ONE-SHOT IMITATION WITH SKILL CHAINING USING A GOAL-CONDITIONED POLICY IN LONG-HORIZON CONTROL

Hayato Watahiki & Yoshimasa Tsuruoka
The University of Tokyo
{watahiki,tsuruoka}@logos.t.u-tokyo.ac.jp

ABSTRACT

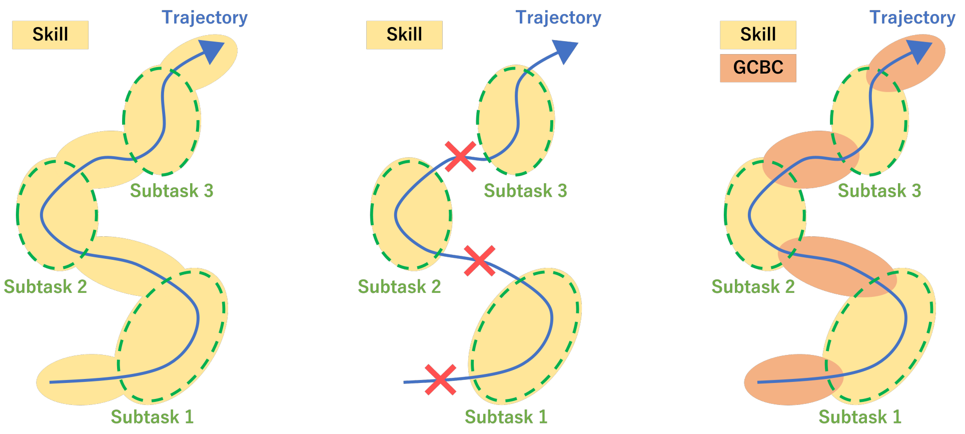
Recent advances in skill learning from a task-agnostic offline dataset enable the agent to acquire various skills that can be used as primitives to perform long-horizon imitation. However, most work implicitly assumes that the offline dataset covers the entire distribution of target demonstrations. If the dataset only contains subtask-local trajectories, existing methods fail to imitate the transitions between subtasks without a sufficient amount of target demonstrations, significantly limiting the scalability of these methods. In this work, we show that a simple goal-conditioned policy can imitate the missing transitions using only the target demonstrations. We combine it with a policy-switching strategy that uses the skills when they are applicable. Furthermore, we present multiple choices that can effectively evaluate the applicability of skills. Our new method successfully performs one-shot imitation with skills learned from a subtask-local offline dataset. We experimentally show that it outperforms other one-shot imitation methods in a challenging kitchen environment, and we also qualitatively analyze how each policy-switching strategy works during imitation.

1 INTRODUCTION

Learning long-horizon policies from a few demonstrations is one of the biggest challenges in developing a practical method of training robot agents in the physical world. The ultimate goal is to create robot agents that we can instruct by a *single* demonstration. However, most imitation learning methods require many demonstrations of a target task to cover a larger state distribution for generalization. Collecting a sufficient number of demonstrations, which sometimes needs human experts to perform the tasks, is expensive and limits the scalability of imitation methods. To remove the necessity of laborious data collection for each target task, we have to utilize task-agnostic data that could be collected less expensively.

Acquiring a behavioral prior or a set of temporally extended actions, often called *skills*, from a large-scale offline dataset is a promising direction toward this goal (Singh et al., 2021; Pertsch et al., 2020; Ajay et al., 2021). Several previous studies investigated the possibility of utilizing skills in imitation as well (Pertsch et al., 2021; Yang et al., 2022; Hakhamaneshi et al., 2022). However, in skill-based reinforcement learning and imitation learning, it is usually assumed that the learned skills are diverse enough to cover the wide distribution necessary for downstream tasks (see Figure 1a, 1b). This assumption could be problematic when the agent needs to deal with a variety of target long-horizon demonstrations, since the entire trajectory of a target task, including transitions between subtasks, should be interpreted with the learned skill set. It means that the dataset for offline skill learning should contain all transitions between subtasks. If a large number of target demonstrations are available for skill fine-tuning, the agent can still adapt to unseen transitions without such a rich offline dataset, but in a one-shot setting, that is not achievable. It is undesirable in a sense that the goal of a skill-based agent is to achieve generalization to unseen compositions of lower-level skills.

For better connecting skills to perform longer tasks, several studies trained a transitional policy through interactions (Lee et al., 2018; Tian et al., 2021). Lee et al. (2021) match the terminal state distribution of a preceding skill and the initial state distribution of the following skill in the long-



(a) Previous skill-based imitation with demonstration-covering dataset (b) Previous skill-based imitation with subtask-local dataset (c) Our method with subtask-local dataset

Figure 1: Requirement of the previous skill-based imitation on the dataset. The yellow zones in the figures show the areas the dataset for skill learning covers. (a) Previous methods assumed the wide coverage of the dataset. (b) If we use the dataset that does not satisfy the requirement (e.g. subtask-local dataset), a purely skill-based agent fails to perform the task. (c) Our method fills the gap with a goal-conditioned policy that achieves better adaptation.

horizon imitation, and other approaches utilize additional interactions to find a better imitation policy for the target task demonstrated in the given trajectories (Dance et al., 2021; Pertsch et al., 2021). However, none of them can satisfy the requrement in our setting, which is one-shot imitation of an unseen sequence of subtasks without additional interactions.

