitive amount of task-skill pairs need to be created to cover the useful robot behaviors in the task space. Alternatively, we can partition $\mathcal{W}$ into multiple modes and pair each skill with a mode of tasks. While a hard partition of the task space could be sub-optimal, we would like to extend the notion of using a task distribution $p(w|z)$ conditioned on the skill index z. Given the same z, $p(w|z)$ is supposed to sample w's that will lead to the same task semantics and require similar robot behaviors to solve.

To learn to generate suitable tasks for discovery of skills, we define a skill-conditioned task generator $g(w|z; \theta_g)$ to capture the task distribution $p(w|z)$, where θ_g are learnable model parameters. As opposed to prior work on curriculum learning and automated task generation [36, 58, 65, 67, 22] which create a sequence of goals or tasks as curricula, the task generator in our method aims to learn a set of task distributions for learning a diverse set of skills. The input

Algorithm 1 Skill Learning In Diversified Environments

Require: parameterized task space $M(\cdot)$ and skill prior $p(z)$

Initialize model parameters θ_g, θ_q, θ_l, α
while not converged **do**
  Sample skill $z \sim p(z)$
  Sample task parameter $w \sim g(w|z;\theta_g)$
  Create task $M(w)$ and initialize $s_1 \sim \rho(s_1|w)$
  Start a new trajectory $\tau = \varnothing$
  for $t = 1, ..., T$ **do**
    Sample action $a_t \sim \pi(a_t|s_t, z; \theta_l)$
    Step the environment $s_{t+1} \sim P(s_{t+1}|s_t, a_t, w)$
    Receive the reward $r_t = R(s_t, a_t, s_{t+1}, w)$
    Append the trajectory $\tau = \tau \cup \{(s_t, a_t, r_t, s_{t+1})\}$
    Update θ_l to maximize the cumulative reward
  end for
  Compute the inter-skill diversity using $\log q(z|\tau;\theta_q)$
  Update θ_q to minimize the classification error
  Update θ_g to maximize the objective in Eq. 3
  Update α by using Eq. 4
end while

z to our task generator is defined as the skill index which indicates the unique task semantics that the corresponding skill learns to solve, instead of being a random noise used in deep generative models. The task generator g can be defined for an arbitrarily complex task parameter space $\mathcal{W}$ and learns to model a categorical or Gaussian distribution for each discrete and continuous variable respectively. In this work, we implement g as a neural network that takes z as input and predicts the arguments (i.e. logits, means, standard deviations) of the probability distribution for each task parameter.

As shown in Fig. 2, the interplay between the task generator g and the skill policy π_l can be formulated by a teacher-student paradigm [58], in which the teacher g learns to provide suitable tasks for the student π_l to solve. During training, the policy learns to maximize the cumulative reward in the updated tasks created by the task generator, while the task generator adapts the task distributions based on the robot behaviors resulted from the current policy. The model parameters θ_g and θ_l are jointly optimized in this process.

B. Learning to Generate Diverse Tasks

The key to the skill discovery in our method is to design an objective function for training the skill-conditioned tasks generator g to create diverse tasks. To enable generalizable skills to emerge, we argue that both *inter-skill diversity* and *intra-skill diversity* need to be taken into considerations and properly balanced when training the task generator g. Inter-skill diversity encourages each task to present unique challenges for the paired skill to solve. While intra-skill diversity gauges the variations of environments that each task can provide. For robotic manipulation tasks, we would like each robot skill to specialize in a different type of interaction (e.g. pushing, grasping, placing, etc.) with a specific type of object. Meanwhile, we expect each skill to have sufficient

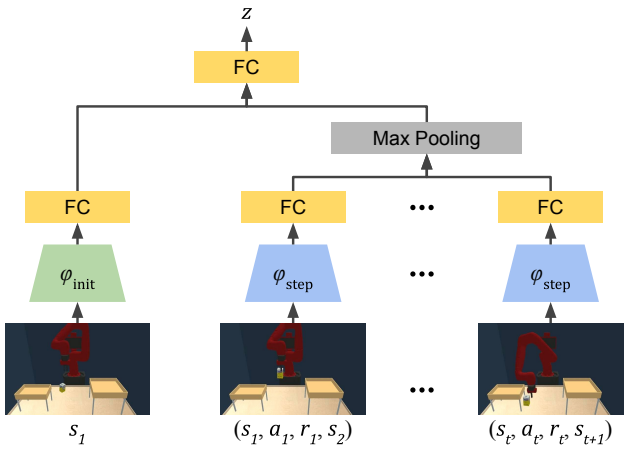

Fig. 3. Task discriminator network architecture. The network takes the trajectory collected by each skill as input. It encodes the initial state and each transition and predicts the skill index z from the pooled feature.

generalizability to handle the scene variability and task initialization. Lastly, we take the *feasibility* of the tasks into account to prevent learning skills in tasks that are infeasible to solve.

