Learning Parameterized Skills from Demonstration Data

Vedant Gupta [1]

Abstract

We present an end-to-end algorithm for discovering *parameterized skills* from demonstration data. Our method jointly learns parameterized skill policies and a meta-policy that selects the appropriate skill and parameters at each timestep. Using a combination of temporal variational inference and information-theoretic regularization methods, we address the challenge of degeneracy common in latent variable models, ensuring that the learned skills are temporally extended, semantically meaningful, and adaptable. We evaluate our approach on LIBERO, a challenging multi-task benchmark for robot manipulation with multi-modal inputs. Quantitatively, our method achieves higher success rates and significantly greater consistency (i.e. lower coefficient of variance) across unseen tasks and random seeds. Qualitatively, we demonstrate the emergence of interpretable parameterized skills, such as grasp object skill whose continuous arguments define the grasp location.

1. Introduction

Applying Reinforcement Learning (Sutton & Barto, 2018) to long-horizon sequential decision making problems can be very sample inefficient. For example, training a single task-specific policy to play an Atari game can require over 10 million samples (Mnih et al., 2015; Hessel et al., 2018), while humans can learn to play effectively after only 20 episodes. This gap is largely due to humans' ability to learn and reuse *strong priors* from past experience, an ability that remains difficult to replicate in RL agents. Zhang et at. (2024) show that basic supervised pretraining of policies can perform *worse* on unseen tasks than a policy trained from scratch, highlighting the need for better methods to learn generalizable priors.

A promising direction to acquire such priors is to learn hierarchically structured policies, often formalized using the options framework (Sutton et al., 1999). The motivating assumption for this approach is that real-world tasks can be decomposed into sequences of recurring subproblems. Hence, learning modular and temporally extended skills or options can reduce the effective planning horizon and improve generalization to new tasks.

Our goal is to discover reusable skills from a dataset of expert demonstrations (i.e. trajectories) of an agent performing various tasks, and use these skills to solve new tasks efficiently. Previous work towards this objective includes work by Fox et al. (2017), Krishnan et al. (2017), Kipf et al. (2019), Sharma et al. (2019), Shankar and Gupta (2020), and Zheng et al. (2024). However, these works focus on learning purely discrete or continuous skills. We posit that learning *parameterized skills*, i.e. skills that are discrete in nature but can be modulated by continuous arguments, will lead to better representations and adaptation to new tasks. Discrete skills alone lack the flexibility needed for real-world variation, while continuous skills can be less structured and harder to interpret.

For example, one may learn a discrete skill to slice fruit with a knife. However, in practice, no two fruits are identical, and one may need to apply this skill in various kitchen environments using different knives/equipment. Learning a different discrete skill for every possible instantiation is evidently intractable. On the other hand, learning a parameterized skill `slice_fruit(x,y,z)`, where the continuous parameters *refine* the slicing skill to the relevant instance, compactly represents a family of policies corresponding to slicing fruit. This enables strong generalization to previously unseen pickup locations.

Recent work on learning parameterized skills includes LOTUS (Wan et al., 2024) and EXTRACT (Zhang et al., 2024), which first identify discrete skills by clustering demonstration trajectories using pretrained models such as VLMs. There are two downsides to this approach: (i) it implicitly assumes that a given skill will occur in visually similar environments, (ii) fixing skills beforehand could make these methods more brittle if the initial clustering happens to not be appropriate.

We instead propose an *end-to-end method* to learn parameterized skills, jointly training (i) a discrete skill selector (ii)

[1]Department of Computer Science, Brown University, Providence, USA. Correspondence to: Vedant Gupta <vedant_gupta@brown.edu>.

a continuous parameter selector conditioned on the discrete skill (iii) a subpolicy conditioned on both. We adopt an approach that is directionally similar to that used by Shankar and Gupta (2020), who introduce *temporal variational inference* to train a latent variable model using a maximum likelihood objective, where the latent variables represent discrete/continuous skills.

However, this approach is *under-specified*, and often admits *degenerate solutions* that maximize the behavior cloning loss without learning useful skill abstractions (Jiang et al., 2022). This issue of under-specification is exacerbated when trying to learn parameterized skills using a latent variable model and behavior cloning loss. At one extreme, the model may reconstruct the trajectories in the training data by assigning a unique continuous parameter at every timestep, effectively using the continuous parameter to encode the desired action directly and without any temporal extension. Perhaps more interestingly, we find it is also common for models to learn to encode all trajectories in a single skill and continuous parameter. As we show in Appendix A, this behavior is enabled by the fact that expert demonstrations for different tasks can occupy disjoint regions of the observation space. In such cases, a single skill can learn to simultaneously complete different tasks in different regions of the observation space, bypassing the need to learn meaningful parameterized skills to minimize the behavior cloning loss.

To prevent the occurrence of such degenerate solutions, we employ various information-theoretic constraints and architectural choices, as outlined in Section 3.5. This includes *compressing* the observation embedding before passing it to subpolicy networks to reduce the amount of information and predicting continuous parameters *for each discrete skill* rather than at every timestep. These design choices force the model to rely on the latent variables to solve the task, resulting in more robust, interpretable, and generalizable skills.