In this work, we present a one-shot imitation method that can adapt to unseen long-horizon tasks with skills learned from an offline dataset that only contains trajectories around subtasks. For transitions between subtasks, we can only use a target demonstration itself for the policy learning, which hinders sufficient skill training for one-shot imitation. We address this problem using a simple goal-conditioned policy trained through behavioral cloning with hindsight goal relabeling similar to Andrychowicz et al. (2017). Thanks to the simplicity of the architecture, it can be trained enough only with a limited number of transition data. With this policy, we can use skills only when they are applicable, which corresponds to the area around subtasks the dataset is collected for. Furthermore, in our method, skills can be used without fine-tuning on a target demonstration, which accelerates the reusability of skills as a fixed behavioral primitive in a modular way. Our method is illustrated in Figure 1c.

We also present two approaches to find where the skill set is applicable so that the agent can switch between the skill policy for presumed subtasks and the goal-conditioned policy for elsewhere. One way is to directly predict the plausibility of using skills given a current state and a subgoal, and the other way is to scan the demonstration in advance and add a tag that represents where to use the skills according to the reconstruction loss of actions with a skill variational autoencoder (skill VAE). In addition to the quantitative comparison of each switching method, we analyze where the skills are used in each switching policy and confirm its soundness.

In summary, our contributions are threefold:

- We show that existing skill-based imitation methods rely on the strong assumption that the offline dataset for skill learning covers the entire trajectories of target tasks.
- We propose a novel one-shot imitation algorithm for long-horizon tasks that switches a skill policy and a goal-conditioned policy according to the applicability of learned skills.
- We quantitatively evaluate the efficacy of our method in a challenging kitchen environment and also visualize how each policy switching strategy works.

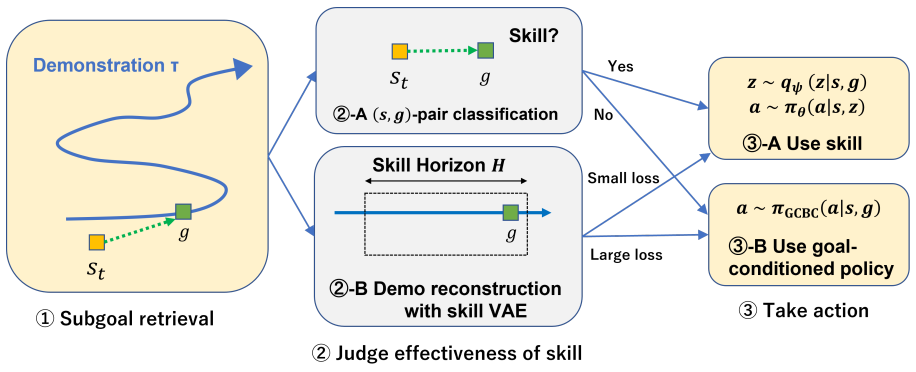

Demonstration τ
s_t
g
① Subgoal retrieval
Skill?
②-A (s,g)-pair classification
Skill Horizon H
②-B Demo reconstruction with skill VAE
② Judge effectiveness of skill
Yes
No
Small loss
Large loss
$z \sim q_\psi(z|s,g)$
$a \sim \pi_\theta(a|s,z)$
③-A Use skill
$a \sim \pi_{\mathrm{GCBC}}(a|s,g)$
③-B Use goal-conditioned policy
③ Take action

Figure 2: Overview of our algorithm. After finding a good subgoal in the procedure proposed in Hakhamaneshi et al. (2022), we judge the applicability of skill given the current state and the subgoal, then finally infer the action using a policy that is selected in the previous process.

2 Approach

Our goal in this paper is to imitate a single long-horizon demonstration with a set of skills acquired from a large-scale unstructured dataset that only covers parts of the target trajectory (e.g. a dataset of subtask demonstrations). As seen in Figure 1, without the assumption that the offline dataset for skill learning contains various trajectories enough to interpret the entire target trajectory with skills, the agent tends to fail to imitate a target demonstration. To compensate for the absence of effective skills where the dataset does not have enough coverage, we introduce a single-step goal-conditioned policy that can adapt a novel target trajectory with a small amount of data.

Our proposed method is composed of three parts: (1) skill extraction and goal-conditioned skill-based imitation based on Hakhamaneshi et al. (2022) (2) learning a single-step goal-conditioned policy from the dataset with goal relabeling, and (3) policy switching between the skill policy and the goal-conditioned policy according to the applicability of skills. An overview of our method is shown in Figure 2. Each component will be described in Sec. 2.2, 2.3 and 2.4, respectively.

2.1 Problem Formulation

We formulate our problem as a Markov decision process (MDP) defined as $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $P(s_{t+1}|s_t, a_t)$ is a transition function, $R(s_t, a_t, s_{t+1})$ is a reward function, and $\gamma \in (0, 1)$ is a discount factor. Instead of learning policy $\pi(a_t|s_t)$ through maximizing the expected cumulative reward $\mathbb{E}_\pi\left[\sum_t \gamma^t R(s_t, a_t, s_{t+1})\right]$ as in reinforcement learning, the goal of an imitation agent is to mimic a single expert demonstration of a target task τ_{demo}, which is defined as a sequence of state-action pairs $\tau_{\text{demo}} = \{(s_0, a_0), \dots, (s_{T-1}, a_{T-1})\}$ where T is the length of the demonstration. In addition to that, we assume the access to the offline play dataset $\mathcal{D}$ from various tasks which is defined as a set of trajectories $\mathcal{D} = \{\tau_i\}$. Although imitating a trajectory in the dataset does not directly empower the performance of the target task, the agent can utilize it to learn skills for better imitation.