Inspired by prior work [19, 32], we define information-theoretic objectives to represent inter-skill and intra-skill diversity. However, instead of directly defining the diversities for the policy, our method aims to diversify the tasks created by the skill-conditioned task generator and the robot behaviors performed by the corresponding skill policy. To this end, we collect the robot's trajectory $\tau = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$ of T steps by sampling different skill index z and define the inter-skill diversity as the mutual information $I(\tau; z)$ between τ and z. The trajectory τ is jointly determined by g and π_l and provides information about the initialization, dynamics, and rewards of the task. When each skill is trained to specialize in its paired task, the τ unrolled by each skill captures the semantics of the task. By measuring how well we can infer about z given τ, this term represents the difference among robot behaviors resulted from choosing different z. To measure the intra-diversity, we use the conditional entropy $\mathcal{H}(w|z)$ of the task generator g. That is, we would like the task parameters associated with each z to be as diverse as possible. To measure the feasibility of task, we compute the average cumulative reward $\mathbb{E}_{p(\tau|z)}[\Sigma_t \gamma^t r_t]$ that the robot achieves by choosing the skill z, where $\mathbb{E}_{p(\tau|z)}[\cdot]$ is a shorthand notation that represents the expectation over distribution of τ conditioned on z. In summary, we train the task generator to maximize:

$$J = I(\tau; z) + \mathcal{H}(w|z) + \mathbb{E}_{p(\tau|z)}[\Sigma_t \gamma^t r_t]. \quad (1)$$

Since the true posterior $p(z|\tau)$ is unavailable, $I(\tau; z)$ cannot be evaluated directly. Instead, we introduce a task discriminator $q(z|\tau;\theta_q)$ as an approximation to $p(z|\tau)$. Note that in contrast with the approximate posterior used in [19] which only takes the next state as the input, q estimates the posterior distribution of z using the information of the full trajectory τ composed of the sequence of states, actions, and rewards. As shown in Fig. 3, the task discriminator first encodes the initial state and each time step using encoder networks ϕ_{init} and ϕ_{ste},

then merges the information across time steps to estimate the posterior of z. The task discriminator is trained to minimize the classification error given the ground truth z sampled during data collection. As a result, this allows us to solve the above problem by maximizing the variational lower bound of Eq. 1:

$$J = \mathbb{E}_{p(\tau|z)}[\log q(z|\tau)] + \mathcal{H}(w|z) + \mathbb{E}_{p(\tau|z)}[\Sigma_t \gamma^t r_t]. \quad (2)$$

The pseudocode of the SLIDE algorithm is summarized in Algorithm 1. During training, new tasks are continuously created and used for training the skills. In each new episode, a skill index z is sampled from $p(z)$ for the task generator g to create a new task instance $M(w)$. Then a trajectory τ is unrolled by the skill policy π_l. The training alternates among updates of the skill policy, the task discriminator, and the task generator using the collected trajectories.

C. Automating Diversity Adjustment

The inter-skill diversity and the intra-skill diversity need to be properly balanced to obtain generalizable skills. Hand-tuning a weight in the objective function or simply forcing the diversity to a fixed value would lead to poor solutions since the task generator should be free to explore the parameterized task space before the objective converges. Inspired by [34], we define $\bar{\mathcal{H}}$ as the target intra-skill diversity for each skill. We use the weight α in the objective to balance the terms. Specifically, we rewrite Eq. 2 to be:

$$J = \mathbb{E}_{p(\tau|z)}[\log q(z|\tau)] + \alpha\mathcal{H}(w|z) + \mathbb{E}_{p(\tau|z)}[\Sigma_t \gamma^t r_t]. \quad (3)$$

The weight α is constantly updated based on the difference between the evaluated intra-skill diversity $\mathcal{H}(w|z)$ and the chosen $\bar{\mathcal{H}}$. The optimal α^* can be solved as:

$$\alpha^* = \arg\min_{\alpha} \mathbb{E}_{z\sim p(z)}[\alpha\mathcal{H}(w|z) - \alpha\bar{\mathcal{H}}]. \quad (4)$$

The suitable $\bar{\mathcal{H}}$ depends on the task domain and is chosen to be 3 in our experiments.

V. Experiments

The goal of our experimental evaluation is to answer the following questions: 1) Can SLIDE discover diverse skills by learning to procedurally generate tasks? 2) Can the skills learned by SLIDE be utilized and generalized for learning to solve unseen tasks? 3) How do the design options in SLIDE affect the learned skills and the task performance?