2. Related Work

2.1. Learning Skills from Demonstrations

A straightforward approach to learning from multitask demonstration data is to train a monolithic policy that maps a state and task label to a single action. However, previous works suggest that the difficulty of a sequential decision-making problem scales with the problem horizon (Ross et al., 2011; Jiang et al., 2015; Rajaraman et al., 2020). To improve sample efficiency and generalization, many methods hence focus on learning temporally-extended action abstractions called options/skills (Sutton et al., 1999).

There is a large body of work focusing on learning skills from demonstrations (Konidaris et al., 2012; Niekum et al., 2013; Krishnan et al., 2017; Kipf et al., 2019; Sharma et al., 2019; Shankar & Gupta, 2020; Zheng et al., 2024; Fu et al., 2024). Notably, Shankar and Gupta (2020) introduce temporal variational inference to learn skills from demonstrations. However, their method is restricted to low-dimensional state spaces. Furthermore, all the methods above are restricted to learning a fixed number of discrete skills or continuous skills.

2.2. Learning Parameterized Skills

As outlined in Section 1, discrete skills can lack the representational capacity to generalize across variations of a task, and while continuous skills offer fine-grained control, they can be harder to utilize and interpret. Parameterized skills, which combine discrete skills with continuous parameters for refinement, offer a promising alternative that blends the advantages of both discrete and continuous skills.

Early work on parameterized skills, including work by Da Silva et al (2012), proposed the construction of parameterized skills by analyzing the structure of policy manifolds. However, this required labeled parameters of tasks for training. More recent methods, such as LOTUS (Wan et al., 2024) and EXTRACT (Zhang et al., 2024), learn parameterized skills by first clustering demonstration trajectories into different discrete skills using pretrained models such as VLMs, and then learning parameterized policies corresponding to each cluster. However, this approach implicitly makes the assumption that the same discrete skill takes place in visually similar environments, which may not hold in practice. Furthermore, fixing the discrete skill set in advance may make the methods brittle when the initial clustering does not align well with the inherent task structure.

In contrast, our method learns both the discrete and continuous components of parameterized skills, along with the subpolicy, in an end-to-end fashion. This allows the model to jointly optimize all components for generalization and robustness. In Section 4, we demonstrate that our method achieves significantly more consistent performance across unseen tasks and random seeds. We attribute this to the expressiveness of parameterized skills as an underlying representation, as well as to the benefits of joint end-to-end optimization.

3. Learning Parameterized Skills from Demonstration Data

We consider a set of tasks $\mathcal{T}$, where each task $l \in \mathcal{T}$ corresponds to a Markov Decision Process $\{S, A, P, G_t\}$, where S is the (potentially multimodal) state space, A is the action space, P is the transition function between successive states (i.e. $P(s_{t+1}|s_t, a_t)$), and G_t is the set of goal states wherein task t will be considered completed. Note that in

this formulation, all tasks are assumed to share the same state/action spaces and transition function.

In the Learning from Demonstrations (LfD) setting, we assume we are provided with a training dataset containing N expert trajectories $D = \{T_i\}_{i=1}^{N}$. Each trajectory $T_i = \{\tau_i, l_i\}$, where $l_i \in \mathcal{T}$ denotes the task the trajectory corresponds to, while $\tau_i = \{s_t, a_t\}_{t=1}^{K_i}$, $s_t \in S$, $a_t \in A$, denotes the observation-action pairs recorded in the trajectory.

Our overarching goal is to learn a task-conditioned policy that not only generalizes well within individual task covered within the training data $\{T_i\}_{i=1}^{N}$, but also learns from the shared structure across these tasks to better generalize to novel tasks drawn from $\mathcal{T}$.

3.1. Background

We assume that tasks in $\mathcal{T}$, and more generally, tasks in the real world, naturally decompose into sequences of discrete skills coming from some set K. For example, a trajectory of a robot successfully making tea involves picking and placing different objects, grasping and turning knobs, etc. Furthermore, we assume that each discrete skill $k \in K$ can be parameterized by a vector $z \in \mathbb{R}^d$ [1], to yield a policy $\pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t)$. Different discrete skills in K are meant to represent fundamentally different policies. Given a discrete skill $k \in K$, the policy $\pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t)$ is assumed to vary smoothly with respect to the continuous parameter $z \in \mathbb{R}^d$. For example, a discrete skill to pick up objects, `pick(x,y,z)`, might be parameterized by a continuous parameter in $\mathbb{R}^3$ representing the location to pick the object up from.

3.2. Generative Process

To choose an action at timestep t using parameterized skills when executing task $l \in T$, we assume that first a discrete skill k_t is sampled from the *discrete-policy*, $\pi^K(k_t|s_{1:t}, a_{1:t-1}, k_{1:t-1}, z_{1:t-1}, l)$. Next, a continuous parameter z_t is sampled from the *continuous-policy* corresponding to k_t, $\pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_{1:t}, z_{1:t-1}, l)$. Finally, an action is sampled from the *subpolicy* corresponding to (k_t, z_t), $\pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t)$[2].

