Language-guided Skill Learning with Temporal Variational Inference

Haotian Fu [1] **Pratyusha Sharma** [2] **Elias Stengel-Eskin** [3] **George Konidaris** [1] **Nicolas Le Roux** [4 5]
Marc-Alexandre Côté [* 5] **Xingdi Yuan** [* 5]

Abstract

We present an algorithm for skill discovery from expert demonstrations. The algorithm first utilizes Large Language Models (LLMs) to propose an initial segmentation of the trajectories. Following that, a hierarchical variational inference framework incorporates the LLM-generated segmentation information to discover reusable skills by merging trajectory segments. To further control the trade-off between compression and reusability, we introduce a novel auxiliary objective based on the Minimum Description Length principle that helps guide this skill discovery process. Our results demonstrate that agents equipped with our method are able to discover skills that help accelerate learning and outperform baseline skill learning approaches on new long-horizon tasks in BabyAI, a grid world navigation environment, as well as ALFRED, a household simulation environment.[1]

1. Introduction

A major issue that makes Reinforcement Learning (RL) hard to use for long-horizon interaction tasks is its sample inefficiency. Conventional Deep RL algorithms explore and learn task-specific policies from scratch, which can require over 10M samples to train just one Atari game (Mnih et al., 2015; Hessel et al., 2018). In contrast, humans can play well after just 20 episodes. Humans have a strong set of priors that helps us efficiently adapt to new tasks. Some recent work has made attempts to introduce such priors for embodied agents as well, like constructing world-models (Hafner et al., 2023; Hansen et al., 2023), a goal/reward-conditioned policy (Reed et al., 2022; Lee et al., 2022), or a toolbox of extracted skills/options (Pertsch et al., 2020; Liu et al., 2022; Wang et al., 2023). Skills are temporally-extended policies (Sutton et al., 1999); each skill is executed for a certain number of steps until switching to another skill. While the observation that pre-specifying the set of skills enables effective generalization to longer-horizon tasks, an effective algorithm to autonomously discover the set of domain-specific skills to enable long-horizon planning remains challenging. Our goal is to enable the discovery of reusable skills from a dataset of expert demonstrations (i.e., trajectories) of an agent performing various complex tasks, and use these skills to solve new complex tasks more efficiently.

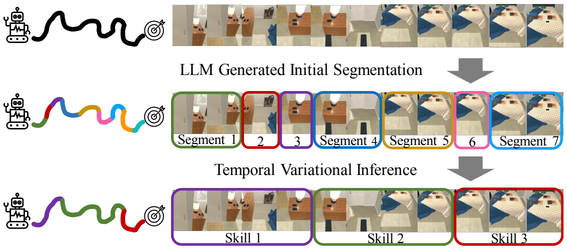

LLM Generated Initial Segmentation
Segment 1
2
3
Segment 4
Segment 5
6
Segment 7
Temporal Variational Inference
Skill 1
Skill 2
Skill 3

Figure 1. The trajectory segmentation and merging procedure.

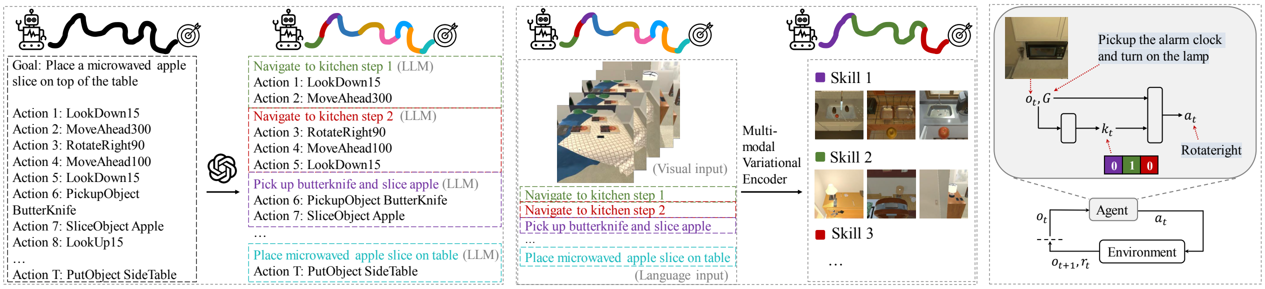
Many recent approaches learn skills from expert demonstrations by first slicing the trajectories into segments and merging them with a latent skill representation, and then reconstruct the trajectories from these latent variables (Shankar & Gupta, 2020; Shankar et al., 2020; Kipf et al., 2019). Such methods in practice often fall into two categories of "local optima": 1. every single step in one trajectory is segmented as one skill and the learned skill space is like a continuous resemblance of the original action space. 2. the agent directly marks the whole trajectory as one skill and does not do any segmentation. In the first case, it is hard to accelerate learning on new tasks as the skills are basically the original actions. For the second case, the learned skills may perfectly reconstruct the training trajectories but cannot generalize on a new task as long as it is different from the training ones.

Why is it hard to balance between compression and reusability when learning from long trajectories? We believe that this is because the search space for trajectory segmentation is too large especially when the task horizon is long. The number of ways to segment a sequence grows exponentially

[*] Equal Advising [1]Brown University [2]MIT [3]University of North Carolina, Chapel Hill [4]Mila [5]Microsoft Research. Correspondence to: Haotian Fu <hfu7@cs.brown.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] Code and videos can be found at https://language-skill-discovery.github.io/.


Goal: Place a microwaved apple slice on top of the table
Action 1: LookDown15
Action 2: MoveAhead300
Action 3: RotateRight90
Action 4: MoveAhead100
Action 5: LookDown15
Action 6: PickupObject ButterKnife
Action 7: SliceObject Apple
Action 8: LookUp15
…
Action T: PutObject SideTable
Navigate to kitchen step 1 (LLM)
Action 1: LookDown15
Action 2: MoveAhead300
Navigate to kitchen step 2 (LLM)
Action 3: RotateRight90
Action 4: MoveAhead100
Action 5: LookDown15
Pick up butterknife and slice apple (LLM)
Action 6: PickupObject ButterKnife
Action 7: SliceObject Apple
…
Place microwaved apple slice on table (LLM)
Action T: PutObject SideTable
(Visual input)
Navigate to kitchen step 1
Navigate to kitchen step 2
Pick up butterknife and slice apple
…
Place microwaved apple slice on table
(Language input)
Multi-modal Variational Encoder
Skill 1
Skill 2
Skill 3
…
Pickup the alarm clock and turn on the lamp
o_t, G
k_t
a_t
Rotateright
0 1 0
Agent
o_t
a_t
Environment
o_{t+1}, r_t

Step 1: Initial Segmentation

Step 2: Temporal Variational Inference

Step 3: Online Hierarchical RL

Figure 2. Overall framework of **LAST**. **Step 1**: given a dataset of expert demonstrations, we query an LLM (only using the goal and actions as input) for an initial segmentation and a language description for each segment. **Step 2**: temporal variational inference takes in multi-modal data as input to improve upon the segmentation by merging different subsequences into skills. **Step 3**: online hierarchical RL on new tasks leveraging the learned skills which can greatly shorten the task horizon and help the agent efficiently learn on new tasks.

with respect to the horizon, which will make the learning difficult and prone to poor local optima. We propose to narrow down the search space via **Initial Segmentation** and **Language-Augmented Temporal Variational Inference**. As shown in Fig. 1, for initial segmentation, we use an LLM to segment each trajectory into many short subsequences and generate corresponding language annotations. Then, we propose a temporal variational inference framework that can improve upon the initial segmentation by merging short subsequences into longer ones and integrate them into reusable skills. The search space is gradually narrowed down via 1) forcing the agent to never split a segment generated by the LLM—only merge them into larger ones; 2) maximizing the trajectory reconstruction likelihood and choosing skills that provide the shortest descriptions of the trajectory. The two steps together greatly reduce the number of possible ways to segment a sequence while still maintaining the reusability of the learned skills.

Specifically, we propose a novel algorithm that can effectively merge semantic priors from language models with temporal variational inference to discover skills. Our contributions are: 1) We propose a method that leverages a pretrained LLM to generate an initial segmentation of the expert trajectories, which greatly decreases the search space for skill segmentation and provides additional learning signals for the following skill discovery steps. 2) We propose a hierarchical variational inference framework that can incorporate the generated language supervision and discover skills on top of initial trajectory segmentations. 3) We augment the temporal variational inference process with a novel auxiliary training objective following Minimum Description Length Principle, which further helps compression and discover reusable skills. 4) The proposed online hierarchical RL framework enables the agent to quickly adapt to new long-horizon tasks with the learned skills in BabyAI (Chevalier-Boisvert et al., 2019) and ALFRED (Shridhar et al., 2020a), the latter of which is a highly complex household simulation environment with multimodal inputs and long-horizon tasks.

2. Related Work

The proposed approach improves over previous work in two ways: First, methods that use language/language model priors, inadvertently restrict the representation of the skills to natural language alone—this restriction doesn't apply to our framework. Second, most methods that use variational inference to discover skills from demonstration do not utilize any external semantic guidance from language models, which makes the skill discovery procedure prone to overfitting to short-horizon plans and the model fails to learn skills that generalize well.

Skill learning from demonstrations: We consider the problem of learning skills from demonstrations (Niekum et al., 2012; 2013; Meier et al., 2011; Sharma et al., 2019; Krishnan et al., 2017; Murali et al., 2016; Fox et al., 2017), for tasks where the action space is made of strings. Shankar & Gupta (2020) propose a framework to use Temporal Variational Inference to learn skills from demonstrations. However, it is limited to low-dimensional state space and Jiang et al. (2022) shows that such methods (Kim et al., 2019) are under-specified and can fail to learn reusable skills. Additionally, they propose an auxiliary training objective that encourages compression. However, we find that in practice when the tasks are much harder and require long-horizon reasoning, despite this training objective the agent may still mark the whole trajectory as one skill and does not do any segmentation. In this paper, we argue that besides information theoretical training objectives, more supervision / prior knowledge needs to be considered to learn reusable skills for complex long-horizon planning tasks. Methods for learning skills from multiple tasks (Heess et al., 2016; Riedmiller et al., 2018; Hausman et al., 2018; Fu et al., 2023) and learning to compose the said skills have also been discussed by a rich body of literature (Konidaris & Barto, 2009; Konidaris et al., 2012; Kipf et al., 2019; Bagaria et al., 2021). Recent works also consider skill learning from offline trajectories (Ajay et al., 2021; Rao et al., 2022; Pertsch et al., 2020; Xu et al., 2022), containing a large amount of sub-optimal data. In this work, we employ a two-level hierarchical RL framework with a frozen low-level policy.

Language and interaction: Several studies have focused on utilizing natural language and language models to assist with planning for long-horizon tasks. While some studies have used natural language to represent intermediate planning steps or skills (Branavan et al., 2009; Chen & Mooney, 2011; Frome et al., 2013; Andreas et al., 2017), others have used language to map directly to sequences of abstract actions (Chen & Mooney, 2011; Tellex et al., 2011; Misra et al., 2017; Anderson et al., 2018; Shridhar et al., 2020b) or plan over a known domain model of the environment (Song et al., 2023; Arora & Kambhampati, 2023; Silver et al., 2024). Several of these approaches are also equipped with the use of language models as a distribution over valid sequences of skills (Sharma et al., 2022; Singh et al., 2023; Liu et al., 2023) and actions themselves (Huang et al., 2022). However, all of these methods require either specifying the planning domain, the set of skills, and/or precise abstract action steps in advance. They do not construct the skill set itself, a crucial aspect of our proposal.