A. Experiment Setup

To learn robot skills and evaluate their generalization capability to unseen target tasks, we design two tabletop manipulation domains. Each domain defines a task space that contains various tasks that share the same state and action spaces but different designs of environments and reward functions. The two task spaces are parameterized by multiple discrete and continuous variables to define the initialization, dynamics, and reward functions. We first train our method to discover skills by procedurally generating tasks from the parameterized task spaces without the notion of the target task. Then we train the hierarchical policy to solve each unseen target task by utilizing the skills learned from the same domain.

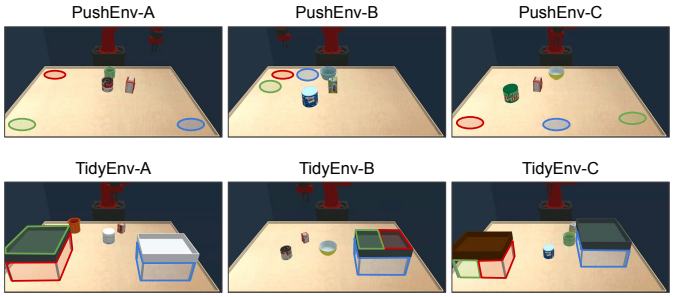

Fig. 4. Target tasks in the two domains. Goals of different object categories are indicated as: cans (red), boxes (green), and dishware (blue).

As shown in Fig. 4, our tasks involve a 7-DoF Sawyer robot arm interacting with multiple objects in a configurable table-top environment. The two task domains, which we name *PushEnv* and *TidyEnv*, are adapted from the task design from prior work on tabletop manipulation and object rearrangement [25, 21, 92, 5]. The task is simulated by a real-time physics engine [11] using the geometry and physical parameters that match a real-world Sawyer robot. In each episode, the environment is initialized with 1-3 movable objects randomly placed on the table. The objects are chosen from 3 categories including cans, boxes, and dishware. The total number, categories, initial placements, and attributes of objects can vary across episodes, so the robot needs to manipulate the objects accordingly. The task requires the robot to move the objects to initially unknown destinations. To complete the task, the robot is supposed to infer the task goals by interacting with the environment and maximize the reward by strategically rearranging the objects. The robot arm operates in a continuous action space through position control, which enables the robot arm to perform a variety of interactions with the objects in the tabletop environments including picking, placing, pushing, etc. Each episode terminates after 20 steps or when robots collide with the table. More specifically, we describe the parameterization, the state space, and the rewards of the two task domains below.

PushEnv. The robot aims to push objects of different categories towards the corresponding locations. In this task domain, the robot can move the end-effector arbitrarily in the constrained 3D space above the table while its fingers are fixed such that the objects cannot be directly picked and placed to the goals. To effectively complete the task, the robot is supposed to alternate between sliding over the table to push target objects towards the goals and lifting the arm to avoid undesired collisions with the distractor objects. The task is parameterized by a totally 15 independent variables which include the object category, object size, goal size, and 2D goal location corresponding to each object category. The robot observes the gripper position as well as the object positions and physical properties but does not know the goal location. The action is defined as a 3-dimensional vector representing the target position that the end-effector is going to be moved