Consider some trajectory $(\tau, l) \in D$, where $\tau = \{s_t, a_t\}_{t=1}^{K}$, and some corresponding sequence of discrete skills $\kappa = \{k_i\}_{i=1}^{K}$ and continuous parameters $\zeta = \{z_i\}_{i=1}^{K}$. Then, we can write the joint likelihood of these sequences $P(\tau, \kappa, \zeta, l)$ as follows:

$$
\begin{aligned}
P(\tau, \kappa, \zeta, l) = P(s_1, l) \prod_{t=1}^{K} \{ & \pi^K(k_t|s_{1:t}, a_{1:t-1}, k_{1:t-1}, z_{1:t-1}, l) \\
& \pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_{1:t}, z_{1:t-1}, l) \\
& \pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t) \\
& P(s_{t+1}|s_t, a_t)\}. \qquad (1)
\end{aligned}
$$

3.3. Deriving a Training Objective

To train policies π^K, π^Z, and π_O that are ultimately usable at inference time, each policy can only be conditioned on prior observations, actions, discrete skills, and continuous parameters. In line with prior literature, we aim to maximize the objective $\mathbb{E}_{\tau,l\sim D}(\log P(\tau, l))$. However, calculating $\log P(\tau, l)$ exactly involves an intractable marginalization over all possible sequences of discrete and continuous skills. We use temporal variational inference (Shankar & Gupta, 2020) to estimate $\mathbb{E}_{\tau,l\sim D}(\log P(\tau, l))$ while maintaining the auto-regressive structure of the policies learned. We introduce a *variational distribution* $q(\kappa, \zeta|\tau, l)$ which is meant to approximate $P(\kappa, \zeta|\tau, l)$. Due to the non-negativity of KL divergence:

$$
\begin{aligned}
\mathbb{E}_{\tau,l\sim D}(\log P(\tau, l)) \geq \mathbb{E}_{\tau,l\sim D}(\log P(\tau, l) \\
- D_{KL}(q(\kappa, \zeta|\tau, l)||P(\kappa, \zeta \mid \tau, l))) \coloneqq \mathcal{L}.
\end{aligned}
$$

Note that the bound above is tight when $q(\kappa, \zeta|\tau, l) = P(\kappa, \zeta|\tau, l)$. Now,

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{\tau,l\sim D,\kappa,\zeta\sim q(\kappa,\zeta|\tau,l)}(\log (P(\tau, l)P(\kappa, \zeta|\tau, l)) \\
&\quad - \log q(\kappa, \zeta|\tau, l)) \\
&= \mathbb{E}_{\tau,l\sim D,\kappa,\zeta\sim q(\kappa,\zeta|\tau,l)}(\log P(s_1, l) \sum_{t=1}^{K} \{ \\
&\quad \log \pi^K(k_t|s_{1:t}, a_{1:t-1}, k_{1:t-1}, z_{1:t-1}, l) \\
&\quad \log \pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_{1:t}, z_{1:t-1}, l) \\
&\quad \log \pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t) \\
&\quad \log P(s_{t+1}|s_t, a_t)\} \\
&\quad - \log q(\kappa, \zeta|\tau, l)). \qquad (2)
\end{aligned}
$$

Where the last step uses the joint distribution derived in Equation (1). Keeping computational efficiency during training in mind, to enable the discrete skills and continuous parameters for each timestep to be sampled in parallel, we assume that conditional on previous states and actions, (k_t, z_t) is independent of (k_l, z_l) for $l < t$. This allows us to rewrite $\pi^K(k_t|s_{1:t}, a_{1:t-1}, k_{1:t-1}, z_{1:t-1}, l)$ as

[1] Here, d represents the number of continuous parameters.

[2] The subpolicy here is assumed to be independent of the task and the previous discrete/continuous parameters. This is consistent with our goal of learning skills that generalize *across* tasks and that can be specified by their discrete and continuous parameter.

$\pi^K(k_t|s_{1:t}, a_{1:t-1}, l)$ and $\pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_{1:t}, z_{1:t-1}, l)$ as $\pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_t, l)$. Since the subpolicy π^A is only conditioned on the *current* continuous and discrete parameters, one can show that for any fixed π^A, the joint likelihood in Equation (1) can be maximized by the simplified expressions for π^K and π^Z above. Our empirical results in Section 4 also validate that this assumption does not prevent useful skills from being learned.

Since $q(\kappa, \zeta|\tau, l)$ is meant to approximate $P(\kappa, \zeta|\tau)$, one can now rewrite $q(\kappa, \zeta|\tau, l)$ as follows:

$$q(\kappa, \zeta|\tau, l) = \prod_{t=1}^{K} q(k_t|\tau, l) q(z_t|\tau, k_t, l). \tag{3}$$

Using Equation (3) and the observation that the dynamics $P(s_{t+1}|s_t, a_t)$ and the joint distribution of the task and initial state, $P(s_1, l)$, do not affect the gradient of our loss, we can rewrite Equation (2) in the following, cleaner, form:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{\tau,l\sim D,\kappa,\zeta\sim q(\kappa,\zeta|\tau,l)}\Big(\sum_{t=1}^{K} \log \pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t)\Big) \\ - \mathbb{E}_{\tau,l\sim D}\Big(\sum_{t=1}^{K}\{D_{KL}(q(\cdot|\tau, l)||\pi^K(\cdot|s_{1:t}, a_{1:t-1}, l)) \\ - \mathbb{E}_{k_t\sim q(k_t|\tau,l)}(D_{KL}(q(\cdot|\tau, k_t, l)||\pi^Z(\cdot|s_{1:t}, a_{1:t-1}, k_t, l)))\}\Big).\end{aligned} \tag{4}$$

We represent the discrete skills using categorical distributions and the continuous parameters using Gaussian distributions. Hence, the KL-divergence terms above can be calculated exactly.