Recent studies have proposed using natural language to guide the discovery of reusable skills (Sharma et al., 2022) from unparsed demonstrations. However, they have assumed that the skills themselves must be represented with natural language . Although powerful in “realistic” domains, natural language as the choice of representation can become restrictive when the optimal skill set is hard to specify using natural language alone. In this paper, we use language (and language models) to guide the discovery of skills, while lifting the restriction. BOSS (Zhang et al., 2023b) starts from a pretrained initial skill library and focuses on learning new skills through online interactions by chaining the existing skills which are guided by LLM directly. SPRINT (Zhang et al., 2023a) proposes to use LLMs to relabel robot trajectories and leverage offline reinforcement learning methods to do skill chaining. Other recent work simultaneously propose the domain model and learn libraries of action operators defined as code-based policies guided by LLMs (Wang et al., 2023; Liu et al., 2023; Wong et al., 2023) as a way to learn hierarchical action representations. However, these approaches require a planner and multiple rounds of interaction in the environment thereafter to iteratively verify the correctness and extensibility of the inferred skills.

3. Problem Setup

We consider learning problems where the agent needs to use the experience from expert demonstrations to quickly solve new RL tasks. We formalize these problems as Partially Observable Markov Decision Processes (POMDPs), defined by a tuple $(S, A, O, \Omega, \mathcal{T}, R)$. We use S to denote the state space, A for the action space, O for the observation space, $\Omega : S \to O$ for the observation function, $\mathcal{T} : S \times A \to S$ for the transition function, and R as the reward function. The dataset is a set of M goal-conditioned trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^M$, and $\tau_i = \{G_i, o_{i1}, a_{i1}, \cdots, o_{iT_i}\}$, where G denotes the task goal, $o \in O$ denotes the observation, $a \in A$ denotes the actions, and T is the length of the trajectory. We assume every action has a semantic meaning described by language. The trajectories are collected from *multiple* tasks so each trajectory τ_i may have a different reward function.

In this work, we assume that our policies are mixtures of time-limited, semantically meaningful, sub-policies which we call *skills*, and that these skills will be shared across tasks. Formally, we introduce two new time-dependent variables: k_t is the skill used at time t and β_t is a *skill-switching variable* meant to identify when we move from one skill to another, i.e. $k_t = k_{t-1}$ when $\beta_t = 0$. Denoting $\phi = \{\beta_t, k_t\}_{t=1}^T$, we have $p(\tau) = \sum_\phi p(\phi)p(\tau \mid \phi)$ as joint probability over trajectory and skills. $p(\tau \mid \phi)$ is the skill (and goal) conditioned policy, and $p(\phi)$ is the high-level policy over the skills. Correctly approximating the posterior distribution $p(\phi \mid \tau)$ over skills given the trajectories will be critical to extract the skills which we will be able to reuse for new tasks.

4. Language-guided Skill Learning with Temporal Variational Inference

We shall now describe our method **LA**nguage-guided **S**kill Learning with **T**emporal Variational Inference (**LAST**), which jointly learns the mixture model $p(\tau) = \sum_\phi p(\phi)p(\tau \mid \phi)$ and approximates the posterior $p(\phi \mid \tau)$. This method is split into three steps which are shown in Fig. 2. First, given a dataset of expert demonstrations collected from multiple tasks with different goals, we use an LLM to generate an initial segmentation for these trajectories (§4.1). Then, we propose a temporal variational inference framework that aims to improve the segmentation and merge different pieces into skills (§4.2). We also present an auxiliary training objective following the Minimum Description Length principle that helps compress the learned skills (§4.3). Finally we introduce how the learned skills can be used to quickly solve new tasks (§4.5).

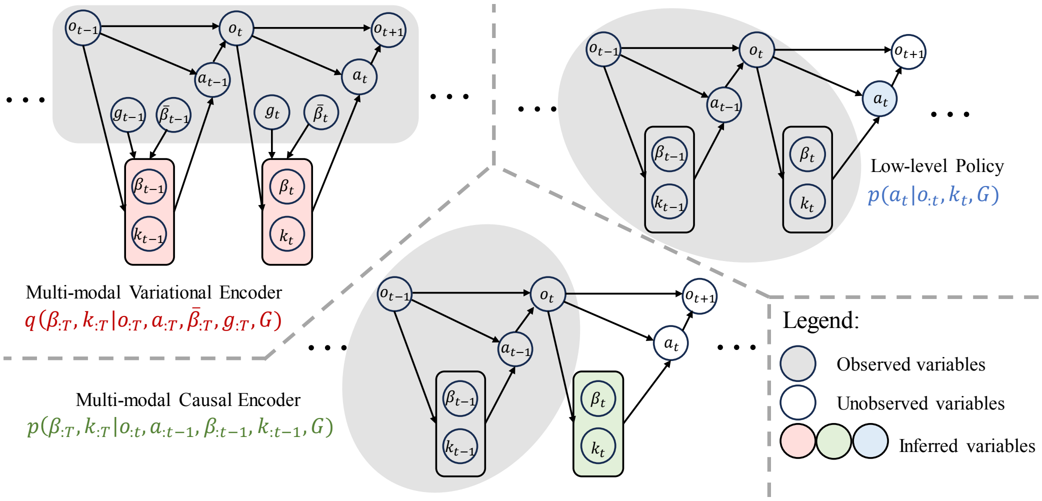

o_{t-1}
o_t
o_{t+1}
a_{t-1}
a_t
g_{t-1}
$\bar{\beta}_{t-1}$
g_t
$\bar{\beta}_t$
β_{t-1}
k_{t-1}
β_t
k_t
Multi-modal Variational Encoder
$q(\beta_{:T}, k_{:T} | o_{:T}, a_{:T}, \bar{\beta}_{:T}, g_{:T}, G)$
Low-level Policy
$p(a_t | o_{:t}, k_t, G)$
Multi-modal Causal Encoder
$p(\beta_{:T}, k_{:T} | o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G)$
Legend:
Observed variables
Unobserved variables
Inferred variables

Figure 3. An overview of the probabilistic graphical model underlying LAST. Distributions are labeled with the same colors in Eqn. 3. We use $q(\beta_{:T}, k_{:T} \mid \cdot)$ as the approximate posterior which has access to all the information we have. $p(\beta_{:T}, k_{:T} \mid \cdot)$ is the true high-level policy that is trained to mimic $q(\beta_{:T}, k_{:T} \mid \cdot)$. $p(a_t \mid o_{:t}, k_t, G)$ denotes the skill-goal conditioned policy.

4.1. Initial Segmentation with LLMs

The core of LAST is to segment trajectories of T actions into N subsequences. One could imagine directly learning the graphical model. However, there are an exponential number of ways to segment a sequence of T actions into N subsequences, which will make the learning difficult and prone to poor local optima. Besides, the posterior $p(\phi \mid \tau)$ will be difficult to approximate. To see this, note that, given a trajectory τ, the variables β_t are heavily correlated since switching skills at timestep t affects the probability of switching skills at nearby timesteps. Hence, approximating the posterior with, e.g., a factorial distribution over the β_t will be very inaccurate and so will be our skill extraction procedure.

To address this, we shall thus first perform a segmentation of each trajectory into subsequences by using an LLM. There is a tradeoff to achieve. On the one hand, having short subsequences will only mildly simplify the posterior distribution. On the other hand, having few, long subsequences will make the posterior easier to approximate but at the expense of a loss of capacity in our generative model, especially if the segmentation is inaccurate. As we wish to make few assumptions about the quality of the underlying LLM, we only use it to find **short segments, i.e., 1 to 5 actions each**. This constraint is introduced by modifying the prompt to the LLM, as shown in Appendix B: in the prompt, we tell the LLM that the number of actions assigned to each skill should not exceed 5 but should be larger than 1.

As shown in Fig. 2 (step 1), at each iteration i, given a trajectory τ, we prompt an LLM with the concatenation of the goal description G and the action sequences $\{a_1, a_2, \cdots\}$, where each a_t is described by language given the definition of the environment's action space. We ask the LLM to return the segmentation of τ and a language annotation describing each segment. We do not include the (visual) observations as input here since it would be expensive to use either the raw image or running an image captioning system. Our system does not require an optimal segmentation to start with. For instance, given a task goal such as **`place a microwaved apple slice on top of the black table`** and a sequence of 44 actions, the LLM might segment that trajectory into 13 pieces, each of which is associated with a language annotation (e.g., **`Navigate to kitchen step 1`**). We show a concrete example in App. A.

The output of this step is a set of enriched trajectories where, at each timestep t, we added variables $\bar{\beta}_t$ and g_t. $\bar{\beta}_t$ equals 0 when a_{t-1} and a_t belonged to the same segment, 1 otherwise. g_t denotes the language description generated by the LLM (**`Navigate to kitchen step 1`**) for the segment that contains a_t (g_t remains the same for the whole segment). The language annotations g_t generated by the LLM are also an important conditioning information for our model to do the inference during the temporal variational inference phase. We call the first variable $\bar{\beta}$, not β, because it is not the final segmentation of the trajectory.

To simplify the learning of the graphical model, we shall make the assumption that a skill will never split a segment generated by the LLM, i.e. there will only be a merging of these initial segments into larger ones. In other words, we have $(\bar{\beta}_t = 0) \implies (\beta_t = 0)$. Therefore, $\sum_t \bar{\beta}_t \geq \sum_t \beta_t$.

4.2. Skill Discovery with Temporal Variational Inference

Our goal is now to improve the initial segmentation obtained by the LLM, and merge the resulting short subsequences into longer ones that can be reused to solve new tasks efficiently, which corresponds to step 2 in Fig. 2. More specifically, we wish to jointly learn:

- $\pi(a_t \mid o_{:t}, k_t, G)$ for each possible value of k_t and G, our goal and skill dependent policies;
- $p(\phi \mid \tau)$, our posterior distribution over the skills and skill-switching variable, which will allow us to do skill extraction.

Note that as we add the generated intermediate signals ($\bar{\beta}_t$ and g_t) from the first step to the trajectory τ, the distribution can be further denoted as $p(\beta_{:T}, k_{:T} \mid o_{:T}, a_{:T}, \bar{\beta}_{:T}, g_{:T}, G)$.

Since we know that $\beta_t = 0$ every time $\bar{\beta}_t = 0$, we only have to compute the posterior for the timesteps t such that $\bar{\beta}_t = 1$. Further, because these timesteps are less likely to be consecutive, we can hope that the posterior will be easier to approximate within our function class.