2.2 Skill Learning and Skill-Based Imitation

Skill-based imitation in our algorithm is based on the method presented in Hakhamaneshi et al. (2022). It performs goal-conditioned skill-based imitation with a subgoal retrieved from target demonstrations using an inverse skill dynamics model.

Skill Extraction In our method, skill z is defined as a continuous embedding for a sequence of state-action pairs with a fixed length H: $\tau = \{(s_t, a_t), \dots (s_{t+H-1}, a_{t+H-1})\}$. It can be employed as a higher-level action to accelerate the learning of long-horizon tasks. To learn the skill embedding z, we train a stochastic latent variable model conditioned by states, which we call a *skill VAE*. The encoder $q(z|\tau)$ is an LSTM-based model that approximates the posterior as a Gaussian distribution.

Algorithm 1 Goal-Conditioned Behavioral Cloning (GCBC)

```
**Input:** offline dataset $\mathcal{D}$, horizon range $[H_{\text{min}}, H_{\text{max}}]$, goal-conditioned policy $\pi_{\theta_g}(a|s, g)$
Initialize goal-conditioned policy: $\pi_{\theta_g}$
**for** every iteration $k = 1, 2, \dots$ **do**
    Sample trajectory: $\tau \sim \mathcal{D}$
    Randomly choose the horizon: $H' \sim \text{Uniform}(H_{\text{min}}, H_{\text{max}})$
    Sample sub-trajectory with length $H'$: $(s_t, a_t, \dots, s_{t+H'-1}, a_{t+H'-1}) \sim \tau$
    Relabel final state as goal: $g = s_{t+H'-1}$
    Update goal-conditioned policy: $\theta_g = \arg\max_{\theta_g} \log \pi_{\theta_g}(a_t|s_t, g)$
**end for**
```

The decoder $\pi(a_t|s_t, z)$ reconstructs an action given the current state and skill, which corresponds to the lower-level policy of a hierarchical model. The training objective of the skill VAE is to maximize the evidence lower bound (ELBO):

$$\log p(a_t|s_t) \geq \mathbb{E}_{\tau\sim\mathcal{D}, z\sim q_\phi(z|\tau)} \left[\log \pi_\theta(a_t|s_t, z) - \beta D_{\text{KL}} \left[q_\phi(z|\tau) \| p(z)\right]\right], \tag{1}$$

where the skill posterior $p(z)$ is set to be a unit Gaussian $\mathcal{N}(0, I)$, and ϕ and θ are the parameters of q_ϕ and π_θ, respectively. In addition to the skill representation z, we learn inverse skill dynamics $q_\psi(z|s_t, s_{t+H-1})$ with parameter ψ that infers the skill used given the current state and the future state. It is trained through KL minimization with a loss $\mathcal{L}_{\text{prior}}$:

$$\mathcal{L}_{\text{prior}} = \mathbb{E}_{\tau\sim\mathcal{D}} \left[D_{\text{KL}} \left[q_\phi(z|\tau) \| q_\psi(z|s_t, s_{t+H-1})\right]\right]. \tag{2}$$

The skill embedding and the inverse skill dynamics model are jointly trained with (1) and (2).

Goal-Conditioned Skill-Based Imitation For inference, we first fine-tune the parameters of the skill VAE and the inverse skill dynamics model using an expert demonstration. And then, in inference, the agent retrieves a subgoal g from the demonstration trajectory τ_{demo} and infers the skill z to perform for a few timesteps by the inverse model $q(z|s_t, g)$. Finally, the primitive action to perform in the next step is predicted using skill decoder $\pi_\theta(a_t|s_t, z)$. The subgoal retrieval from the demonstration is done with a learned distance function $d(s_i, s_j)$. The subgoal g is set to the state which is H-step ahead of the state $s^* = \arg\min_{s\in\tau_{\text{demo}}} d(s_t, s)$. The distance metric $d(s_i, s_j)$ is learned through the InfoNCE loss (Oord et al., 2018) based on the reachability within H steps. The summary of the procedure and the details of distance learning is provided in Appendix A.1, A.2.

2.3 Learning Goal-Conditioned Policy

We learn a single-step goal-conditioned policy from the same dataset used in the skill learning. To train the policy, we use goal-conditioned behavioral cloning (GCBC). GCBC is sometimes employed as a baseline or a component of previous work (Lynch et al., 2019; Gupta et al., 2019). Given the offline play dataset, it takes randomly cropped sub-trajectories and relabels the terminal state as a goal of each sequence as in Lynch et al. (2019). GCBC then updates the policy $\pi_{\text{GCBC},\theta_g}(a_t|s_t, g)$ so that it maximizes the likelihood of action a_t conditioned by the current state s_t and relabeled terminal state g. The window size is randomly selected around the skill horizon H so that the same goal can be used for both the skill-based imitation and the goal-conditioned policy in the policy switching explained in the following section. The procedure is summarized in Algorithm 1.