3.4. Algorithm

Pretraining (Skill Discovery): Equation (4) immediately suggests Algorithm 1 to jointly train a *discrete network* to learn π^K, a *continuous network* to learn π^Z, a *subpolicy network* to lean π^A, and a *variational network* to learn q.

Intuitively, sampling (κ, ζ) from *bidirectional* network enables the discovery of better skill abstractions by considering both past and future states/actions in the input trajectory. The first term in Equation (4) ensures q and subpolicy π^A can jointly predict the actions in the demonstration trajectory, while the KL-divergence terms train the autoregressive networks (π^K, π^Z), while also restricting q to distrubutions that can be approximated autoregressively at inference time.

Adaptation to Unseen Tasks and Inference: Note that policies (π^K, π^Z, π^A) are only conditioned on previously observed variables, and therefore can be used for inference. To finetune the model to an unseen task, we use the same procedure as in Algorithm 1, with the exception that in Line 4 of Algorithm 1, κ and ζ are sampled from $\pi^K(k_t|s_{1:t}, a_{1:t-1}, l)$ and $\pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_t, l)$ repectively (instead of $q(\kappa, \zeta|\tau, l)$). By making this change, we finetune the subpolicy using the discrete skills and continuous parameters predicted by the networks that will by used at *inference time* (i.e. π^K and π^Z). In practice, these distributions are close but not identical to the distributions predicted by $q(\kappa, \zeta|\tau, l)$. While this sampling scheme is biased, we empirically observe that it improves performance at inference time.

Algorithm 1 Learning Parameterized Skills

1: **Input:** Dataset D of trajectories $\{\tau, l\}$, $\tau = \{s_t, a_t\}_{t=1}^K$
2: Initialize networks: π^K, π^Z, π^A, q
3: **repeat**
4: Sample batch $\{\tau, l\}$ from D
5: Sample $\kappa, \zeta \sim q(\kappa, \zeta|\tau, l)$
6: Calculate $\mathcal{L}$ according to Equation (4)
7: Gradient step to maximize $\mathcal{L}$
8: **until** convergence
9: **Output:** π^K, π^Z, π^A

3.5. Avoiding Degenerate Solutions

A naive implementation of Algorithm 1 would be *underspecified*, i.e. the loss $\mathcal{L}$ in Equation (4) can be minimized without learning meaningful parameterized skills. In practice, we observe two primary modes of failure: (i) the continuous variables predicted by the variational-network change *at every timestep*, preventing compression from taking place (ii) the discrete and continuous parameters remain constant, essentially making the subpolicy network identical to one learned with multitask behaviour cloning. In this section, we outline the architecture of each network trained in Algorithm 1, explaining the design decisions made to prevent the issues above.

After sampling a batch of trajectories (Line 4, Algorithm 1), we first encode each observation using an image encoder. This *encoded input* is then passed to each of the networks described below

Variational Network: The variational network learns the distribution $q(\kappa, \zeta|\tau, l)$. We implement this as a bidirectional GRU that takes the encoded states and actions corresponding to a sampled demonstration trajectory as well as a task ID, as input. The outputs of the GRU (one for each timestep) are then passed through a prediction head to predict the discrete skill at each timestep as a categorical distribution. A separate output head is used to predict the continuous parameters. However, to prevent continuous parameters from changing at each timestep, we predict

the continuous parameters *for each discrete skill instead of each timestep*. This "restricts" the amount of information conveyable through the discrete skills and continuous parameters encoding a trajectory, forcing the subpolicy networks to learn *temporally extended* policies. [3]. We model the discrete parameters as Gaussian distributions using the reparameterization trick (Kingma & Welling, 2014).

Once the discrete skills are sampled, they can be used as indices into the set of predicted continuous parameters to choose the appropriate continuous parameter for each timestep. [4]

As training progresses, we observe that the average distance between continuous parameters increases, possibly indicating overfitting. To encourage compact and generalizable representations, we introduce a *Skill Parameter Norm Penalty* that discourages large-magnitude continuous parameters. Formally, we add the following regularization term to the loss in Equation (4):

$$\mathcal{L}_{\text{norm}} = \lambda_{\text{norm}} \sum_{i=t}^{K} \|z_t\|^2.$$

where z_t denotes the t-th continuous parameters predicted by the variational network, and λ_{norm} is a regularization coefficient.

Subpolicy Network: The subpolicy network learns the distribution $\pi^A(a_t|s_{1:t}, a_{1:t-1}, k_t, z_t)$, taking as input the current discrete and continuous parameters along with the previous observations. The input observations are then passed through a one-directional LSTM. We concatenate the selected continuous parameters to output embedding from the LSTM before passing them to a head returning the action distribution as a Gaussian mixture model.

As discussed, a common failure mode of Algorithm 1 is to encode all trajectories with the same continuous and discrete parameters. Empirically, we find that an algorithm trained with constant discrete/continuous parameters (i.e. multitask behavior cloning) achieves *the same behavior cloning loss* as an implementation learning abstractions.