We will learn both our mixture model and approximate posterior using temporal variational inference (Kingma & Welling, 2014). We show the probabilistic graphical model underlying LAST in Fig. 3. We start by lower bounding the probability of a given trajectory by:

$$\log p(\tau) \geq \sum_{\phi} q(\phi \mid \tau) \log \frac{p(\tau, \phi)}{q(\phi \mid \tau)} \ , \tag{1}$$

with q our approximate posterior. We will use q for a factorial distribution, both over timesteps and over β and k. We factor the variational distribution as:

$$q(\beta_{:T}, k_{:T} \mid o_{:T}, a_{:T}, \bar{\beta}_{:T}, g_{:T}, G) =$$
$$\textstyle\prod_{t=1}^{T} q(\beta_t \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G) q(k_t \mid o_{:T}, a_{:T}, \beta_t, k_{t-1}, g_{:T}, G), \tag{2}$$

Due to space limit, we show the factorized form of $p(\tau, \phi)$ in App. D, Eqn. 9. Substituting the corresponding terms in Eqn. 1 with Eqn. 9 and Eqn. 2, we get the following objective:

$$\begin{aligned} J(\theta) \approx & \mathbb{E}_{q_\theta(\phi \mid \tau)} \sum_{t=1}^{T} \log \pi_\theta(a_t \mid o_{:t}, k_t, G) \\ & - \sum_{t=1}^{T} \Big\{ \mathrm{KL}[q_\theta(\beta_t \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G) \| \\ & \qquad p_\theta(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G)] \\ & \quad + \mathrm{KL}[q_\theta(k_t \mid o_{:T}, a_{:T}, \beta_t, k_{t-1}, g_{:T}, G) \| \\ & \qquad p_\theta(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)] \Big\}, \end{aligned} \tag{3}$$

where we parameterize all models with θ. As we will introduce below, we use the same transformer to model the distributions highlighted with the same color. Note that while both p_θ and q_θ infer the distribution over β_t and k_t; q_θ is conditioned on the whole trajectory (history and future) and the generated language annotations; p_θ is only conditioned on the history. We add such causal constraints to ensure the inference policy can be used online where only the information till the current step is accessible.

Intuitively, we want q_θ to perform the best inference of the skills by conditioning on all the information we have. Meanwhile, we want p_θ , the true high-level policy to be used on new tasks, to mimic the results of q_θ without the generated language annotations and future information, because both of which are absent during online testing. The first term of $J(\theta)$ encourages q_θ to generate proper skill-switching variables β_t and skills k_t such that we can train a policy π to accurately predict the actions conditioned on these latent variables. The other two KL-divergence terms serve two purposes. First, they encourage the inference p_θ of β_t and k_t given only the history to be as close as possible to the inference q_θ given all the information; second, they serve as regularization terms for q_θ to avoid overfitting.

After this temporal variational inference training stage, we have obtained a low-level policy π_θ that predicts actions given the observations and a specific skill, and a high-level policy p_θ that predicts the skill given the information available at the current timestep.

Constraints on inferring β_t. We made the assumption that no further segmentation is needed after step 1 and the agent only needs to merge these segments. To ensure this is met in practice, we sample β_t and k_t with the following process in step 2:

$$\begin{aligned} & \beta_t \sim \bar{\beta}_t q_\theta(\beta_t \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G), \\ & k_t \sim \beta_t q_\theta(k_t \mid o_{:T}, a_{:T}, \beta_t, g_{:T}, G) + (1 - \beta_t)\delta(k_t == k_{t-1}), \end{aligned} \tag{4}$$

where δ denotes the delta function. The same strategy applies to p_θ as well. This greatly simplifies the inference problem and enables the agent to efficiently discover the reusable skills. However, it also requires the segmentations made in step 1 to be maximally detailed, so that we do not need to further break down the subsequences in step 2.

4.3. Minimum Description Length for Skills

While jointly learning the posterior distribution and skill-conditioned policies, we would like to merge the resulting subsequences from step 1 into longer and reusable ones. That is, we would like to further compress the results and decrease the number of subsequences of each trajectory. Inspired by Jiang et al. (2022), we introduce an auxiliary compression objective, following the Minimum Description Length (MDL) Principle (Rissanen, 1978).

MDL favors the simplest model that accurately fits the given data, which provides guidance for finding the common structures inside the data and further compress them. In this context, MDL suggests choosing the skills that provides the shortest description of the trajectories. We consider the two-part form of MDL:

$$\phi^* = \mathrm{argmin}_\phi L(\mathcal{D} \mid \phi) + L(\phi), \tag{5}$$

where $L(\mathcal{D} \mid \phi)$ denotes the number of bits required to encode the trajectories $\mathcal{D}$ given the model ϕ, and $L(\phi)$ is the number of bits required to encode the model ϕ itself.

Computing $L(\phi)$ generally requires approximating the complexity of neural networks. In this paper, we simply add one constraint to the training process as an approximation: we set a maximum number of skills for the skill library so that the number of skills used to encode all the trajectories cannot exceed this number. After training we only keep the skills that are used frequently enough (over some threshold) and use them for online testing.

We focus on minimizing the first term $L(\mathcal{D} \mid \phi)$. Recall that we use $\phi = \{\beta_t, k_t\}_{t=1}^{T}$ to encode the trajectory $\tau = \{G, o_t, a_t, \bar{\beta}_t, g_t\}_{t=1}^{T}$. According to optimal code length theory (Cover & Thomas, 2001), the expected number of bits of code generated by $p(k)$ is: $\mathbb{E}_k[-\log p(k)]$. Applying the formulation defined in § 4.2, we can derive the following objective (see Appendix F for detailed derivations):

$$\begin{aligned}\mathcal{L}_{\text{MDL}}(\theta) = -\sum_t \mathbb{E}_k \log[q_t(k \mid \cdot) q_t(\beta_t = 1 \mid \cdot) \\ + q_t(\beta_t = 0 \mid \cdot)\mathbb{1}(k == k_{t-1})],\end{aligned} \tag{6}$$

where $q_t(k \mid \cdot)$ and $q_t(\beta_t \mid \cdot)$ refer to the variational posterior (highlighted in red in Eqn. 3). Intuitively, by minimizing this objective, the agent is encouraged to infer fewer skills for each trajectory (increase $\sum_t q_t(\beta_t = 0 \mid \cdot)$), as well as to increase the average confidence for choosing the skill (decrease $-\mathbb{E}_k \log q_t(k \mid \cdot)$) when it has to. A discussion of the difference between the proposed objective and the objective proposed in Jiang et al. (2022) can be found in App. E. In general, our objective function is able to more accurately reflect the skill switches's influence on the code length and also increase trainability, since our formulation and derivation do not ignore the effect of the skill k_{t-1} chosen at last time step $t-1$.

The overall training objective for step 2 is:

$$\max_\theta J(\theta) - \lambda L_{\text{MDL}}(\theta), \tag{7}$$

where λ is a hyperparameter that controls the weight of the auxiliary compression objective.

4.4. Practical Implementation and Model Architecture

We show the model architectures that implement the three distributions in App. Fig. 6. As an overview, we use three transformers to parameterize the three distributions in Eqn. 3 (in three colors) respectively. The Multi-modal Variational Encoder approximates $q(\beta, k \mid \cdot)$, which has access to all information in each trajectory. The Multi-modal Causal Encoder approximates $p(\beta, k \mid \cdot)$, which has access to information only up to the current step (implemented using a causal transformer). The Low-level Policy is also a causal transformer that approximates $\pi(a \mid o, k, G)$. We use individual pretrained encoders for the multi-modal input, and then train a linear encoder for each modality to map the inputs into the same embedding space.

We model the skills k as discrete variables and set the maximum number of skills to be K. To stabilize the optimization with the joint objective in Eqn. 7, we perform a warm-up stage at the beginning of training. Specifically, for a certain amount of episodes, we set $\lambda = 0$ which stops the gradients from the MDL loss to q_θ in Eqn. 3. We have this pretraining stage to prevent uninformative gradients from q at the beginning of training, and also prevent from the model quickly converge to inferring the whole trajectory as a single skill because of the gradients from the MDL term.

4.5. Online Hierarchical RL

Now we introduce how we use the learned skills for online hierarchical RL. Following prior skill learning approaches, we **freeze** the low-level policy $\pi(a \mid \cdot)$ and train a high-level control policy that directly outputs skills k. Changing the action space from the primitive action space to the learned skill space yields efficient adaptation as the skills are temporally-extended, which can both shorten the task horizon and perform structural exploration.

Specifically, we let the agent learn a high-level control policy $\pi_\psi(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)$ and termination condition $\pi_\psi(\beta_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t-1}, G)$ by fine-tuning on $p(k_t \mid \cdot)$ and $p(\beta_t \mid \cdot)$. We train the policy by maximizing the following objective:

$$\begin{aligned}&\mathcal{J}(\psi) = \mathbb{E}_{\pi_\psi}[\sum_t \gamma^t (r_t \\ &- \alpha_1 \text{KL}(\pi_\psi(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G) \| p(k_t \mid \cdot)) \\ &- \alpha_2 \text{KL}(\pi_\psi(\beta_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t-1}, G) \| p(\beta_t \mid \cdot)))],\end{aligned} \tag{8}$$

where we augment the standard maximizing cumulative return objective with additional KL regularization between the high-level policy and the offline pretrained distributions over the skills and termination conditions. This is to ensure the high-level policy remains close to the pre-learned prior over the skills. We adopt Soft Actor-Critic (SAC) (Haarnoja et al., 2018) with Gumbel-Softmax (Jang et al., 2017) (for the categorical action distribution) to train the agent to maximize the cumulative return. During sampling, at the first step of each episode, the agent will first choose a skill k_1 according to $\pi_\psi(k_1 \mid \cdot)$, using the skill to interact with the environment through $\pi_\theta(a \mid \cdot)$ until the termination condition $\beta_t = 1$ is met, and then switches to the next skill.

5. Experiments

Our experiments focus on, 1. Can LAST discover distinct and semantically meaningful skills from a set of demonstrations? 2. How well do the hierarchical policies transfer to new tasks (zero-shot)? 3. Do the learned skills help accelerate learning on downstream tasks? 4. How do components of LAST contribute to its overall performance?

5.1. Experimental Setup

BabyAI (Chevalier-Boisvert et al., 2019) is an environment where an agent navigates and interacts in a grid world to achieve a goal described in language (e.g., `open a red door and go to the ball on your left`). For our experiments, we use the environment's symbolic partially observable state space, where the state vector describes the type and color of each grid cell. At each time step, the agent can choose from 6 actions including pickup/drop/toggle an object. The BabyAI dataset contains expert demonstrations collected from 40 different task types of varying levels of difficulty. We uniformly at random sample 100 trajectories from each task type perform LAST. Note that compared to previous imitation learning work on BabyAI, our setting is much harder in a few key ways, 1) We sample a very small subset of trajectories to train our model unlike previous work often requiring up to 1M trajectories. 2) We use LAST on data from multiple tasks jointly. This is in contrast to prior work, where each task type requires learning its own separate policy.