2.4 Policy Switching

In skill-based imitation described in Sec. 2.2, we propose to use skills only when they are applicable. For that purpose, we present two ways to find where the learned skills are applicable. Combined with one of these strategies, the agent can select a better policy for one-shot imitation. The whole procedure is summarized in Algorithm 2. The red lines are the differences from the original skill-based imitation (c.f. Appendix A.1)

2.4.1 Training Skill Effectiveness Classifier

One way of judging the skill effectiveness is to train a classifier $f_\xi(s, g)$ that discriminates state-goal pairs found in the offline transition dataset from the random ones, since the dataset is used for skill

Algorithm 2 Imitation with Policy Switching

```
Input: inverse skill dynamics $q_\psi(z|s_t, s_{t+H-1})$ skill decoder $\pi_\theta(a|s, z)$, distance function $d(s, s')$, demonstration $\tau_{\text{demo}}$, skill horizon $H$, task horizon $T$, goal-conditioned policy $\pi_{\text{GCBC}}(a|s, g)$
Method specific input: effectiveness classifier $f_\xi(s, g)$ or skill encoder $q_\phi(z|\tau)$
Set initial state $s_0$
(In VAE method only) Scan demonstration $\tau_{\text{demo}}$ by Algorithm 3 to get policy label $\{b_i\}$
for $t = 0, \ldots, T-1$ do
    Look up goal state: $g = H$-step-lookahead$(\arg\min_{s \in \tau_{\text{demo}}} d(s_t, s))$
    if is_skill_effective() then
        Infer skill: $z \sim q_\psi(z|s_t, g)$
        Infer action: $a_t \sim \pi_\theta(a|s_t, z)$
    else
        Infer action: $a_t \sim \pi_{\text{GCBC}}(a|s_t, g)$
    end if
    Take action: $s_{t+1} =$ env.step$(a_t)$
end for
```

Algorithm 3 Offline Demonstration Scanning

```
Input: demonstration $\tau_{\text{demo}}$, skill horizon $H$, skill encoder $q_\phi(z|\tau)$ skill decoder $\pi_\theta(a|s, z)$, loss threshold $\alpha$, policy labels $\{b_0, \ldots, b_{T-1}\}$
Initialize policy labels: $\{b_0, \ldots, b_{T-1}\} = \{$"GCBC"$, \ldots,$ "GCBC"$\}$
for $t = 0, \ldots, T-H$ do
    Pick up sub-trajectory from $\tau_{\text{demo}}$: $\tau = (s_t, a_t, \ldots, s_{t+H-1})$
    Infer skill representation: $z \sim q_\phi(z|\tau)$
    Reconstruct actions for each $t' \in [t, t+H-1]$: $\hat{a}_{t'} \sim \pi_\theta(a_{t'}|s_{t'}, z)$
    Calculate reconstruction error: $l = \frac{1}{H}\sum_{t'=t}^{t+H-1}(a_{t'} - \hat{a}_{t'})^2$
    if $l < \alpha$ then
        Label terminal state as "Skill": $b_{t+H-1} =$ "Skill"
    end if
end for
return policy labels $\{b\}$
```

learning as well. The state-goal pairs for the training are created from random sub-trajectories of length H' close to the skill horizon H as done in GCBC. A positive example (s_t, g) is composed of the initial state s_t and the terminal state $g = s_{t+H'-1}$ of the sub-trajectory, while a negative example is created by replacing either or both states of the tuple with a random state in the dataset or randomly generated state from some distribution. The positive examples, the negative examples with random states from the data, and the negative examples with randomly generated states are equally contained in the batch in an iteration. The algorithm is summarized in Appendix A.3.

2.4.2 Offline demonstration scanning with Skill VAE

Another way of estimating the effectiveness of skills is to refer to the reconstruction loss of the skill VAE. Assuming that the agent follows the demonstration trajectory accurately enough, which means that every state in inference is close to the state in the demonstration, we can measure the accuracy of action prediction using the reconstruction loss of the skill VAE. As this process does not depend on any online information, it could be processed before the imitation starts. We scan the provided demonstration with a fixed window size H and assign the decision to the terminal state of each sub-trajectory so that the label of the goal retrieved in the imitation process shows the effectiveness of the skill toward the goal state. In this work, we judge the skill as effective when the reconstruction loss is smaller than the pre-defined threshold α, which is a hyperparameter that can be set according to the average reconstruction loss during the training of the skill VAE. To summarize, the scanning process should look like Algorithm 3.

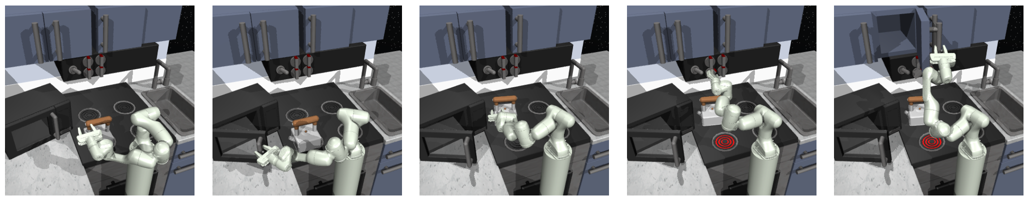


Figure 3: Kitchen environment. The agent is required to perform four subtasks consecutively. Each picture shows the initial position, the robot opening a microwave, moving a kettle, turning on a bottom burner, and opening a hinge cabinet, respectively.