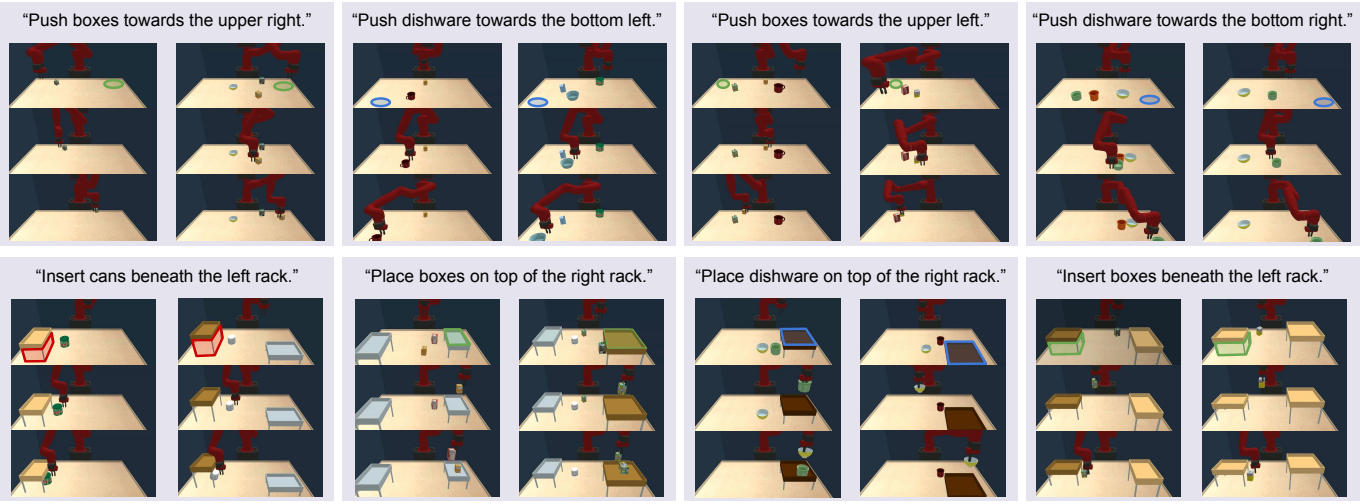

Fig. 5. Example tasks and skills discovered by SLIDE. We demonstrate the inter-skill and intra-skill diversity by showing two sampled trajectories associated with the same skill index in each grey block. Each column shows the initialization of the generated task (the top tile) and the execution of the skill (the second and the third tiles). Different colors indicate destinations of different object categories including cans (red), boxes (green), and dishware (blue).

to. The reward is defined as the displacement of the object towards the goal and can be either positive or negative.

TidyEnv. The robot is asked to tidy up the table by rearranging objects to specific destinations above or beneath the racks. In addition to moving the end-effector, the robot is enabled to open or close its fingers to pick or place objects. Up to two static racks are attached to the table. Each rack can have varying shapes, heights, and locations across episodes. The objects of each category may have a destination as the upper or lower level of one of the racks or remain on the table surface. To move the objects to the destinations, the robot needs to strategically pick, place, and insert the objects. The task is parameterized by 12 independent variables. Aside from the aforementioned object properties, the parameter includes categorical variables presenting the destination of each object category as well as the height of the rack. The robot observes the gripper position, the finger status, and the object properties. The action is defined as a 4-dimensional vector representing the target end-effector position as well as an additional value chosen between 0 and 1 to indicate whether the robot fingers are closed. The robot receives a sparse reward of 1 when an object is moved to the correct destination.

Implementation Details. Soft-Actor-Critic (SAC) [34] is applied to train the skill policies in the generated tasks. The actor networks and critic networks for SAC and the Q-network for deep Q-learning are implemented with an object-centric feature extractor adapted from [21]. The task discriminator network is adapted from [22] which first separately encodes each modality of the trajectory and then concatenates the encoded features for each time step in order to eventually predict the skill index z for the input trajectory τ. 64-dimensional fully-connected (FC) layers are used in the network architectures and 64 skills are learned in SLIDE. For all experiments, we use the ADAM optimizer [45] with learning rate of 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and the batch size of 128. Hyperparameters are chosen by random search.

B. Analysis of Generated Tasks and Learned Skills

We train SLIDE for 500K iterations and visualize the discovered tasks and skills in Fig. 5. Each grey block shows two example trajectories associated with a different skill index z in one of the two task domains. The initialization of the generated task (the top tile) and the execution of the skill policy (the second and the third tiles) are demonstrated in each column. Marks of different colors are used to indicate destinations for different object categories including cans (red), boxes (green), and dishware (blue). As shown in the figure, the robot learns to perform a variety of behaviors. Although the task semantics are not predefined in our model, we found that the learned skills can often be interpreted as semantically meaningful interactions with different types of objects such as pushing, picking, placing, and inserting. The task generator usually learns to generate a concentrated distribution of object types and goal locations for the skill to focus on. Given generated tasks of diverging semantics, we found that a single skill suffers to make consistent progress and the task discriminator can hardly identify the skill index based on the resultant trajectories. As a result, the inter-task diversity and the feasibility term often make the distribution of task parameters that are critical to the task semantic quickly converge to a single mode. Meanwhile, the intra-skill diversity encourages the task parameters that have a smaller effect on the robot behaviors to diverge as much as possible. The task parameters often need to co-adapt to create meaningful tasks. For instance, the rack heights are required to be higher than a threshold to allow objects to be inserted beneath. Their distributions become more uniform when the task is about placing the objects on top of the rack.