ALFRED (Shridhar et al., 2020a) is a complex environment based on the AI2-THOR (Kolve et al., 2017) simulator and is a domain where an agent is required to perform diverse household tasks following a goal described by natural language. The environment itself contains 100+ floorplans and 100+ objects to interact with. The observation space consists of the agent's egocentric view (image) of the current environment. The action space consists of 12 discrete action types (e.g., `move forward`, `turn right`, `toggle`) and 82 discrete object types. At each time step the agent needs to choose an action type along with an object type to interact with the environment. For the ALFRED dataset, we follow the settings in Pashevich et al. (2021) where the training dataset consists of more than 20000 trajectories. Each trajectory includes a goal description and a sequence of observations and actions. Note that different from many previous work on ALFRED (especially the leaderboard methods) **we do not use the step-by-step instructions provided by environment (neither during training nor testing)**, which gives us a standard goal-conditioned RL setting and is potentially more practical as instructions annotated by humans could be expensive. We want to highlight that the goal of this work is not to directly compare to systems on the ALFRED leaderboard in terms of overall performance, rather, we adopt ALFRED as a challenging testbed demonstrating our proposed skill discovery method.

We use GPT-4 as the LLM to generate the initial segmentation in all our experiments[2]. In ALFRED, we use pretrained T5 encoder (Raffel et al., 2020) and Faster R-CNN encoder (Ren et al., 2015) to preprocess the image and language data. Once again, we do not aim to compare various LLMs and encoders, we adopt the strongest systems off-the-shelf and focus on our system described in § 4, which is presumably orthogonal from specific model choice.

Table 1. Online (zero-shot) task success rate comparison on BabyAI (**top**) and ALFRED (**bottom**). Note that the subtasks in ALFRED have short horizons (i.e., subgoals of each individual task which requires significantly fewer steps to reach). In comparison, in the downstream task learning setting, the agent is tested on six individual tasks with the full length.

Model	BossLevel	MiniBoss	Synth	SynthLoc	SynthSeq	Average
BC	25	23	**53**	48	20	34
LOVE	27	31	40	63	25	37
LISA	27	26	49	49	27	36
LAST	**36**	**34**	48	**66**	**32**	**43**

Model	Avg	Clean	Cool	Heat	Pick	Put	Slice	Toggle	GoTo
BC	45	66	31	36	57	47	38	32	41
$(SL)^3$	34	68	82	75	50	45	**55**	32	15
seq2seq	17	16	33	64	20	15	25	13	14
seq2seq2seq	26	15	69	58	29	42	50	32	15
seq2seq	17	16	33	64	20	15	25	13	14
LOVE	47	76	**92**	41	58	57	51	28	38
LISA	46	73	82	21	**66**	49	35	27	38
LAST	**52**	**82**	65	**82**	63	**53**	48	**37**	**44**

5.2. Zero-shot Transfer

We first study the zero-shot transfer performance on BabyAI and ALFRED. For BabyAI, we compare with standard Behavior Cloning (BC) augmented with some inductive bias for object predictions as we explained in appendix C; LOVE (Jiang et al., 2022), the state-of-the-art method for learning skills from demonstrations (without language) which first proposes the compression objective; and LISA (Garg et al., 2022), a language-based skill learning method. We further equip LISA with our network architectures and provide it with the initial segmentation results. It can thus be seen as an simpler version of our algorithm without temporal variational inference and directly mapping the initial segmentation results to skills. We do not compare to (Shankar & Gupta, 2020) since it is similar to LOVE but without the compression term. We implement all the baselines using the same transformer structure we use for LAST to make fair comparison. For ALFRED, since our setting does not assume the access to the step-by-step human-annotated instructions, the best performing method we are able to find in a similar setting is $(SL)^3$ (Sharma et al., 2022). We also compare against seq2seq and seq2seq2seq, both of which are methods provided by the original paper of ALFRED in the same setting (Shridhar et al., 2020a), as well

[2] As an alternative, we have used open-sourced LMs (e.g., Phi-2 and Mistral 7B) to generate the initial segmentation. However, 1) they support insufficient context length and 2) they often fail to follow instructions on generating output with specific format.

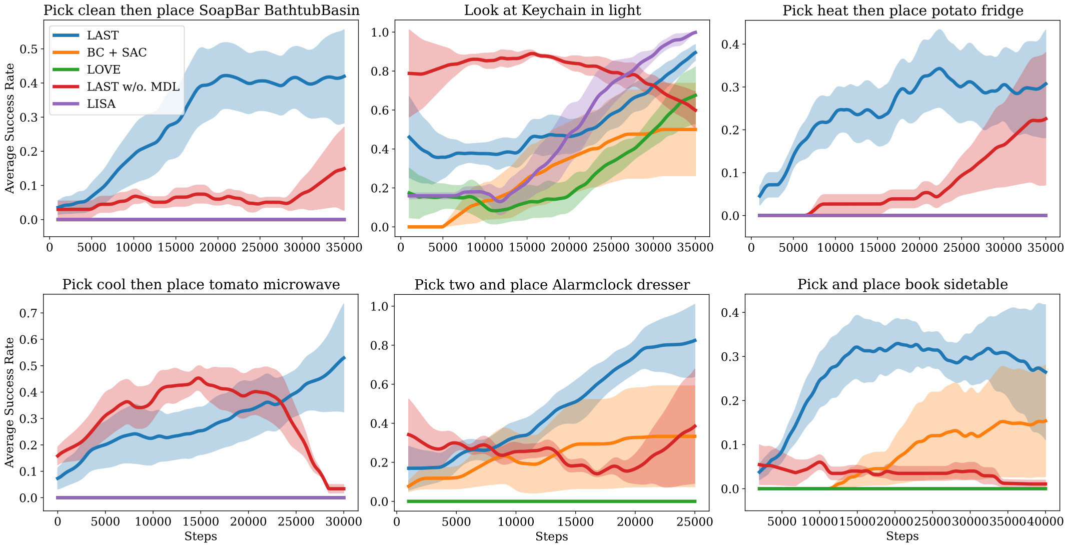


Figure 4. Comparison results of our method LAST against other baselines in six downstream tasks of ALFRED. We plot average success rate v.s. timesteps with 95% confidence interval error bar ($\geq$ 5 seeds).

as BC (modified using the same policy network architecture of LAST).

We show the results in Table 1 (top: BabyAI, bottom: ALFRED). On both domains, LAST can achieve the highest average success rate and outperforms the baselines in most tasks, indicating that the skills accompanied with the hierarchical policies our method learns can be well transferred on diverse new tasks. We notice that the largest performance gain of LAST in ALFRED comes from the ”GoTo” tasks, which is the navigation tasks that previous language-based methods find hard to solve (Sharma et al., 2022).

5.3. Learning on Downstream Tasks

We further test whether the learned skills can facilitate the agent’s learning on different downstream tasks. We randomly pick one long-horizon task from each of the six task categories in ALFRED and run online RL fine-tuning with SAC (Haarnoja et al., 2018). Note that learning from scratch using SAC is not able to achieve any success in these six tasks. We compare our method with the same set of methods introduced in the last section, as well as an additional baseline that does not include the MDL term as the auxiliary training loss during temporal variational inference.

As shown in Fig. 4, LAST outperforms baselines in 5 out of 6 tasks, demonstrating that the learned skills can facilitate downstream task learning for multiple types of tasks. In *Pick clean*, *Pick heat* and *Pick cool* tasks, where the task horizon ($\sim$ 200) is significantly larger, only LAST-based methods can achieve positive success rate. Other skill discovery baselines like LOVE and LISA perform comparably only on *Look at Keychain in Light* task, which has the shortest task horizon (< 50) among the six tasks. The results indicate that both the initial segmentation phase (compared with LOVE) and the temporal variational inference phase (compared with LISA) are important to learn skills that can solve complex tasks. We further note that without the MDL term, LAST is still able to learn skills that can help downstream learning compared to the other baselines, however, the absence of the MDL term decreases LAST’s performance.

Ablation study results can be found in App. F, where we further investigate the influence of the initial segmentation, the generated language annotation g, MDL objective, replacing initial segmentation with the one provided by the dataset’s human annotated instructions and the constraints on inferring β_t (Eqn. 4).

5.4. Qualitative Evaluation of the Learned Skills

Videos of the learned skills can be found at language-skill-discovery.github.io. We aim to gain a better understanding on the skills discovered by LAST by investigating the sub-trajectories segmented by it across three runs with different random seeds. We notice that there are five skills being discovered in all three runs. We manually identify the semantics of these five skills by looking at their corresponding sub-trajectories, we list them in Fig. 5 (left). To better demonstrate this finding, in Fig. 5 (left), we show a trajectory, segmented and labelled by LAST, with our interpretation of the skills. In this example, the trajectory has been split into six sub-trajectories of varying length. In which, for example, the sequence of actions to take a potato from the sink has been labeled as the take from open receptacle skill, and thus it may share the skill representation with sub-trajectories that take object from a table in another trajectory.

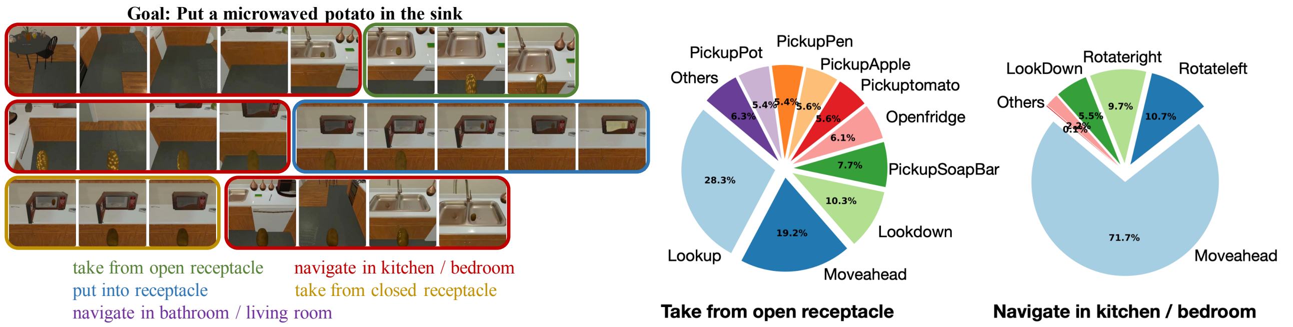


Figure 5. **Left**: LAST's skill segmentation for task `Put a microwaved potato in the sink`. **Right**: Example of discovered skills and their most-commonly used actions. We show more trajectories, segmentations, and common skills in App. Fig. 9 and 10.

In Fig. 5 (right), we statistically show the proportion of each ALFRED action being included in the segments correspond to the take from open receptacle and navigate in kitchen / bedroom skills. For example, for skill take from open receptacle, action `Pickup` appears frequently; while in the navigate in kitchen / bedroom skill, more than 70% of the actions are `Moveahead`. We are delighted to see that in addition to being helpful quantitatively (as shown in previous subsections), the skills discovered by LAST could to some extent be interpreted by humans, we believe this is due to our design that leveraging LLMs in skill/segment initialization. We provide more example trajectory segments in App. Fig. 9. **As we mentioned before, in ALFRED we find there are five categories of skills that are always discovered by our approach across different random seeds and we include the pie charts for all of them in App. Fig. 10.** We also show the transition probability matrix between discovered skills in App. Fig. 11.