3 Related Work

Our approach is built upon previous work on offline skill discovery that extracts behavioral prior from static task-agnostic datasets (Pertsch et al., 2020; Ajay et al., 2021; Singh et al., 2021; Yang et al., 2022). Some studies attempted to use skill or equivalent action representation in imitation as well (Pertsch et al., 2021; Yang et al., 2022; Hakhamaneshi et al., 2022), but it is commonly assumed that the dataset covers the entire trajectory of possible downstream tasks including transitions between subtasks. Thus, these approaches should struggle to deal with unseen transitions between subtasks in a long-horizon demonstration if the dataset does not satisfy the assumption.

Some previous methods explicitly trained a transitional policy toward a specific subtask (Lee et al., 2018) or between specific subtasks (Tian et al., 2021), and the other method improved the connection between specific subtasks (Lee et al., 2021). Also, several imitation methods allow the agent to interact with the environment (Dance et al., 2021; Pertsch et al., 2021), which could deal with unseen transitions. Our method, in contrast, realizes one-shot imitation without assuming a specific order of subtasks or additional interaction using skills to perform complex long-horizon tasks.

There is also another line of work that aims to realize one-shot imitation, first proposed in Duan et al. (2017). MAML (Finn et al., 2017a)-based approaches adapt policy parameters to an unseen task from a visual demonstration (Finn et al., 2017b; Yu et al., 2018) and it can also deal with long-horizon tasks using a hierarchical structure (Yu et al., 2019). James et al. (2018) learn a task representation to learn to a new task from one or more demonstrations. In contrast, our approach simply fine-tunes the goal-conditioned policy to connect known subtask skills in long-horizon control without requiring an additional framework or representation.

4 Experiments

Our experiments aim to answer the following questions: (i) When the offline dataset does not cover the entire demonstration, how does the performance change depending on the number of demonstrations? (ii) Can the goal-conditioned policy help long-horizon skill-based imitation with a single demonstration? (iii) How does each policy switching strategy work in actual imitation processes?

4.1 Environments

We evaluate our one-shot imitation method in simulated robotic manipulation tasks (Gupta et al., 2019) shown in Figure 3. In the environment, a 9-DoF robot interacts with a kitchen scene to perform four consecutive subtasks such as opening a microwave, moving a kettle, turning on a light switch. The agent receives a state vector in each timestep and outputs a 9-dimensional continuous action. Although the agent receives a reward of +1 for the completion of each subtask, it is only used for evaluation purposes. For offline skill learning, we design a dataset that only contains subtask-local trajectories by removing transitions between subtasks from the original dataset collected via human teleoperation. The dataset contains around 2300 trajectories of all seven subtasks in total. We use three long-horizon tasks in all experiments: (Task 1) Microwave $\rightarrow$ Kettle $\rightarrow$ BottomBurner $\rightarrow$ Hinge, (Task 2) Microwave $\rightarrow$ Kettle $\rightarrow$ Switch $\rightarrow$ Slide, (Task 3) BottomBurner $\rightarrow$ TopBurner $\rightarrow$ Slide $\rightarrow$ Hinge. More details are provided in Appendix C.1.

Table 1: Effect of the dataset and the number of demonstrations. The percentage shows the success rate of all four consecutive subtasks, while the number in parentheses presents the reward received in imitation. The performances with the ideal dataset are shown in columns with (ideal).

Task	0-shot	**1 demo**	10 demos	0-shot (ideal)	1 demo (ideal)	10 demos (ideal)
Task 1	0 % (0)	0 % (0.85)	65 % (3.5)	0% (2.45)	35% (3.0)	80% (3.75)
Task 2	0 % (0.05)	0 % (0.6)	45 % (2.45)	20% (2.45)	55% (3.1)	55% (3.0)
Task 3	0 % (0)	35 % (2.4)	80 % (3.7)	80% (3.4)	65% (2.95)	60% (3.0)

Table 2: Success rate and reward of each method. Switch achieves better adaptation than FIST and other baselines with the help of GCBC where a sufficient amount of data is not available.

Task	BC	GCBC	FIST	Switch (CLS)	Switch (OldVAE)	Switch (VAE)
Task 1	0 % (1.2)	**70 % (3.55)**	0 % (0.85)	**70 % (3.5)**	5 % (2.55)	**70 % (3.55)**
Task 2	0 % (0.3)	65 % (3.45)	0 % (0.6)	40 % (2.3)	35 % (2.6)	**95 % (3.9)**
Task 3	10 % (1.65)	10 % (1.95)	35 % (2.4)	**35 % (3.05)**	0 % (1.95)	10 % (2.45)

4.2 Effect of dataset and the number of demonstrations

We measure the performance of skill-based imitation changing the number of expert trajectories using the subtask-local dataset and ideal, demonstration-covering dataset. The ideal dataset is made by eliminating demonstrations of the target task from the original dataset. We first trained the skill using an offline dataset, and then fine-tune every parameter using target demonstrations. We use the same single trajectory for retrieving the subgoal for a fairer comparison. As shown in Table 1, if we can use an ideal dataset that contains trajectories between subtasks, one-shot or even zero-shot imitation achieved a partial success. On the other hand, without access to the ideal dataset or a sufficient amount of demonstrations, the skill-based imitation mostly failed even though a lot of trajectories are available for every subtask.