C. Quantitative Results in Target Tasks

We evaluate the robot's performance in the target tasks using different methods. Our method and baselines that learn skills or generate tasks are first trained for 500K iterations without

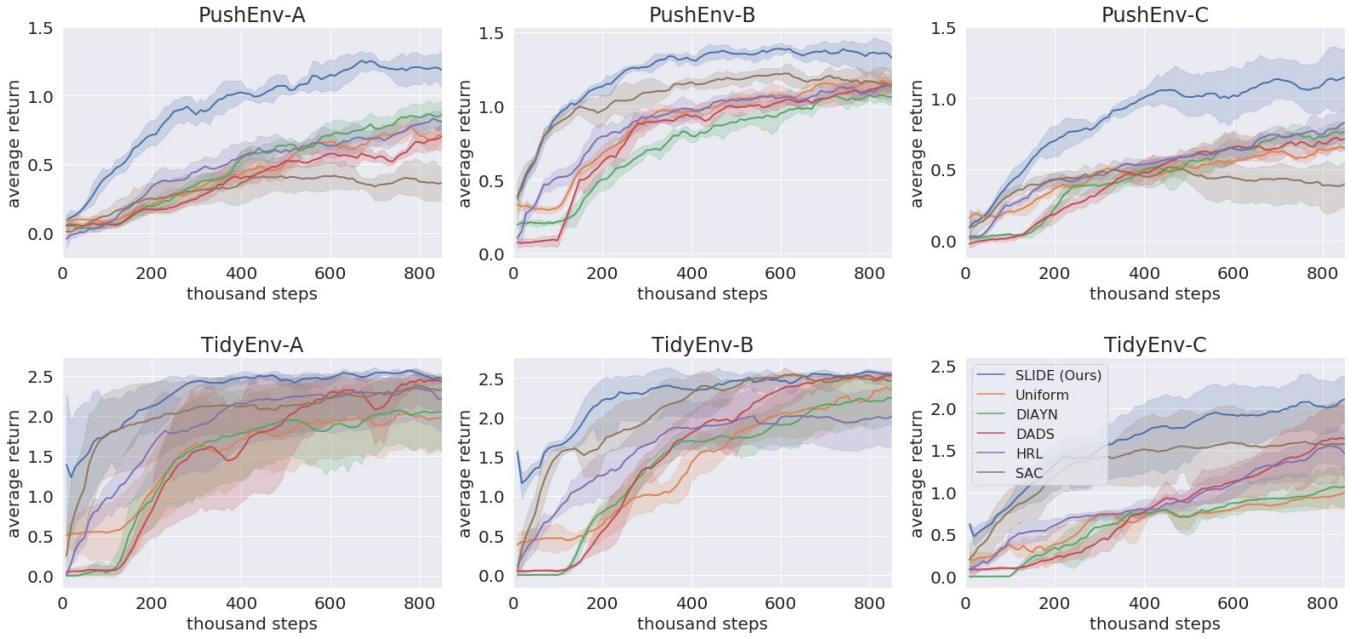

Fig. 6. Quantitative results in the target tasks. Given the skills learned by SLIDE, we train the hierarchical policy to solve unseen target tasks. Our method outperforms multiple reinforcement learning and skill learning baselines in terms of the average return.

the notion of the target tasks. Then each method is trained and evaluated in the target tasks for 800k iterations. During training, we evaluate the average return of the trained model for 50 episodes every 10k iterations.

Baselines. We compare SLIDE with multiple baseline methods which include two model-free RL algorithms trained from scratch, two skill learning algorithms, and a task generation baseline. **SAC** [34] is the state-of-the-art model-free RL baseline for continuous action spaces that are directly trained in the target task. It uses similar actor network and critic network as in our method except that the outputs are not conditioned on the skill index. **HRL** [4] trains the same hierarchical policy described in Sec. III except that the skill policy is jointly trained from scratch in the target tasks. **DIAYN** [19] is the state-of-the-art method of unsupervised discovery of skills that maximizes an information-theoretic objective directly on the skill policy. Similarly, **DADS** [72] aims to discover predictable behaviors by maximizing the diversity of the skills defined by a jointly learned dynamics model. We train the two skill learning baselines in randomly initialized environments sampled from the parameterized task space. We also include a task generation baseline **Uniform** which uniformly samples tasks from the parameterized task space without using a learned task generator. To have a fair comparison, we use the same network architecture for the corresponding components (e.g.the high-level policy and the skill policy) and search for the optimal hyperparameters for each method.