6. Discussion

In this paper, we studied the problem of learning skills from demonstrations. We propose LAST, a framework that learns reusable skills from expert trajectories by 1) querying LLMs for an initial segmentation; 2) leveraging temporal variational inference to merge subsequences into skills; 3) training with an auxiliary MDL term that further compresses the skills. We found empirically that LAST enables the agent to learn semantically meaningful skills that can help solve long-horizon complex tasks.

The proposed initial segmentation approach requires agents equipped with discrete action tokens. This is one limitation of the proposed method. However, we would like to point out that many practical applications have the same settings (e.g., policies that use API calls or tools, AI assistants that help with typical performance in a program or OS, gaming settings such as Atari where actions tend to be discretized). Besides, we should also note that the core idea of initial segmentation is to leverage the commonsense knowledge from foundation models to perform the initial segmentation. At the moment, we are limited to using Large Language Models (hence using the text description of the actions). But it is also possible to use large video foundation models to improve the initial segmentation and extend our work to other domains in the future.

Acknowledgement

This work was partially done while Haotian Fu, Pratyusha Sharma, and Elias Stengel-Eskin were interning at MSR. The authors would like to thank Sumana Basu, Ayush Jain, Yiding Jiang, Michael Littman, Vincent Micheli, Benjamin Spiegel, Tongzhou Wang, Tian Yun, Zilai Zeng, ICML anonymous reviewers, as well as other colleagues and friends at Microsoft Research and Brown University for discussions and helpful feedback. This work was supported in part by NSF grant #1955361, CAREER award #1844960 to Konidaris, and ONR grant #N00014-22-1-2592. Nicolas Le Roux is a Canada CIFAR AI Chair.

Impact Statement

We do not foresee significant societal impact resulting from our proposed method. The proposed LAST system leverages LLMs to label pre-collected trajectory datasets and then learns to merging the segments into reusable skills. Despite LAST does not directly interact with humans in any way, caution needs to be exercised if one extends our work to a setting where humans are more involved. We use pre-trained LLMs for obtaining the initial segments and their labels, this could potentially bring hallucinations into the system. Although our Temporal Variational Inference framework is designed to refine and merge the LLM-generated segments into more reusable skills, mis-information from the initial labels could nonetheless have an effect on the converged skills.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. OPAL: offline primitive discovery for accelerating offline reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Hengel, A. V. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR, 06–11 Aug 2017.

Arora, D. and Kambhampati, S. Learning and leveraging verifiers to improve planning capabilities of pre-trained language models, 2023.

Bagaria, A., Senthil, J. K., and Konidaris, G. Skill discovery for exploration and planning using deep skill graphs. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 521–531. PMLR, 2021.

Branavan, S., Chen, H., Zettlemoyer, L., and Barzilay, R. Reinforcement learning for mapping instructions to actions. In Su, K.-Y., Su, J., Wiebe, J., and Li, H. (eds.), *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Chen, D. L. and Mooney, R. Learning to interpret natural language navigation instructions from observations. In *AAAI 2011*, 2011.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. Babyai: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley, 2001.

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

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., and Mikolov, T. Devise: A deep visual-semantic embedding model. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Fu, H., Yu, S., Tiwari, S., Littman, M., and Konidaris, G. Meta-learning parameterized skills. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10461–10481. PMLR, 2023.

Garg, D., Vaidyanath, S., Kim, K., Song, J., and Ermon, S. LISA: learning interpretable skill abstractions from language. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. P. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023.

Hansen, N., Su, H., and Wang, X. TD-MPC2: scalable, robust world models for continuous control. *CoRR*, abs/2310.16828, 2023.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. A. Learning an embedding space for transferable robot skills. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T. P., Riedmiller, M. A., and Silver, D. Learning and transfer of modulated locomotor controllers. *CoRR*, abs/1610.05182, 2016.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

Ichter, B., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K., Kuang, Y., Jesmonth, S., Joshi, N. J., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as I can, not as I say: Grounding language in robotic affordances. In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2022.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Jiang, Y., Liu, E. Z., Eysenbach, B., Kolter, J. Z., and Finn, C. Learning options via compression. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Kim, T., Ahn, S., and Bengio, Y. Variational temporal abstraction. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11566–11575, 2019.

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.

Kipf, T., Li, Y., Dai, H., Zambaldi, V. F., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. W. Compile: Compositional imitation learning and execution. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3418–3428. PMLR, 2019.

Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017.

Konidaris, G. D. and Barto, A. G. Skill discovery in continuous reinforcement learning domains using skill chaining. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pp. 1015–1023. Curran Associates, Inc., 2009.

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.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. DDCO: discovery of deep continuous options for robot learning from demonstrations. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, volume 78 of *Proceedings of Machine Learning Research*, pp. 418–437. PMLR, 2017.

Lee, K., Nachum, O., Yang, M., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., and Mordatch, I. Multi-game decision transformers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. Llm+p: Empowering large language models with optimal planning proficiency, 2023.

Liu, I.-J., Yuan, X., Côté, M.-A., Oudeyer, P.-Y., and Schwing, A. Asking for knowledge (AFK): Training RL agents to query external knowledge using language. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14073–14093. PMLR, 17–23 Jul 2022.

Meier, F., Theodorou, E. A., Stulp, F., and Schaal, S. Movement segmentation using a primitive library. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pp. 3407–3412. IEEE, 2011.

Misra, D. K., Langford, J., and Artzi, Y. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Murali, A., Garg, A., Krishnan, S., Pokorny, F. T., Abbeel, P., Darrell, T., and Goldberg, K. TSC-DL: unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning. In Kragic, D., Bicchi, A., and Luca, A. D. (eds.), *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pp. 4150–4157. IEEE, 2016.

Niekum, S., Osentoski, S., Konidaris, G. D., and Barto, A. G. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5239–5246. IEEE, 2012.

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.

Pashevich, A., Schmid, C., and Sun, C. Episodic transformer for vision-and-language navigation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 15922–15932. IEEE, 2021.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. In Kober, J., Ramos, F., and Tomlin, C. J. (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Rao, D., Sadeghi, F., Hasenclever, L., Wulfmeier, M., Zambelli, M., Vezzani, G., Tirumala, D., Aytar, Y., Merel, J., Heess, N., and Hadsell, R. Learning transferable motor skills with hierarchical latent mixture policies. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Reed, S. E., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *Trans. Mach. Learn. Res.*, 2022, 2022.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 91–99, 2015.

Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4341–4350. PMLR, 2018.

Rissanen, J. Modeling by shortest data description. *Autom.*, 14(5):465–471, 1978.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8624–8633. PMLR, 2020.

Shankar, T., Tulsiani, S., Pinto, L., and Gupta, A. Discovering motor programs by recomposing demonstrations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 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.

Sharma, P., Torralba, A., and Andreas, J. Skill induction and planning with latent language. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1713–1726. Association for Computational Linguistics, 2022.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10737–10746. Computer Vision Foundation / IEEE, 2020a.

Shridhar, M., Yuan, X., Côté, M., Bisk, Y., Trischler, A., and Hausknecht, M. J. Alfworld: Aligning text and embodied environments for interactive learning. *CoRR*, abs/2010.03768, 2020b.

Silver, T., Dan, S., Srinivas, K., Tenenbaum, J., Kaelbling, L., and Katz, M. Generalized planning in PDDL domains with pretrained large language models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., and Su, Y. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

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

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A., Teller, S., and Roy, N. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *CoRR*, abs/2305.16291, 2023.

Wong, L., Mao, J., Sharma, P., Siegel, Z. S., Feng, J., Korneev, N., Tenenbaum, J. B., and Andreas, J. Learning adaptive planning representations with natural language guidance, 2023.

Xu, M., Veloso, M., and Song, S. Aspire: Adaptive skill priors for reinforcement learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Zhang, J., Pertsch, K., Yang, J., and Lim, J. J. Minimum description length skills for accelerated reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop-ICLR*, volume 2021, 2021.

Zhang, J., Pertsch, K., Zhang, J., and Lim, J. J. SPRINT: scalable policy pre-training via language instruction relabeling. *CoRR*, abs/2306.11886, 2023a.

Zhang, J., Zhang, J., Pertsch, K., Liu, Z., Ren, X., Chang, M., Sun, S., and Lim, J. J. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 302–325. PMLR, 2023b.

A. Example Input&Output of Initial Segmentation with LLM

Input:
Goal:place a microwaved apple slice on top of the black table
Action sequence: LookDown15, MoveAhead300, RotateRight90, MoveAhead100, LookDown15, PickupObject ButterKnife, SliceObject Apple, LookUp15, RotateLeft180, MoveAhead25, RotateRight90, LookDown15, PutObject DiningTable, LookUp15, RotateRight90, MoveAhead25, LookDown15, PickupObject Apple, LookUp15, RotateRight90, MoveAhead25, RotateRight90, MoveAhead150, RotateLeft90, MoveAhead250, RotateRight90, MoveAhead75, RotateLeft90, MoveAhead25, OpenObject Microwave, PutObject Microwave, CloseObject Microwave, ToggleObjectOn Microwave, ToggleObjectOff Microwave, OpenObject Microwave, PickupObject Apple, CloseObject Microwave, RotateLeft90, MoveAhead100, RotateLeft90, MoveAhead300, RotateRight90, MoveAhead75, PutObject SideTable.
Output: Action alignment:
[{ index: 0, summary action str: navigate to kitchen step 1, robot actions str: LookDown 15, MoveAhead300 },
{ index: 1, summary action str: navigate to kitchen step 2, robot actions str: RotateRight90, MoveAhead100, LookDown15 },
{ index: 2, summary action str: ”pick up butterknife and slice apple”, robot actions str: ”PickupObject ButterKnife, SliceObject Apple” },
{ index: 3, summary action str: ”return butterknife to table”, robot actions str: ”LookUp15, RotateLeft180, MoveAhead25, RotateRight90, LookDown15, PutObject DiningTable” },
{ index: 4, summary action str: ”pick up apple slice”, robot actions str: ”LookUp15, RotateRight90, MoveAhead25, LookDown15, PickupObject Apple” },
{ index: 5, summary action str: ”navigate to microwave step 1”, robot actions str: ”LookUp15, RotateRight90, MoveAhead25, RotateRight90, MoveAhead150” },
{ index: 6, summary action str: ”navigate to microwave step 2”, robot actions str: ”RotateLeft90, MoveAhead250, RotateRight90, MoveAhead75” },
{ index: 7, summary action str: ”navigate to microwave step 3”, robot actions str: ”RotateLeft90, MoveAhead25” },
{ index: 8, summary action str: ”place apple slice in microwave and turn on”, robot actions str: ”OpenObject Microwave, PutObject Microwave, CloseObject Microwave, ToggleObjectOn Microwave, ToggleObjectOff Microwave” },
{ index: 9, summary action str: ”retrieve microwaved apple slice”, robot actions str: ”OpenObject Microwave, PickupObject Apple, CloseObject Microwave” },
{ index: 10, summary action str: ”navigate to black table step 1”, robot actions str: ”RotateLeft90, MoveAhead100, RotateLeft90, MoveAhead300” },
{ index: 11, summary action str: ”navigate to black table step 2”, robot actions str: ”RotateRight90, MoveAhead75” },
{ index: 12, summary action str: ”place microwaved apple slice on black table”, robot actions str: ”PutObject SideTable” }],