4.3 Comparative results of one-shot imitation

We compare the efficacy of our method with the following baselines: **BC**: A behavioral cloning agent without being conditioned by goals, **GCBC**: A goal-conditioned agent described in Section 2.3 with goals retrieved in the same way as in skill-based imitation (c.f. Section 2.2), **FIST** (Hakhamaneshi et al., 2022): A baseline skill-based imitation agent on which we built our method. Our approach is denoted as **Switch**, which switches GCBC and FIST in the way described in Algorithm 2. We also have three variants for skill switching policies of Switch: **CLS** is a classifier-based switching (Section 2.4.1), **VAE** is a method based on the reconstruction loss (Section 2.4.2) using a skill VAE fine-tuned with a target demonstration, and **OldVAE** is the same as VAE except that it uses the skill VAE without fine-tuning. For every method, we first trained the model with the offline dataset and then fine-tuned it with a single demonstration.

The results are summarized in Table 2. BC cannot deal with multi-modality in an undirected offline dataset and long-horizon tasks, which results in low scores. GCBC exhibits good performance in some compositions of subtasks like Task 1 and 2 since they happened to be easy to imitate only with the flat policy when combined with the goal retrieval of FIST, but if the task contains a part that is difficult to perform without skills, GCBC failed to perform the whole task. It shows how fast GCBC can adapt to the target tasks only with a single trajectory. As for FIST, as already seen in Table 1, failed to adapt to the task due to the insufficiency of demonstration. In contrast, Switch achieves higher performance than purely skill-based imitation by taking advantage of the fast adaptation of GCBC. Although no switching strategy we present achieves the best performance consistently, one of the Switch variants shows the best performance in every task.

We also compare the performance of skill-based agents without skill fine-tuning with the target demonstration. The results in Table 3 show the necessity of skill fine-tuning for downstream imitation in FIST. However, we would sometimes like to use the skill in its original form, since the update can break the performance of skills, that is, the performance in subtasks in this experiment. Switch

Table 3: Success rate and reward of each method without skill fine-tuning. Even with un-updated skills, Switch can imitate a target demonstration that has a novel composition of subtasks and transitions between them.

Task	FIST	Switch (CLS)	Switch (OldVAE)	Switch (VAE)
Task 1	0 % (0.0)	**55 % (3.0)**	10 % (2.95)	5 % (1.05)
Task 2	0 % (0.05)	50 % (2.8)	**90 % (3.75)**	**85 % (3.6)**
Task 3	0 % (0.1)	0 % (0.8)	10 % (2.8)	**40 % (3.04)**

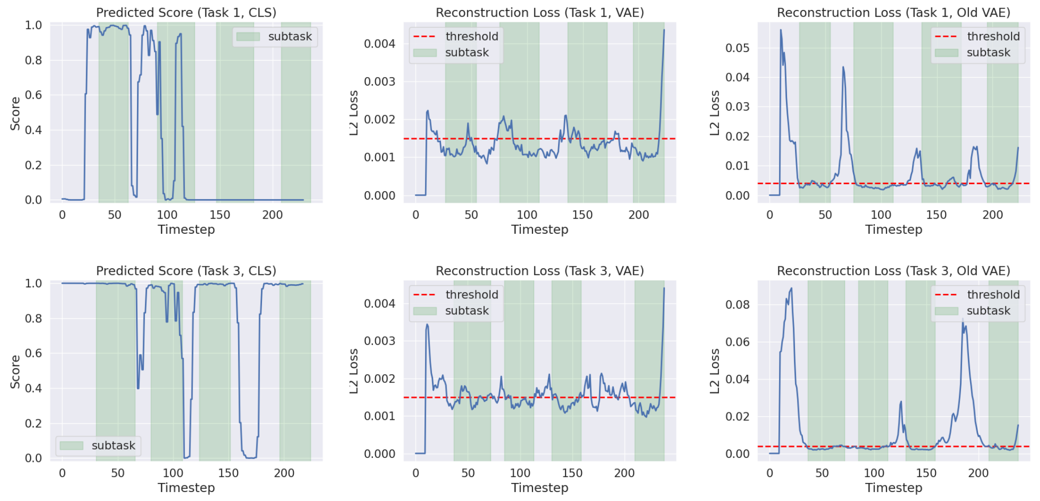


Figure 4: Visualization of policy switching. The prediction score is plotted for CLS while the L2 loss of the reconstruction is plotted for VAE and OldVAE. The green zone shows the time for subtasks based on when the agent received the reward signal. See Appendix B for more results.

achieves comparative performance to the one with fine-tuned skill, which enables us to use a skill as a fixed module. Additional results for two more tasks can be found in Appendix. B.

4.4 Qualitative analysis of policy switching

Figure 4 shows how each switching strategy works in three tasks. The green area in the figure shows the time that corresponds to the subtasks contained in the offline dataset. Thus, ideally, skills are used in the green zone whereas they should not be used outside of it. Both CLS and VAE/OldVAE provide a correct switching between the skill policy and the single-step goal-conditioned policy. CLS tends to recognize skills as ineffective as time goes on, possibly because the agent is likely to deviate from the ideal states that appeared in the dataset. In OldVAE, which uses the skill VAE without an update, the difference in reconstruction loss is more salient than in VAE, since the transitions between subtasks are completely novel for OldVAE. However, unfortunately, it does not directly improve the imitation performance in our experiment.