Comparative analysis. We evaluate and analyze the robot's performance in the target tasks using different methods. The evaluation across 5 runs is shown in Fig. 6 where the curves indicate the average return and the shades indicate the standard deviations. Our method consistently outperforms the baseline methods in all target tasks in terms of the learning efficiency and the average return at convergence. The skills learned by SLIDE successfully enable the robot to solve the unseen target tasks with a high learning efficiency. The hierarchical policy effectively composes and finetunes the learned skills to interacts with different types of objects by performing various behaviors. As a comparison, the SAC baseline without using learned skills often learns quickly at the beginning of training since it has a simpler policy network to train and only focuses on a single mode of robot behavior. However, it fails to discover and utilize diverse behaviors that are required for completing the target task at a later stage. While SAC can achieve a reasonably good performance in simpler tasks that require limited modes of behavior (e.g.PushEnv-B and TidyEnv-B), the performance gap between our method and SAC is much larger in more challenging tasks. The HRL baseline trained from scratch enables the robot to utilize multiple modes of behaviors. Nevertheless, HRL cannot effectively explore the environment using randomly initialized skills. The two skill learning baselines, DIAYN and DADS, can hardly to discover semantically meaningful skills in these two task domains. Instead of discovering robot behaviors such as pushing and grasping, these baselines tend to learn to move the robot gripper to different 3D locations without effective interactions with the objects on the table, which are sufficient for satisfying their diversity metrics defined on the next state. As a result, they achieve similar or even worse performance than HRL since the sub-optimal skills can lead to poor exploration in the target tasks. By randomly sampling from the task space, the Uniform baseline often struggles to provide suitable tasks to learn generalizable skills and as a result it cannot efficiently learn to solve the target tasks.

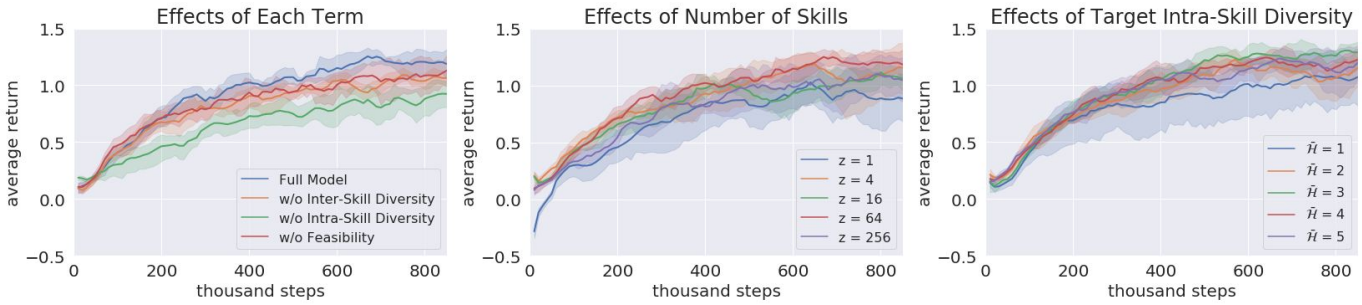

Fig. 7. Ablation study. We investigate the effects of each term in the object, the number of skills, and the target intra-skill diversity. The training and evaluation of each ablation follow the same protocol as in the quantitative results and the average returns are plotted.

D. Ablation Study

We run three ablation studies to investigate the importance of each component and hyperparameter to the task performance in the target tasks. Following the same training and evaluation protocols as in Sec. V-C, we run each ablation to learn skills in for PushEnv and then train the hierarchical policy with the obtained skills to solve the target task PushEnv-A. The resultant average return of each ablation across 5 runs is summarized in Fig. 7.

Effects of each term. To understand the importance of each term in the objective function, we train our method by removing one of the terms in the objective function (Eq. 3) in each ablation. As a result, the average return in the target task at convergence is reduced by 0.1 - 0.3 as shown in Fig. 7. Removing the intra-skill diversity leads to the largest performance regression since each generated task would collapse into a concentrated task distribution which causes the corresponding skill to overfit to a very specific scenario. Without considering the inter-skill diversity, the generated tasks associated with different skill indices tend to overlap with each other and therefore the learned skills would have poor coverage of the robot behaviors needed for solving the target task. Nevertheless, the task distribution of each skill sometimes can still become unique if they happen to be differently initialized. Since the feasibility term encourages each generated task to be solvable by the paired skill, the task distribution would still capture tasks of similar semantics at convergence. Without the feasibility term, we found that less meaningful tasks can be created by the task generator and the high-level policy is thus overwhelmed by useless skills when learning to solve the unseen target tasks.

Effects of number of skills. While each task domain can inherently contain a specific set of behavior modes, we do not assume that we know how many skills can be learned beforehand. Instead, we define the number of skills to be a hyperparameter of the method. As shown in Fig. 7, we run our method by sweeping the number of skills from 1 to 256. On one hand, learning only a single skill leads to very limited modes of robot behaviors during the discovery of skills. As a consequence, the robot struggles to effectively explore the environment in the target task at the beginning and the performance at convergence is sub-optimal. On the other hand, setting the number too large can make the learned tasks and skills highly redundant, which confuses the model in the target task. We found the optimal number of tasks and skills is around 64 in the chosen task domain.