B. Prompt Example

You are watching a robot do household tasks. For the following task and sequence of actions taken by the robot, segment the actions into no more than 40 skills where each skill corresponds to one part of the action sequence. The answer should a python dictionary in the form of: {(description of the first skill): (list of the actions that the robot took which correspond to the first skill), (description of the second skill): (list of the actions that the robot took which correspond to the second skill), etc.}. The number of actions assigned to each skill should not exceed 5 but should be larger than 1. The segmentation should be as reasonable and fine-grained as possible. There should not be any leftover actions and should recover the given sequence of actions in the exact same order if we concatenate these actions in the order of the skills.
Example 1:
Input Goal: turn on light on bureau top while holding clock. Input Actions: LookDown15, MoveAhead150, RotateLeft90, MoveAhead50, LookDown15, PickupObject AlarmClock, LookUp15, RotateLeft90, MoveAhead50, RotateRight90, MoveAhead75, RotateRight90, ToggleObjectOn DeskLamp
Output: {approach bureau (step 1): [LookDown15, MoveAhead150], approach bureau (step 2): RotateLeft90, MoveAhead50, LookDown15], pick up clock: [PickupObject AlarmClock], move to desk lamp (step 1): [LookUp15, RotateLeft90, MoveAhead50], move to desk lamp (step 2): [RotateRight90, MoveAhead75, RotateRight90], turn on desk lamp: [ToggleObjectOn DeskLamp]}
Goal: place a microwaved apple slice on top of the black table
Actions: LookDown15, MoveAhead300, RotateRight90, MoveAhead100, LookDown15, PickupObject ButterKnife, SliceObject Apple, LookUp15, RotateLeft180, MoveAhead25, RotateRight90, LookDown15, PutObject DiningTable, LookUp15, RotateRight90, MoveAhead25, LookDown15, PickupObject Apple, LookUp15, RotateRight90, MoveAhead25, RotateRight90, MoveAhead150, RotateLeft90, MoveAhead250, RotateRight90, MoveAhead75, RotateLeft90, MoveAhead25, OpenObject Microwave, PutObject Microwave, CloseObject Microwave, ToggleObjectOn Microwave, ToggleObjectOff Microwave, OpenObject Microwave, PickupObject Apple, CloseObject Microwave, RotateLeft90, MoveAhead100, RotateLeft90, MoveAhead300, RotateRight90, MoveAhead75, PutObject SideTable

C. Implementation and Training Details

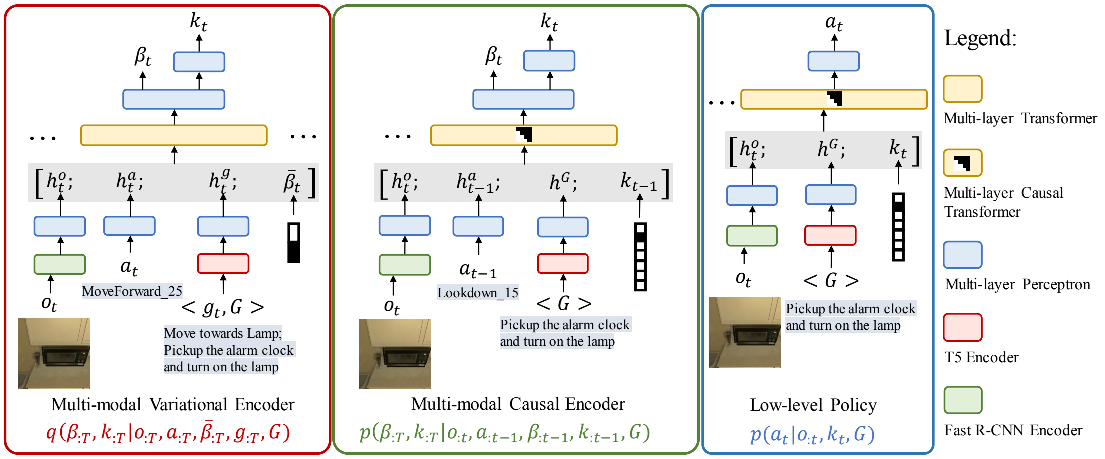


Figure 6. An overview of our Temporal Variational Inference model architecture. Each transformer corresponds to one distribution with the same color in Eqn. 3.

Note that compared with BabyAI, ALFRED is a much more complex environment and contains significantly more types of behaviors and tasks. Thus in this paper we conduct most of our downstream task evaluation and ablation studies in ALFRED.

For BabyAI, we follow the settings in the original paper. We use the partially-observable state space, with size $7 \times 7 \times 3$. We use a discrete action space with size 6: `turn left`, `turn right`, `move forward`, `pick up an object`, `drop`, `toggle`.

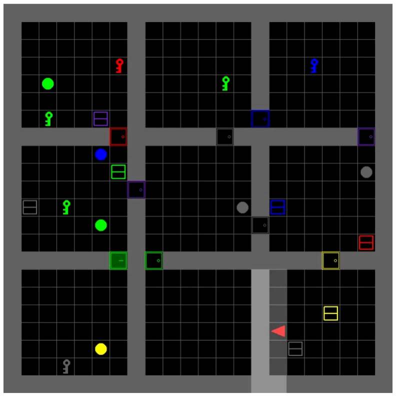


Figure 7. BabyAI

For ALFRED, we follow the settings in (Pashevich et al., 2021). We train our algorithm on the "training" dataset with cross validation and test on the "valid" dataset. There are 6 different categories of tasks in ALFRED: `pick & place`, `pick two & place`, `clean & place`, `heat & place`, `cool & place`, `examine in light`. And we randomly pick one task from each of them as the downstream tasks (we consider pick & place and stack & place as the same type of task). The action space has two components, the first component is chosen from 12 discrete action types: `Moveahead`,`Lookdown`, `Lookup`, `Rotateleft`, `Rotateright`, `pick up`, `Open`, `Close`, `Put`, `Toggle on`, `Toggle off`, `slice`. The second component is chosen from 82 object types. We use the original reward function provided by the environment for downstream task training. The observations given to agents are 300× 300 images. We use a pretrained frozen Faster R-CNN encoder (Ren et al., 2015) to preprocess the image, which gives us a $512 \times 7 \times 7$ embedding input.

Also note that many recent papers for ALFRED use a planner pretrained/programmed with heuristic methods as the navigation policy, whereas in this paper all the policies are learned only from the given data with imitation learning.

For the behavior cloning (BC) baseline in ALFRED, during online training, besides the standard RL training loss (SAC), we also augment it with a KL divergence loss the keeps **the prediction for the object type** close to the original prediction from the pretrained model. We empirically found that, **without this auxiliary loss, BC model can hardly learn anything on the downstream tasks through the RL loss.** Note that this loss is for the object prediction only, we found that adding the same loss for the action type prediction will make the performance worse.

For our approach, across the three transformers, we have an observation encoder (two convolution layers with RELU activations, followed by one linear layer with Tanh activation) that maps from the image to an embedding with dimension 256, an action encoder (one linear layer with Tanh activation) that maps from the original action (action type + object type) to the same dimension size 256, and a language encoder (one linear layer with Tanh activation) that maps from the output of T-5 encoder to the same embedding size 256. And we concatenate all the embeddings as well as the two-dimensional boundary variable $\bar{\beta}_t$ (multi-modal variational encoder), or the one-hot vector representing the skill label k (multi-modal causal encoder & low-level policy). We use the output after passing through one linear layer as the input (512) to the transformer. For multi-modal causal encoder and low-level policy, the transformers are causally-masked transformers. The transformers have 3 layers and 8 attention heads. For multi-modal variational encoder and multi-modal causal encoder, the output is processed with one linear later, followed by another linear layer with RELU activation and gumber-softmax to output the prediction for β. Similarly, two separate linear layers with RELU activation and Gumbel-Softmax will output the prediction for k. For low-level policy, the output is followed by two separate MLPs with Gumbel-Softmax that outputs the predictions for action type and object type respectively. During offline training when calculating the supervised learning loss, we calculate the cross entropy loss for the prediction of action types and object types separately, and take the weighted sum of them as the total loss (weight 1 for the action type and weight 0.1 for the object type).

We provide a list of hyperparameters and their values in Table 2:

Table 2. Hyperparameters of LAST

Hyperparameters	Value
learning rate	$3e-4$
batch size	16
Size of skill library	100
weight of KL loss	0.0001
λ	1, 0.1, 0.01
γ	0.99
temperature (SAC)	1
α_1	0.01
α_2	1
training epochs	80,140

D. Temporal Variational Inference

We first write the generative model for the joint distribution $p(\tau, \phi)$ as:

$$\begin{aligned} p(o_{:T}, a_{:T}, \beta_{:T}, g_{:T}, \bar{\beta}_{:T}, k_{:T}, G) =& p(o_1)p(\bar{\beta}_{:T})p(g_{:T})p(G) \\ & \prod_{t=1}^{T} p(o_{t+1} \mid o_t, a_t)\pi(a_t \mid o_{:t}, k_t, G)p(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G) \\ & p(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G) \end{aligned} \tag{9}$$

where the inference for both variables $p(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G)$ and $p(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)$ is only affected by history and current variables.We add such causal constraints to ensure the inference policy can be used online where only the information till the current step can be used.

According to (Shankar & Gupta, 2020), we want to calculate the variational lower bound:

$$\log p(\tau) \geq \mathbb{E}_{q(\phi \mid \tau)} \log \frac{p(\tau, \phi)}{q(\phi \mid \tau)}.$$

Recall that in § 4.2, we factor $q(\phi \mid \tau)$ as:

$$q(\beta_{:T}, k_{:T} \mid o_{:T}, a_{:T}, \bar{\beta}_{:T}, g_{:T}, G) = \prod_{t=1}^{T} q(\beta_t \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G)q(k_t \mid o_{:T}, a_{:T}, \beta_t, k_{t-1}, g_{:T}, G),$$

Then:

$$\begin{aligned} \log p(o_{:T}, a_{:T}, G) \geq& \mathbb{E}_{q(\beta_{:T}, k_{:T} \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G)} \log \frac{p(o_{:T}, a_{:T}, \bar{\beta}_{:T}, g_{:T}, \beta_{:T}, k_{:T}, G)}{q(\beta_{:T}, k_{:T} \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G)} \\ &= \mathbb{E}_{q(\beta_{:T}, k_{:T} \mid \cdot)}[\log p(G) + \log p(\bar{\beta}_{:T}) + \log p(g_{:T}) + \sum_{t}^{T}\{\log p(o_{t+1} \mid o_t, a_t) \\ &+ \log \pi(a_t \mid o_{:t}, k_t, G) + \log p(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G) + \log p(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)\} \\ &- \log q(\beta_{:T}, k_{:T} \mid \cdot)] \end{aligned}$$