5 Conclusion

In this work, we show that a purely skill-based one-shot imitation method struggles to adapt to the demonstration if we do not assume enough coverage of the dataset over the trajectory. To address the issue, we present a method to imitate a long-horizon trajectory by combining a goal-conditioned policy with skill-based imitation. Our experiment shows the efficacy of our method, and also presents the possibility of skills to be used as a fixed module for better compositionality and scalability toward generalizable policy learning. Promising directions of future work should include improving the policy-switching strategy, evaluating our method using a real robot system, and extending our motivation to the setting with observation-only demonstrations.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. {OPAL}: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Christopher R. Dance, Julien Perez, and Théo Cachet. Demonstration-conditioned reinforcement learning for few-shot imitation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 2376–2387. PMLR, 2021.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017a.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pp. 357–368. PMLR, 2017b.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, pp. 1025–1037. PMLR, 2019.

Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. In *International Conference on Learning Representations*, 2022.

Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. In *Conference on robot learning*, pp. 783–795. PMLR, 2018.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2018.

Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In *5th Annual Conference on Robot Learning*, 2021.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pp. 1113–1132. PMLR, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. In *4th Annual Conference on Robot Learning*, 2020.

Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J Lim. Demonstration-guided reinforcement learning with learned skills. In *5th Annual Conference on Robot Learning*, 2021.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021.

Qiangxing Tian, Jinxin Liu, Guanchu Wang, and Donglin Wang. Unsupervised discovery of transitional skills for deep reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533820.

Mengjiao Yang, Sergey Levine, and Ofir Nachum. TRAIL: Near-optimal imitation learning with suboptimal data. In *International Conference on Learning Representations*, 2022.

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems*, 2018.

Tianhe Yu, Pieter Abbeel, Sergey Levine, and Chelsea Finn. One-shot hierarchical imitation learning of compound visuomotor tasks. In *International Conference on Intelligent Robots and Systems*, 2019.

A Details of previous work and proposed method

A.1 Summary of skill-based imitation

The baseline algorithm for skill-based imitation from Hakhamaneshi et al. (2022), called FIST, is summarized as follows.

Algorithm 4 Skill-based Imitation with FIST

```
Input: inverse skill dynamics $q_\psi(z|s_t, s_{t+H-1})$ skill decoder $\pi_\theta(a|s, z)$, distance function $d(s, s')$, demonstration of target task $\tau_{\text{demo}}$, skill horizon $H$, task horizon $T$
Set initial state $s_0$
for $t = 0, \ldots, T-1$ do
    Look up goal state: $g = \text{H-step-lookahead}(\arg\min_{s \in \tau_{\text{demo}}} d(s_t, s))$
    Infer skill: $z \sim q_\psi(z|s_t, g)$
    Infer action: $a_t \sim \pi_\theta(a|s_t, z)$
    Take action: $s_{t+1} = \text{env.step}(a_t)$
end for
```

A.2 Distance learning in FIST

The distance between states for the goal retrieval in FIST is defined as the euclidean distance between the encoded states with encoder $h(s)$.

$$d(s, s') = \|h(s) - h(s')\|^2 \tag{3}$$

The encoder h is learned through the InfoNCE loss (Oord et al., 2018) using the offline dataset and a downstream demonstration. Given a state s_t, another state which cannot be feached within the skill horizon H, i.e. $s_{t'}$ s.t. $t' \leq t + H$ in the same trajectory, is regarded as a positive example, while the every other state is treated as a negative example for the contrastive objective.

A.3 Summary of training of effectiveness classifier training

The procedure of the training of the effectiveness classifier is summarized in Algorithm 5.

Algorithm 5 Training Effectiveness Classifier

```
Input: offline trajectories $\mathcal{D}$, horizon range $[H_{\text{min}}, H_{\text{max}}]$, effectiveness classifier $f_\xi(s, g)$, state distribution $\sigma(s)$
Initialize effectiveness classifier: $f_\xi(s, g)$
for every iteration $k = 1, 2, \ldots$ do
    Sample sub-trajectory with length $H' \in [H_{\text{min}}, H_{\text{max}}]$: $(s_t, a_t, \ldots, s_{t+H'-1}, a_{t+H'-1}) \sim \mathcal{D}$
    Relabel final state as goal: $g = s_{t+H'-1}$
    Sample random value: $x \sim \text{Uniform}(0, 1)$
    if $x < 1/3$ then
        Set a positive target: $y = 1$
    else if $x < 2/3$ then
        Sample random states from buffer: $s_{\text{random}}, g_{\text{random}} \sim \mathcal{D}_{\text{demo}}$
        Set random state or goal randomly: $s_t = s_{\text{random}}, g = g_{\text{random}}$
        Set negative target: $y = 0$
    else
        Generate fake states: $s_{\text{fake}}, g_{\text{fake}} \sim \sigma(s)$
        Set fake state or goal randomly: $s_t = s_{\text{fake}}, g = g_{\text{fake}}$
        Set negative target: $y = 0$
    end if
    Learn classifier $\xi = \arg\max_\xi \log f_\xi(y|s_t, g)$
end for
```

Table 4: Success rate and reward of each method in Task 4 and 5.

Task	BC	GCBC	FIST	Switch (CLS)	Switch (OldVAE)	Switch (VAE)
Task 4	0 % (0.85)	0 % (0.65)	**20 % (2.8)**	10 % (1.85)	**25%** (2.05)	0% (1.2)
Task 5	0 % (1.2)	**40 % (2.8)**	10% (1.45)	25 % (2.35)	25 % (2.4)	**40 % (2.6)**

Table 5: Success rate and reward of each method without skill fine-tuning in Task 4 and 5. Even with un-updated skills, Switch can imitate a target demonstration that has a novel composition of subtasks and transitions between them.