[3]While this restriction of one continuous parameter value per discrete skill per trajectory works in our experiments, it is possible that longer-horizon trajectories will require the same discrete skill to be used with multiple continuous parameterizations. However, the same principle easily extends to this setting, as one can predict n sets of continuous parameters per discrete skill per trajectory, where n is a chosen hyperparameter.

[4]Another key difference between our architecture and that used by Shankar and Gupta (2020) is that we do not predict binary variables representing skill termination at each timestep. We find that including this variable makes it significantly harder to get training to stabilize by adding dependencies between different time steps, which also slows the implementation down.

This issue stems from the fact that, in the datasets we use, the state-spaces covered by demonstration trajectories from different tasks have a small intersection (we validate this in Appendix A), making it possible for a multitask network to "memorize" the training data.

Our key insight to tackle this issue is to use *information hierarchies*, i.e. instead of passing the full observation to the subpolicy network, we pass a lower-dimensional, "compressed", version. This makes the subpolicy network dependent on meaningful discrete skills and continuous parameters to minimize the behavior cloning loss as the compressed observation space does not contain sufficient "information" to predict the correct action in isolation.

Discrete & Continuous Networks: The Discrete and Continuous networks learn the distributions $\pi^K(k_t|s_{1:t}, a_{1:t-1}, l)$ and $\pi^Z(z_t|s_{1:t}, a_{1:t-1}, k_t, l)$ respectively. Both take as input the state/action history and task ID, which are passed through a one-directional GRU. The Discrete Network predicts the discrete parameter for each timestep as a categorical distribution, while the continuous network predicts one set of continuous parameters for each timestep and discrete skill (this output is then indexed with the sampled discrete skills).

Unlike the variational network, the continuous network predicts a continuous parameter *for every timestep*. This allows us to "close the loop" during inference time, i.e. the continuous network can refine the chosen continuous parameter at each timestep based on the most recent observation.

4. Experiments

We evaluate the performance of our algorithm on LIBERO (Liu et al., 2023), a multitask robot manipulation benchmark with a multi-modal observation space. First, we present quantitative results, where we benchmark the performance of our approach against pre-existing approaches. We then present a qualitative analysis showing that the approach learns semantically meaningful parameterized skills.

4.1. Quantitative Results

To pre-train model architectures, we use 80 tasks from LIBERO-90 using the offline dataset presented in the original paper. For each of the 80 pretraining tasks, the dataset provides 50 demonstrations collected using human teleoperation. The state-space for each task includes images from two different viewpoints, a 9-dimensional vector representing the robot's state, and a language description of the task at hand. The action space consists of a 6-dimensional real-valued vector representing desired arm movements, along with a binary variable to open/close the gripper.

After pretraining, we test each model architecture on 10

Table 1. Comparison of average success rates by amount of pretraining (PT) and algorithm. We report average success rates for each algorithm across 3 seeds and 20 epochs of finetuning for each unseen task (1-10). We also report the average success rates across tasks (Mean), the coefficient of variance over the success rates of different tasks (T-CV), and the average of the coefficient of variance across seeds for each task (S-CV). Best performance in each column is in bold. Learning parameterized skills results in superior performance for every metric, pretraining amount, and (almost every) downstream task.

PT	Algorithm	Unseen Task Index										Mean	T-CV	S-CV
		1	2	3	4	5	6	7	8	9	10			
0	BC (Untrained)	0.00	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.04	0.01	1.59	1.27
5	Parameterized Skills	**0.17**	**0.08**	**0.25**	**0.13**	0.16	**0.23**	**0.27**	**0.33**	**0.39**	**0.24**	**0.22**	**0.42**	**0.27**
	PRISE	0.02	0.02	0.02	0.01	0.02	0.08	0.00	0.00	0.01	0.05	0.02	1.06	1.61
	BC	0.02	0.01	0.06	0.03	**0.24**	0.13	0.02	0.10	0.05	0.07	0.07	0.94	0.87
10	Parameterized Skills	**0.12**	**0.08**	**0.23**	**0.12**	**0.24**	**0.27**	**0.24**	**0.47**	**0.35**	**0.21**	**0.23**	**0.50**	**0.23**
	PRISE	0.10	0.06	0.13	0.06	0.20	0.21	0.05	0.05	0.15	0.14	0.12	0.53	1.56
	BC	0.00	0.03	0.05	0.02	0.20	0.16	0.12	0.10	0.19	0.15	0.10	0.72	1.11
15	Parameterized Skills	0.09	**0.08**	**0.23**	**0.10**	**0.21**	0.21	**0.23**	**0.41**	0.30	**0.21**	**0.21**	**0.49**	**0.18**
	PRISE	**0.11**	0.02	0.08	0.03	0.17	**0.28**	0.11	0.09	0.22	0.12	0.12	0.65	1.29
	BC	0.00	0.07	0.01	0.01	0.19	0.09	0.19	0.16	**0.36**	0.11	0.12	0.93	1.00
20	Parameterized Skills	**0.06**	**0.08**	**0.24**	**0.07**	**0.24**	0.14	**0.21**	**0.36**	0.29	0.20	**0.19**	**0.52**	**0.27**
	PRISE	0.05	0.05	0.17	0.06	0.15	**0.31**	0.10	0.03	0.19	0.19	0.13	0.67	1.08
	BC	0.00	0.04	0.02	0.01	0.18	0.12	0.12	0.24	**0.45**	**0.31**	0.15	1.00	0.93