Effects of the target intra-skill diversity. We study the effects on the target intra-skill diversity $\bar{\mathcal{H}}$ which is defined as a hyperparameter in our method. $\bar{\mathcal{H}}$ controls the balance between the intra-skill diversity and other terms in the objective. In Fig. 7, we report the performance by sweeping $\bar{\mathcal{H}}$ from 1 to 5. We found that using too high or too low values of $\bar{\mathcal{H}}$ can cause either the inter-skill diversity or the intra-skill diversity to outweigh each other. In our experiments, the $\bar{\mathcal{H}}$ leads to the best performance is around 3.

VI. Conclusion and Discussion

We present our method, Skill Learning In Diversified Environments (SLIDE), which learns generalizable skills by the automated generation of a diverse set of tasks. By maximizing the diversity of the generated tasks, our method is able to discover a variety of tasks to enable skill policies performing diverse robot behaviors to emerge. By training a hierarchical reinforcement learning algorithm that utilizes the learned skills as low-level policies, our method effectively improves the performance and the learning efficiency in unseen target tasks from two tabletop manipulation domains.

Several aspects of the proposed method can be further investigated in future work. First, while the proposed method is designed to learn a fixed number of skills, an exciting direction would be conducting open-ended discovery of tasks and skills with a flexible number of skills. Second, we assume the parameterized reward function is predefined in the task space which suggests task goals that are potentially useful in target tasks, but future work could instead generate tasks based on intrinsically motivated reward functions. Lastly, we hope this work could encourage more endeavors in utilizing procedural content generation for robot learning and similar approaches can be proposed for a broader scope of applications such as visual navigation and humanoid robots.

Acknowledgement: We acknowledge the support of Toyota (1186781-31-UDARO) and HAI-AWS cloud credits. We would like to thank Ademi Adeniji, Ajay Mandlekar, and Eric Li for their constructive feedback.

REFERENCES

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 166–175. JMLR. org, 2017.

[2] P. Bacon, J. Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

[3] A. Bagaria and G. Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations (ICLR)*, 2020.

[4] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

[5] Dhruv Batra, A. X. Chang, S. Chernova, A. Davison, Jun Deng, V. Koltun, S. Levine, J. Malik, Igor Mordatch, R. Mottaghi, M. Savva, and Hao Su. Rearrangement: A challenge for embodied ai. *ArXiv*, abs/2011.01975, 2020.

[6] Darrin C Bentivegna and Christopher G Atkeson. Learning from observation using primitives. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1988–1993. IEEE, 2001.

[7] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis – a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[8] Philip Bontrager and J. Togelius. Fully differentiable procedural content generation through generative playing networks. *ArXiv*, abs/2002.05259, 2020.

[9] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

[10] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *ArXiv*, abs/1912.01588, 2019.

[11] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[12] C. Daniel, G. Neumann, Oliver Kroemer, and Jan Peters. Learning sequential motor tasks. *2013 IEEE International Conference on Robotics and Automation*, pages 2626–2632, 2013.

[13] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281. PMLR, 2012.

[14] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical search: Multi-step retrieval of a target object occluded by clutter. *arXiv preprint arXiv:1903.01588*, 2019.

[15] P. Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, 1992.

[16] M. Deisenroth, P. Englert, Jan Peters, and D. Fox. Multitask policy search for robotics. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3876–3881, 2014.

[17] Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. Programmable agents. *arXiv preprint arXiv:1706.06383*, 2017.

[18] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[19] Benjamin Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *International Conference on Representation Learning*, 2019.

[20] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kuryenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *Robotics: Science and Systems (RSS)*, 2018.

[21] Kuan Fang, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. *Conference on Robot Learning (CoRL)*, 2019.

[22] Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Adaptive procedural task generation for hard-exploration problems. *arXiv preprint arXiv:2007.00350*, 2020.

[23] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

[24] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[25] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[26] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[27] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *ArXiv*, abs/1708.02190, 2017.

[28] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

[29] Tanmay Gangwani, Qiang Liu, and Jian Peng. Learning self-imitating diverse policies. *arXiv preprint arXiv:1805.10309*, 2018.

[30] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[31] Daniele Gravina, A. Khalifa, Antonios Liapis, J. Togelius, and Georgios N. Yannakakis. Procedural content generation through quality diversity. *IEEE Conference on Games (CoG*, 2019.

[32] K. Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *International Conference on Representation Learning*, 2017.

[33] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *ArXiv*, abs/1806.04640, 2018.