Task	FIST	Switch (CLS)	Switch (OldVAE)	Switch (VAE)
Task 4	0 % (0.0)	5% (1.45)	**20% (1.75)**	5% (1.4)
Task 5	0 % (0.05)	0% (0.95)	30% (2.3)	**50% (2.65)**

B Additional results

We conducted the additional experiment on two more tasks: (Task 4) Microwave → Topknob → Switch → Hinge, (Task 5) Kettle → Topknob → Switch → Slide. The results with and without skill fine-tuning are shown in Table 4, 5.

The visualization of the policy switching in Task 2 is shown in Figure 5.

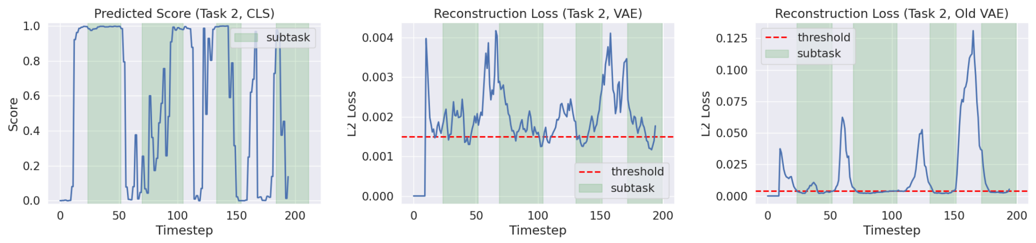


Figure 5: Visualization of policy switching for Task 2. The prediction score is plotted for CLS while the L2 loss of the reconstruction is plotted for VAE and OldVAE. The green zone shows the time for subtasks based on when the agent received the reward signal.

C Details of experiment

C.1 Environment and Dataset

The agent is a 9-DoF robot that has a 60-dimensional continuous observation of the robot itself and objects, while the action space is a 9-dimensional continuous vector. There are seven subtasks in the environment: opening the microwave, moving the kettle, turning on the bottom/top burner, turning on the light switch, opening the hinge cabinet, opening the slide cabinet.

The original dataset of Gupta et al. (2019) is available at `https://github.com/google-research/relay-policy-learning`. It is composed of 567 demonstrations of long-term tasks and each of them has four subtasks in it. It amounts to 130k timesteps in total. To remove the transition between subtasks, we use the reward in the dataset as a signal of where subtasks are performed. As shown in Table 6, we set the window size for each subtask to crop it from long-horizon demonstrations. For example, (-25, 14) in the table means that, if the agent receives the reward for a subtask at timestep t, the trajectory between $t - 25$ to $t + 14$ is to be recognized as the subtask execution. After this data preprocessing, our dataset has 2245 trajectories of subtasks that consist of 92k timesteps. It means that 30% of the transitions were removed from the dataset. Note that we ignored the subtask execution that failed to obtain the reward.

Table 6: The window size for subtask cropping. The value means the relative time from when the agent receives the reward.

SubTask	Microwave	Kettle	Bottomknob	Topknob	Switch	Slide	Hinge
Range	(-25, 14)	(-15, 34)	(-25, 24)	(-20, 19)	(-20, 9)	(-15, 24)	(-30, 9)

C.2 Hyperparameters and training details

Hyperparameters in this work are listed in Table 7. The models are represented as feedforward neural networks unless explicitly mentioned otherwise. For the parameters of FIST, we use the default parameters in the official implementation (`https://github.com/kouroshHakha/fist`). Since we use a single trajectory for fine-tuning while the original paper used ten trajectories, we set the number of iterations for fine-tuning ten times larger than the original one. Policy learning from the offline dataset took about a few hours on a single GPU. We ran 20 seeds for each experiment to calculate the average scores. In the fine-tuning of the goal-conditioned policy, we found that using only a demonstration for fine-tuning results in overfitting. Thus, we add samples from the offline dataset so that the ratio of samples from a demonstration is reduced to 2%. Note that we did not use this strategy in fine-tuning of the skill-based imitation model since we did not observe that it improved the performance.

Table 7: Hyperparameters.

Method, Model	Parameter	Value
FIST	dim of skill z	10
	skill horizon H	10
	number of hidden layers in q_ψ and skill decoder	5
	number of units per layer in q_ψ and skill decoder	128
	number of hidden layers in skill encoder	1 LSTM + 1 linear
	number of units per layer in skill encoder	128
	number of hidden layers in distance encoder	2
	number of units per layer in distance encoder	128
	skill extraction: batch size	128
	skill extraction: traning epochs	200
	skill extraction: traning epochs (fine-tuning)	500
	distance learning: batch size	1024
	distance learning: training epochs	500
	distance learning: training epochs (fine-tuning)	50
	distance learning: dim of embedding	32
GCBC	horizon range $[H_{\min}, H_{\max}]$	[5, 20]
	optimizer	Adam (lr=5e-4)
	batch size	256
	trainig step	2M
	trainig step (fine-tuning)	100k
	number of hidden layers	2
	number of units per layer	400
	activation (mid layers)	ReLU
	activation (final layer)	tanh
Eff. Classifier	horizon range $[H_{\min}, H_{\max}]$	[5, 15]
	optimizer	Adam (lr=5e-4)
	batch size	256
	trainig step	200k
	number of hidden layers	2
	number of units per layer	400
	activation (mid layers)	ReLU
	activation (final layer)	sigmoid
Switch	threshold α for OldVAE	4e-3
	threshold α for VAE	1.5e-3