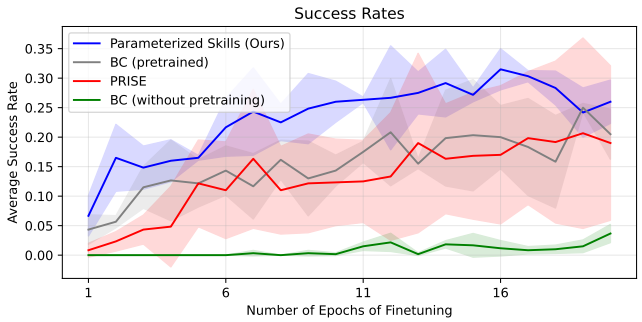

Figure 1. Success rate v.s. number of finetuning epochs (averaged over 10 unseen tasks and 3 seeds), with the shaded regions representing standard deviation across the 3 seeds. We find that our method achieves the highest success rate across all amounts of finetuning.

unseen tasks in Libero-90 involving previously unseen goals and environments. [5] We finetune the pretrained model architectures on 50 provided demonstration trajectories for each task in the test set *separately*. Descriptions of the 10 tasks in the test set can be found in Appendix B.

[5]Previous work has typically instead used the tasks in Libero-10 to test generalization. However, we find that most tasks in Libero-10 are a concatenation of tasks in Libero-90, and therefore a weak test of generalisation. On the other hand, our held-out set of test tasks involves unseen goals and environments.

4.2. Baselines

We compare our approach against the following baselines (1) A multitask behaviour cloning (BC) network using a ResNet-RNN setup analogous to LIBERO (Liu et al., 2023), (2) the same multitask BC architecture, but *without any pretraining*, and (3) PRISE (Zheng et al., 2024), a strong baseline learns action "tokens" and then applies byte pair encoding to learn common action sequences.

We use the same Resnet-18 architecture for all baselines, along with a hidden size of 1024 for all subsequent layers. More details on specific architectures and hyperparameters used can be found in Appendix C.

4.3. Experiment Setting and Results

For each algorithm, we perform pretraining, if applicable, on the 80 training tasks for 20 epochs using three seeds, saving checkpoints after every 5 epochs (i.e. four checkpoints are saved for each of the three seeds). Then, for each of the 10 tasks in the held-out test set, we finetune the pretrained models over 20 epochs. After each epoch of finetuning, we evaluate the success rate of each task with 20 rollouts. Success rates are then averaged across the three seeds for each amount of pretraining ($\{5, 10, 15, 20\}$ epochs).

In Figure 1, we show the success rate achieved by each algorithm as a function of the number of finetuning epochs. Results are averaged across the 10 held-out tasks and 3 seeds, and for each algorithm, we pick the amount of pretraining giving the highest *average* success rate. As seen in the figure, learning parameterized skills using our algorithm

achieves the highest success rate *irrespective* of the amount of finetuning performed.

We find that this result holds consistently for all amounts of pretraining. Table 1 shows the average success rate of each algorithm for every amount of pretraining and every unseen task. We highlight key takeaways below:

- Our method achieves the highest average success rate *for each amount of pretraining*, showing stronger performance across pretraining amounts.
- Learning parameterized skills results in the most *consistent* performance across unseen tasks. We measure this by calculating the coefficient of variance over the average success rates for each task (T-CV). For each amount of pretraining, our method has a significantly lower T-CV than other baselines. We attribute this to the fact that we learn *parameterized* skills in an *end-to-end fashion*. Learning parameterized skills yields expressive generalizations (leading to consistent downstream adaptation), while learning skills end-to-end makes it possible for our algorithm to adapt even when the previously learned abstractions are not the most appropriate.
- Learning parameterized skills is the most consistent method across different seeds/initializations. We measure this by calculating the coefficient of variance of the average success rate across 3 seeds for each task, and then taking the average of this value across the 10 unseen tasks (S-CV). Regardless of the amount of pretraining, learning parameterized skills results in significantly lower S-CV. This shows that our method is robust to different initializations.

4.4. Qualitative Results

We find that on pretraining on LIBERO-90, our algorithm discovers intuitive parameterized skills, primarily consisting of separate skills for grasping, moving and releasing objects. The same discrete skills are used consistently across environments and object types, with changing continuous parameters to encode task-specific details. We provide a representative visualization of the decomposition of trajectories into skills in Figure 2.

4.5. Conclusion

We presented an end-to-end method to learn parameterized skills from demonstration data. To do this, we initially derive a loss function using temporal variational inference. However, we explain that this loss function admits *degenerate solutions*. To avoid this, we utilize a series of information-theoretic methods such as information hierarchies. We show that the resulting algorithm learns semantically meaningful skills and consistently achieves higher success rates on unseen tasks in LIBERO compared to existing baselines, irrespective of pretraining amounts. We also show that our approach leads to more *consistent performance*, displaying a significantly lower coefficient of variance both across unseen tasks and across random seeds. We believe our approach shows promise towards better data-efficient task generalization in robotics.