Remove constant terms:

$$\begin{aligned} J(\theta) =& \mathbb{E}_{q(\beta_{:T}, k_{:T} \mid \cdot)}[\sum_{t}^{T}\{\log \pi(a_t \mid o_{:t}, k_t, G) + \log p(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G) \\ &+ \log p(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)\} - \log q(\beta_{:T}, k_{:T} \mid \cdot)] \\ &= \mathbb{E}_{q_\theta(\phi \mid \tau)} \sum_{t=1}^{T} \log \pi_\theta(a_t \mid o_{:t}, k_t, G) - \sum_{t=1}^{T} \Big\{ \mathrm{KL}[q_\theta(\beta_t \mid o_{:T}, a_{:T}, \bar{\beta}_t, g_{:T}, G) \| p_\theta(\beta_t \mid o_{:t}, a_{:t-1}, \beta_{:t-1}, k_{:t-1}, G)] \\ &+ \mathrm{KL}[q_\theta(k_t \mid o_{:T}, a_{:T}, \beta_t, k_{t-1}, g_{:T}, G) \| p_\theta(k_t \mid o_{:t}, a_{:t-1}, k_{:t-1}, \beta_{:t}, G)] \Big\} \end{aligned}$$

E. Minimum Description Length training objective

We focus on minimizing the first term $L(\mathcal{D} \mid \phi)$. For the original trajectory:

$$\tau = \{G, o_1, a_1, \bar{\beta}_1, g_1, o_2, a_2, \bar{\beta}_2, g_2, \cdots, o_H, a_H, \beta_H, g_H\},$$

we encode it as

$$\tau_{\beta,k} = \{G, o_1, k_1, \beta_1, k_2, \beta_2, \cdots, k_H, \beta_H\}.$$

As β_t is a binary variable denoting whether to switch the skill at timestep t, we can further write the code as:

$$\tau_{\beta,k} = \{G, o_1, k_1^{\text{new}}, k_2^{\text{new}}, \cdots, k_H^{\text{new}}\},$$

where:

$$k_t^{\text{new}} \sim \beta_t q_\theta(k_t \mid o_{:T}, a_{:T}, \beta_t, g_{:T}, G) + (1 - \beta_t) k_{t-1},$$

which we already mentioned in Eqn. 4. According to the optimal code length theory (Cover & Thomas, 2001), the expected number of bits of code generated by $p(k)$ is:

$$L_k = \mathbb{E}_k[-\log p(k)].$$

Then for each k_t^{new}, the expected length is:

$$L_{k_t^{new}} = -\mathbb{E}_k \log[q_\theta(k_t \mid o_{:T}, a_{:T}, \beta_t, g_{:T}, G) q_t(\beta_t = 1 \mid \cdot) + q_t(\beta_t = 0 \mid \cdot) \mathbb{1}(k == k_{t-1})], \tag{10}$$

Then for the expected number of bits of code for the whole trajectory, as o_1 and G are constant terms not affected by θ:

$$\mathcal{L}_{\text{MDL}}(\theta) = -\sum_t \mathbb{E}_k \log[q_t(k \mid \cdot) q_t(\beta_t = 1 \mid \cdot) + q_t(\beta_t = 0 \mid \cdot) \mathbb{1}(k == k_{t-1})], \tag{11}$$

In LOVE (Jiang et al., 2022), the compression objective can be described as (using our denotations):

$$L_{CL}(\theta) = n_s \mathcal{H}[k] = \mathbb{E}\Big[\sum_t \beta_t \log q_t(k \mid \cdot)\Big] \tag{12}$$

We argue that our objective function is Eqn. 11 can more accurately reflect the skill switches's influence on the code length and also increase trainability. The biggest differences here are that 1. The objective in Eqn. 12 ignores the effect of the skill k_{t-1} chosen at last timestep to the length of the code for current timestep t, i.e., Eqn. 12 does not have the second term in Eqn. 11($q_t(\beta_t = 0 \mid \cdot) \mathbb{1}(k == k_{t-1})$). The approximation may make huge differences in practice as shown in the following example (next paragraph). 2. In Eqn. 12 the sampled β (binary values) are used to calculate the overall objective and then use straight-through estimator to approximate the gradients for $q_\theta(\beta)$, while our formulation allows us to directly calculate the gradients through $q_\theta(\beta)$, giving us less noisy training signals, and thus increase trainability.

Specifically, consider an example where the skill label k has three possible values (1, 2, 3) at timestep t, $q_t(1 \mid \cdot) = 0.1$, $q_t(2 \mid \cdot) = 0.2$, $q_t(3 \mid \cdot) = 0.7$, where choosing the third skill has the largest probability, and $q_t(\beta = 1 \mid \cdot) = 0.1$, $q_t(\beta = 0 \mid \cdot) = 0.9$. Using Eqn. 12 in LOVE, **no matter what the last skill is**, the number of bits of the code at timestep t can be calculated as: $l = \mathbb{E}\Big[\beta_t \log q_t(k \mid \cdot)\Big] = -0.1 * (0.1 \log 0.1 + 0.2 \log 0.2 + 0.7 \log 0.7) - 0.9 * 1 \log 1 = \mathbf{0.08}$. If using our objective, when the last skill $\mathbf{k_{t-1} = 1}$, the results would be: $l = -\mathbb{E}_k \log[q_t(k \mid \cdot) q_t(\beta_t = 1 \mid \cdot) + q_t(\beta_t = 0 \mid \cdot) \mathbb{1}(k == k_{t-1})] = -((0.01 + 0.9) \log(0.01 + 0.9) + 0.02 \log 0.02 + 0.07 \log 0.07) = \mathbf{0.35}$. When the last skill $\mathbf{k_{t-1} = 3}$, the results would be: $l = -(0.9 \log 0.9 + 0.02 \log 0.02 + (0.07 + 0.9) \log(0.07 + 0.9)) = \mathbf{0.20}$, which is smaller than the case (0.2) when the last skill $k_{t-1} = 1$. This is what we want the code length to express: **If the current inference policy gives a high probability for the skill that is the same as the last skill (3), we should need fewer bits to encode current timestep and the code length should be shorter, because if it's the same skill then it is just one intermediate step of the skill we chose before and we do not need to communicate it.** And the agent will be encouraged to infer fewer boundaries (switching skills) in each trajectory. In contrast, the objective in LOVE does not reflect this property and the code length remains the same for different last skills and thus may increase the difficulty of optimization in practice.

Note that Zhang et al. (2021) also leverages MDL to learn skills but their proposed objective is actually equivalent to variational inference with a different graphical model.

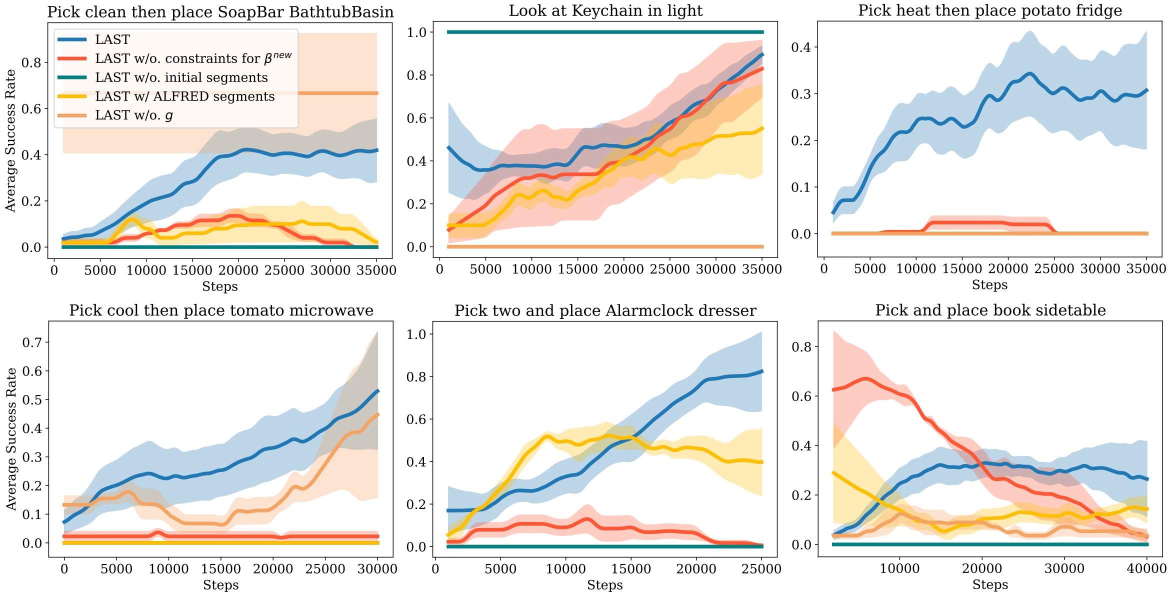


Figure 8. Ablation studies of LAST against different variants in six downstream tasks of ALFRED. We plot success rate against timesteps.

F. Ablation study and more experimental results

We compare LAST with several variants: 1. Using the segments and language instructions given by ALFRED dataset in place of LLM generated initial segmentation. 2. Removing the constraints on inferring β_t^{new} (Eqn. 4) during temporal variational inference. 3. Removing the first step (initial segmentation) and directly do temporal variational inference training on the original data. 4. Removing the generated language annotation g from the input to the variational inference model (only giving it the segmentation results $\bar{\beta}_{:T}$).

Table 3. Online (zero-shot) task success rate comparison on BabyAI.

Model	BossLevel	MiniBoss	Synth	SynthLoc	SynthSeq	Average
LOVE	27	31	40	63	25	37
BC	25	23	**53**	48	20	34
LISA	27	26	49	49	27	36
LAST w/o. MDL	**38**	**38**	42	54	28	40
LAST	36	34	48	**66**	**32**	**43**

We evaluate in the "learning on Downstream Tasks" setting. As shown in Fig. 8, LAST achieves the best overall performance. LAST without the initial segments achieves 100 percent success rate in the `Look at Keychain in light` task but completely fails in all the other tasks, indicating that it overfits to the short horizon tasks. Without the constraints in Eqn. 4, the search space becomes much larger and LAST struggles to find the solution that leads to reusable skills. The segmentation of ALFRED dataset is labeled by human and we found that usually a trajectory is segmented into 4/5 pieces, which makes sense for decomposing each individual task, but it is hard to merge these relatively long subsequences into reusable skills as shown in the results. The initial segmentation made by LLMs gives a more detailed decomposition, as a result, we can more easily find reusable skills by merging these segments. LAST without the generated language annotations g generally performs not as good LAST, it reaches the highest performance in the pick clean task, but completely fails in the look at Keychain task where all other methods perform relatively well.

In Table 3, we compare our algorithm with a variant that does not use MDL as the auxiliary training objective in BabyAI zero-shot scenario. While LAST still achieves higher average success rate, the gap between using & not using MDL is smaller than we saw in the online adaptation scenario.

Table 4. Online (zero-shot) subtask success rate comparison for different number of skills discovered on ALFRED. The 4 & 5 skills results are generated by setting the weight of MDL loss term to 1. The 9 & 11 skills results are from weight 0.1, and the 15 skills & 20 skills are from weight 0.01.