[34] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[35] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[36] David Held, Xinyang Geng, Carlos Florensa, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, 2018.

[37] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8565–8574, 2019.

[38] A. Ijspeert, Jun Nakanishi, Heiko Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 2013.

[39] Allan Jabri, Kyle Hsu, Ben Eysenbach, Abhishek Gupta, Alexei A. Efros, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. *ArXiv*, abs/1912.04226, 2019.

[40] Stephen W. James, Z. Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5:3019–3026, 2020.

[41] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

[42] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[43] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *IEEE International Conference on Robotics and Automation*, pages 4304–4311. IEEE, 2015.

[44] A. Khalifa, Philip Bontrager, Sam Earle, and J. Togelius. Pcgrl: Procedural content generation via reinforcement learning. *ArXiv*, abs/2001.09212, 2020.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[47] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[48] George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22:1015–1023, 2009.

[49] Tejas D. Kulkarni, Karthik Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

[50] V. Kumar, Sehoon Ha, and C. Liu. Expanding motor skills using relay networks. In *CoRL*, 2018.

[51] A. Kurenkov, Ajay Mandlekar, R. Martin, S. Savarese, and Animesh Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. In *Conference on Robot Learning*, 2019.

[52] A. Kurenkov, Joseph Taglic, R. Kulkarni, Marcus Dominguez-Kuhne, Animesh Garg, Roberto Martín-Martín, and S. Savarese. Visuomotor mechanical search: Learning to retrieve target objects in clutter. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8408–8414, 2020.

[53] Youngwoon Lee, Jingyun Yang, and Joseph J Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International Conference on Learning Representations*, 2019.

[54] S. Li, Fangda Gu, Guangxiang Zhu, and C. Zhang. Context-aware policy reuse. In *AAMAS*, 2019.

[55] Siyuan Li and Chongjie Zhang. An optimal online method of selecting source policies for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[56] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[57] M. T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5:53 – 71, 1986.

[58] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.

[59] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher Joseph Pal, and Liam Paull. Active domain randomization. *ArXiv*, abs/1904.04762, 2019.

[60] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex

Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[61] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[62] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jadwiga Tworek, Peter Welinder, Lilian Weng, Qi-Ming Yuan, Wojciech Zaremba, and Lefei Zhang. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019.

[63] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. *Conference on Robot Learning (CoRL)*, abs/2010.11944, 2020.

[64] I. Popov, N. Heess, T. Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerík, T. Lampe, Y. Tassa, T. Erez, and Martin A. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *ArXiv*, abs/1704.03073, 2017.

[65] R'emy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. *ArXiv*, abs/1910.07224, 2019.

[66] Doina Precup and R. Sutton. Temporal abstraction in reinforcement learning. In *ICML 2000*, 2000.

[67] Sébastien Racanière, Andrew Kyle Lampinen, Adam Santoro, David P. Reichert, Vlad Firoiu, and Timothy P. Lillicrap. Automated curriculum generation through setter-solver interactions. In *ICLR*, 2020.

[68] Sebastian Risi and Julian Togelius. Procedural content generation: From automatically generating game levels to increasing generality in machine learning. *arXiv preprint arXiv:1911.13071*, 2019.

[69] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9339–9347, 2019.

[70] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[71] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[72] Archit Sharma, Shixiang Gu, S. Levine, V. Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *International Conference on Representation Learning*, 2020.

[73] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.

[74] Freek Stulp, Evangelos A Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on robotics*, 28(6):1360–1370, 2012.

[75] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

[76] R. Sutton and A. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.

[77] R. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112:181–211, 1999.

[78] Valentin Thomas, Emmanuel Bengio, William Fedus, Jules Pondard, Philippe Beaudoin, Hugo Larochelle, Joelle Pineau, Doina Precup, and Yoshua Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018.

[79] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems*, 1994.

[80] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. *Robotics: Science and Systems (RSS)*, 2018.

[81] Renmin Wang, J. Lehman, A. Rawal, Jiale Zhi, Yulun Li, J. Clune, and K. Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. *ArXiv*, abs/2003.08536, 2020.

[82] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Poet: open-ended coevolution of environments and their optimized solutions. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[83] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[84] Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1503–1511, 2020.

[85] Jason Wolfe, Bhaskara Marthi, and Stuart Russell. Combined task and motion planning for mobile manipulation. In *Proceedings of the International Conference on Auto-*

mated Planning and Scheduling, 2010.

[86] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[87] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[88] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.

[89] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.

[90] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[91] Andy Zeng, Shu-Ran Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36:1307–1319, 2020.

[92] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *arXiv preprint arXiv:2012.07277*, 2020.