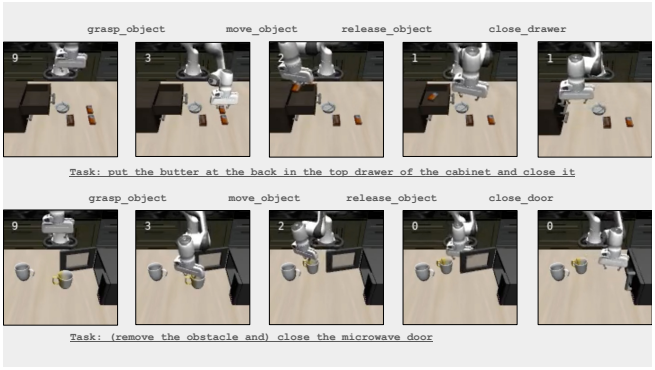

Figure 2. Visualization of the discovered discrete skills for two sample tasks. Images represent the points at which the sampled discrete skill changes, with the discrete skill taking place between two images indicated on top (the algorithm labels discrete skills with integers, which are visible in the figure. The natural language skills were selected to be consistent with the discovered segmentations of the demonstration trajectories). We find that general skills corresponding to grasping, moving, and releasing objects are learned and consistently applied across visually diverse tasks. Furthermore, for less frequent subtasks such as closing drawers and microwaves, dedicated discrete skills are discovered.

5. Acknowledgments

This work was carried out in collaboration with Haotian Fu, Calvin Luo, Yiding Jiang, and George Konidaris. I am deeply grateful for their advice and ideas, and especially for their patience through all the let's-start-from-scratch-and-rewrite-the-codebase phases it's endured. This project has truly challenged me and taught me a lot (so much so that it became my go-to answer to the "tell me about a time you took on a challenging project and faced failure" job interview question). None of this would have happened without the support I've received.

I especially want to thank Haotian for handing this project to me and for his willingness to work on it together at non-typical hours. I thank George for letting me constantly pester him for meetings and for giving in to my requests. Calvin and Yiding for being amazing mentors, and I thank them for their patience in answering all my questions. I also thank Michael for kindly agreeing to be my second reader.

This project would not have been possible without the tireless labour of far too many GPUs. On that note, I'd like to thank George for giving me access to extra GPUs, and Haotian for giving me even more by letting me use his GPUs, albeit via the wildly inefficient means of VLSMCTRs. [6]

References

da Silva, B. C., Konidaris, G. D., and Barto, A. G. Learning parameterized skills. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL http://icml.cc/2012/papers/826.pdf.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. Multi-level discovery of deep options. *CoRR*, abs/1703.08294, 2017. URL http://arxiv.org/abs/1703.08294.

Fu, H., Sharma, P., Stengel-Eskin, E., Konidaris, G., Roux, N. L., Côté, M., and Yuan, X. Language-guided skill learning with temporal variational inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=awo5H10K6v.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pp. 1181–1189, 2015.

Jiang, Y., Liu, E., Eysenbach, B., Kolter, J. Z., and Finn, C. Learning options via compression. *Advances in Neural Information Processing Systems*, 35:21184–21199, 2022.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pp. 3418–3428. PMLR, 2019.

[6] Very Long Slack Messages Containing Training Runs

Konidaris, G. D., Kuindersma, S., Grupen, R. A., and Barto, A. G. Robot learning from demonstration by constructing skill trees. *Int. J. Robotics Res.*, 31(3):360–375, 2012. doi: 10.1177/0278364911428653. URL https://doi.org/10.1177/0278364911428653.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pp. 418–437. PMLR, 2017.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Niekum, S., Chitta, S., Barto, A. G., Marthi, B., and Osentoski, S. Incremental semantically grounded learning from demonstration. In Newman, P., Fox, D., and Hsu, D. (eds.), *Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, June 24 - June 28, 2013*, 2013. doi: 10.15607/RSS.2013.IX.048. URL http://www.roboticsproceedings.org/rss09/p48.html.

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33: 2914–2924, 2020.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020.

Sharma, M., Sharma, A., Rhinehart, N., and Kitani, K. M. Directed-info GAIL: learning hierarchical policies from unsegmented demonstrations using directed information. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJeWUs05KQ.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press,

2018. URL http://www.incompleteideas.net/book/the-book-2nd.html.

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

Wan, W., Zhu, Y., Shah, R., and Zhu, Y. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 537–544. IEEE, 2024.

Zhang, J., Heo, M., Liu, Z., Biyik, E., Lim, J. J., Liu, Y., and Fakoor, R. EXTRACT: efficient policy learning by extracting transferable robot skills from offline data. In Agrawal, P., Kroemer, O., and Burgard, W. (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1148–1172. PMLR, 2024. URL https://proceedings.mlr.press/v270/zhang25c.html.

Zheng, R., Cheng, C., III, H. D., Huang, F., and Kolobov, A. PRISE: llm-style sequence compression for learning temporal action abstractions in control. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=p225Od0aYt.