Model	Avg	Clean	Cool	Heat	Pick	Put	Slice	Toggle	GoTO
4 skills	47.0	82.3	22.0	9.6	63.8	48.9	24.3	46.8	41.8
5 skills	48.4	70.8	34.9	63.2	62.0	51.0	40.5	42.2	40.7
9 skills	49.6	83.2	32.1	40.4	64.5	56.4	37.8	37.6	41.3
11 skills	52.1	81.9	65.0	82.0	63.4	52.7	48.2	36.5	43.9
15 skills	51.7	77.9	75.2	95.6	63.0	57.4	34.2	69.4	39.1
20 skills	50.0	82.3	45.0	96.3	60.1	54.1	45.0	56.6	39.7

We also empirically investigate the influence of the weight for the MDL auxiliary objective. Interestingly, we find that changing the weight of the MDL loss can change the number of discovered skills in the same order. As shown in Table 4, changing the weight from 0.01 to 1 results in an increase in the number of skills from 4 to 20. The same table suggests that the zero-shot transfer performance using the discovered skills does not monotonically increase/decrease with respect to the number of the skills. A number in the middle (11 skills) gives the best performance.

F.1. Directly using LLM to plan in ALFRED

As we assume language-annotated low-level action space is available, it is reasonable to compare to a baseline where we use an LLM to directly output action probabilities and interact with the environment. We run this evaluation on ALFRED. We directly use GPT-4V(ision) as the LLM-based policy and run SayCan (Ichter et al., 2022)-style zero shot evaluation on ALFRED. Specifically, at each environment step, we provide GPT-4V with the current observation from the environment (image), the previously executed action sequence, the task goal, the language-annotated action space, example trajectories and ask it to predict the action. The action is further examined by a predefined function to make sure it is among the available ones at the current time step. We find that **GPT-4V is not able to solve any one of the six downstream tasks we tested.**

G. More Qualitative Results

We show some examples of LAST's skill segmentation in Fig. 9, and the five frequently discovered skills in Fig. 10.

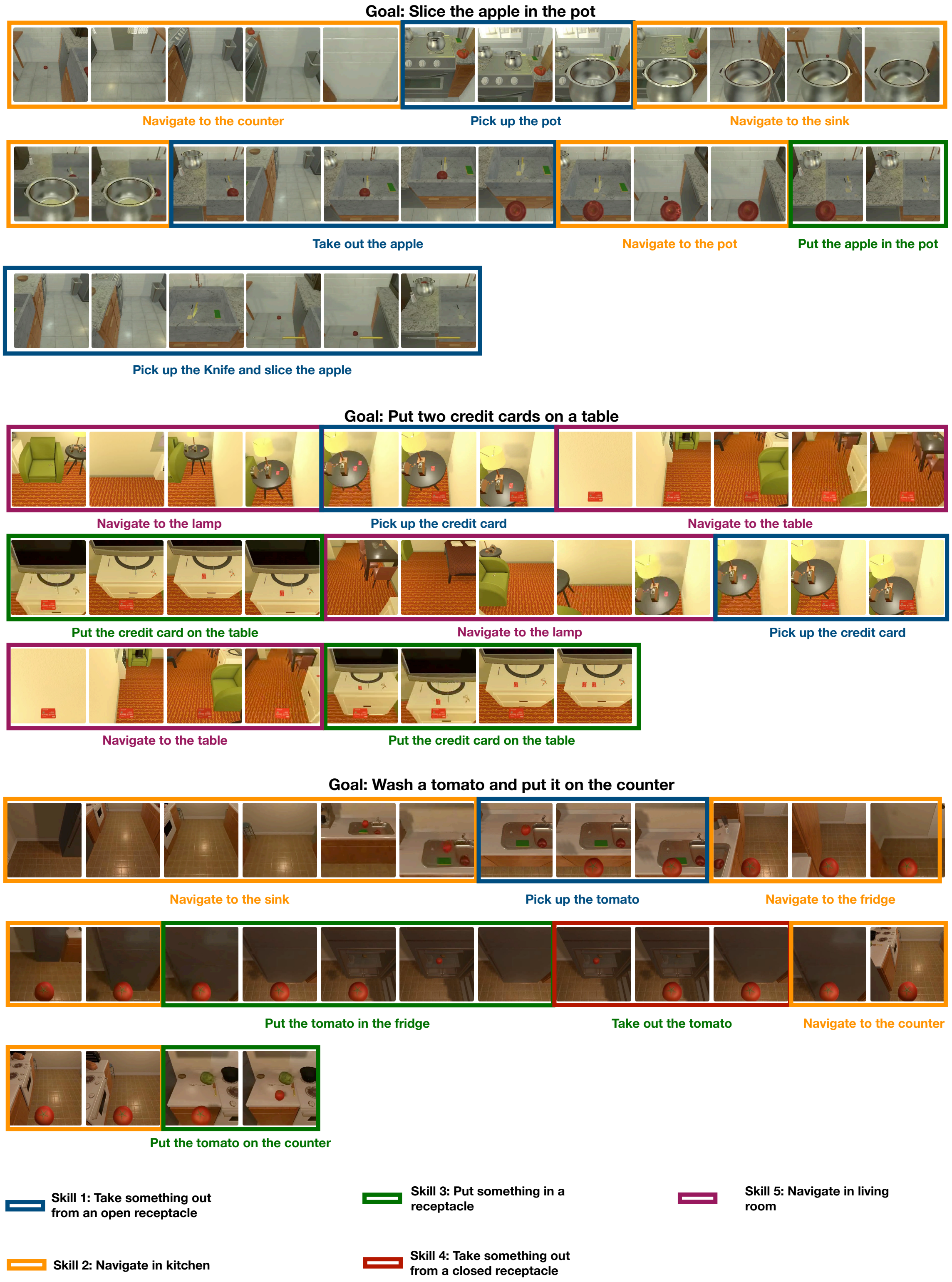

Goal: Slice the apple in the pot
Navigate to the counter
Pick up the pot
Navigate to the sink
Take out the apple
Navigate to the pot
Put the apple in the pot
Pick up the Knife and slice the apple
Goal: Put two credit cards on a table
Navigate to the lamp
Pick up the credit card
Navigate to the table
Put the credit card on the table
Navigate to the lamp
Pick up the credit card
Navigate to the table
Put the credit card on the table
Goal: Wash a tomato and put it on the counter
Navigate to the sink
Pick up the tomato
Navigate to the fridge
Put the tomato in the fridge
Take out the tomato
Navigate to the counter
Put the tomato on the counter
Skill 1: Take something out from an open receptacle
Skill 2: Navigate in kitchen
Skill 3: Put something in a receptacle
Skill 4: Take something out from a closed receptacle
Skill 5: Navigate in living room

Figure 9. LAST's skill segmentation for different trajectories.

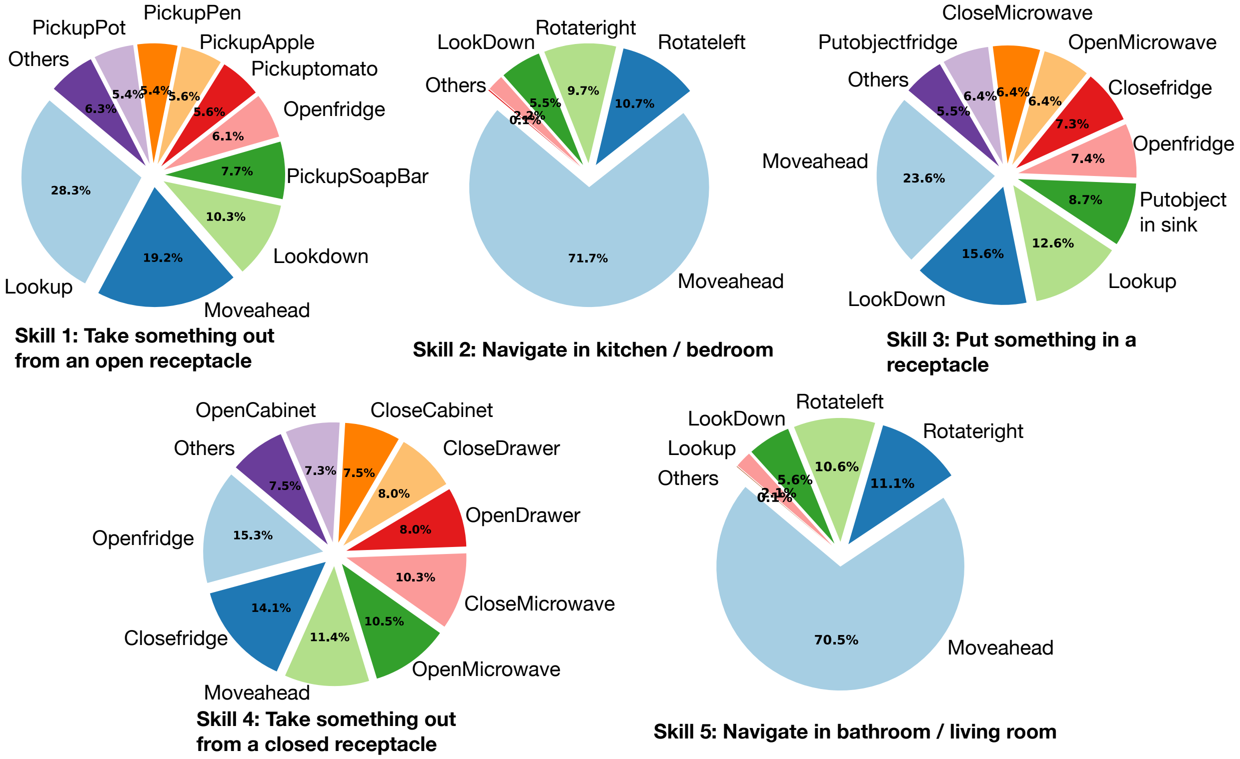


Figure 10. Five frequently discovered skills of LAST and their most-commonly used actions.

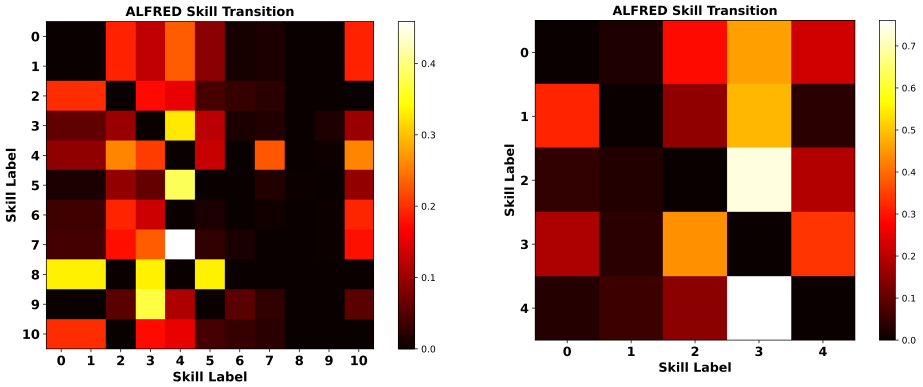


Figure 11. Transition probability matrix for two sets of learned skills from LAST. Each row shows the probabilities of changing from one skill to all the other skills. The first five skills ($0 \sim 4$) correspond to the five commonly-discovered skills in Fig. 5(the order may be different). We see that in the first skill set, there are another six skills discovered besides the five most common skills but the transition probabilties from the other skills to them in general are much lower than the first five skills.