A. Task Prediction from Observations

To motivate the need to provide a compressed version of the observation space to subpolicy networks, we study the overlap in the observation space across different tasks in Libero-90. Specifically, for each task in Libero-90, we split the 50 provided demonstration trajectories into 45 training trajectories and 5 test trajectories. Using the $45 * 90 = 4050$ trajectories in the training set, we train a simple classifier to predict which of the 90 tasks a **single observation** belongs to. Our model consists of the same Resnet-18 image encoder used in our experiments, with a two-layer MLP on top (with a hidden size of 1024). After two epochs through the training data, our model achieves an **accuracy of 84%** on the held-out test data. [7]

Note that our model has access to a single observation as input, and therefore can not analyse temporal information about the demonstration trajectory. Furthermore, the training and test datasets come from separate demonstration trajectories, so it is also not possible for the model to reason based on trajectory-specific attributes. We conclude that there must be little overlap in the observation space covered by demonstration trajectories corresponding to different tasks.

Note that the subpolicy network we train has access to previous observations in addition to the current observation, possibly increasing the disjointedness between observation encodings across different tasks. This explains why we observe no significant difference in the cloning loss achieved by a single model mapping observation to action compared to a naive implementation of temporal variational inference. As explained in Section 3.5, we mitigate this issue by passing a compressed version of the observation space to the subpolicy network, forcing meaningful discrete skills and continuous parameters to be learned in order to minimize behavior cloning loss.

B. Descriptions of Tasks in the Test Set

We provide descriptions of the tasks used in the test set below. For more details on the individual tasks, as well as those used in the training set, please refer to LIBERO (Liu et al., 2023).

Task Index	Task Description
1	STUDY SCENE2 pick up the book and place it in the right compartment of the caddy
2	STUDY SCENE3 pick up the book and place it in the front compartment of the caddy
3	STUDY SCENE3 pick up the book and place it in the left compartment of the caddy
4	STUDY SCENE3 pick up the book and place it in the right compartment of the caddy
5	STUDY SCENE3 pick up the red mug and place it to the right of the caddy
6	STUDY SCENE3 pick up the white mug and place it to the right of the caddy
7	STUDY SCENE4 pick up the book in the middle and place it on the cabinet shelf
8	STUDY SCENE4 pick up the book on the left and place it on top of the shelf
9	STUDY SCENE4 pick up the book on the right and place it on the cabinet shelf
10	STUDY SCENE4 pick up the book on the right and place it under the cabinet shelf

Table 2. Task descriptions with scene information

C. Model Architectures and Hyper-parameters

We use the Resnet-18 architecture provided in LIBERO (Liu et al., 2023) for each of the baselines. We highlight the architectural details of each baseline below:

C.1. Parameterized Skills

The variational network contains a two-layer bidirectional GRU with hidden size 1024. The individual heads for the discrete and continuous parameters both have two layers each with a hidden size of 1024. For every timestep, the output head for continuous parameters returns an array of dimensions $(|\kappa|, d)$, where $|\kappa|$ is the number of discrete skills (set to 10) and d is the dimensionality of the continuous parameters (set to 2). We then average the outputs of the continuous parameters output heads to get one set of continuous parameters per timestep, as highlighted in Section 3.5.

[7] While the observations used in LIBERO (Liu et al., 2023) include an embedding of the current task, we remove this embedding for the purposes of this experiment

The discrete and continuous networks both consist of one-directional GRU with a hidden size of 1024, followed by two layers of hidden size 1024.

The subpolicy network consists of a one-layer one-directional LSTM with hidden size 1024, followed by a Gaussian mixture model head with two intermediate layers and a hidden size of 1024. We use the same Gaussian mixture model implementation as LIBERO (Liu et al., 2023) with one exception - the action space of libero consists of 6 continuous variables and one binary variable. Therefore, we model the last dimension in the action space with a categorical distribution instead of a mixture of Gaussians (this helps made the behavior cloning loss values more instructive, as modeling a binary variable as a mixture of Gaussians leads to the variances of the corresponding Gaussian collapsing, making the loss deceptively low). We concatenate the continuous parameters to the output of the LSTM before passing them to the Gaussian mixture model.

The observation space of LIBERO consists of images from two viewpoints, the task description, and a 9-dimensional vector representing the position of the robot arm. To "compress" the observation space before passing to the sub-policy network (Section 3.5), we simply remove the images and task description from the observation, leaving behind the 9-dimensional vector representing *robot state*. This forces the learned discrete skills and continuous parameters to encode all relevant goal and environment information necessary for successful task completion.

To calculate the loss during pretraining, we weigh the KL divergence terms corresponding to the discrete skills and continuous variables by 0.1 and 0.01, respectively. The skill parameter norm penalty is weighted by 0.1 We train the resulting architecture with a batch size of 3 during pretraining and a batch size of 2 during finetuning. We use a learning rate of $3e-4$ and the random seeds 95, 96, and 97.

C.2. Behavior Cloning

We use the same general setup as described for parameterized skills. The main changes are that the variational, discrete, and continuous networks are no longer used, and the subpolicy network no longer receives discrete skills and continuous parameters in its inputs and receives an uncompressed version of the state space.

C.3. PRISE

We use the hyperparameters used in the original implementation (Zheng et al., 2024) along with the same seeds as the previously described approaches. The dataloader in the original PRISE implementation contains multiple instances of the same observation-action pair. Hence, to ensure fair comparison between baselines, we use a batch size of 35, which makes each batch in the PRISE implementation contain the same number of observations as the previous approaches on average. We then train PRISE for the same number of gradient steps as the approaches above.