electronics

MDPI

Article

Self-Supervised Skill Learning for Semi-Supervised Long-Horizon Instruction Following

Benhui Zhuang [1], Chunhong Zhang [2] and Zheng Hu [1,*]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; zhuangbenhui@bupt.edu.cn
[2] Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China; zhangch@bupt.edu.cn
* Correspondence: huzheng@bupt.edu.cn

Abstract: Language as an abstraction for hierarchical agents is promising to solve compositional long-time horizon decision-making tasks. The learning of the agent poses significant challenges, as it typically requires plenty of trajectories annotated with languages. This paper addresses the challenge of learning such an agent under the scarcity of language annotations. One approach for leveraging unannotated data is to generate pseudo-labels for unannotated trajectories using sparse seed annotations. However, as the scenes of the environment and tasks assigned to the agent are diverse, the inference of language instructions is sometimes incorrect, causing the policy to learn to ground incorrect instructions to actions. In this work, we propose a self-supervised language-conditioned hierarchical skill policy (SLHSP) which utilizes unannotated data to learn reusable and general task-related skills to facilitate learning from sparse annotations. We demonstrate that the SLHSP that learned with less than 10% of annotated trajectories has a comparable performance to one that learned with 100% of annotated data. Our approach to the challenging ALFRED benchmark leads to a notable improvement in the success rate over a strong baseline also optimized for sparsely annotated data.

Keywords: semi-supervised learning; language grounding; skill learning

Citation: Zhuang, B.; Zhang, C.; Hu, Z. Self-Supervised Skill Learning for Semi-Supervised Long-Horizon Instruction Following. *Electronics* **2023**, *12*, 1587. https://doi.org/10.3390/electronics12071587

Academic Editor: Fernando De la Prieta Pintado

Received: 26 January 2023
Revised: 22 March 2023
Accepted: 22 March 2023
Published: 28 March 2023

1. Introduction

Application areas such as robotics and finance typically involve long-horizon compositional decision-making tasks, and natural language as an abstraction for hierarchical policies is a promising paradigm for solving these tasks. As a type of hierarchical policy that applies language representations, latent language policy (LLP) maps from a *high-level goal* to a sequence of *natural language instructions*, and then the policy grounds these instructions into actions. Here, the grounding of a language instruction into actions involves associating the language instruction with a sequence of actions that reflect its meaning and meet its requirements. However, existing LLP methods assume either short-time horizon tasks [1,2] or *fully-supervised datasets* that include both the mapping from high-level goals to instruction sequences and the mapping from instructions to actions [3–5]. An annotated trajectory, as shown in Figure 1, contains a high-level task goal, a set of annotated language instructions (in red), and alignments between instructions and state–action pairs. As manually annotating such a dataset is time-consuming and expensive, it is extremely challenging to scale up such fully-supervised methods in complex environments and long-horizon tasks.

To obviate the need for a large number of language annotations, existing methods either resort to exploiting supervision with information in various modalities (e.g., a voxel map) [3,6], pre-trained models [7–10], instructions generated using rule-based templates [11], or unannotated data [12,13]. The focus of our work is to leverage unannotated trajectories in a *sparsely annotated dataset*, in which less than 10% of trajectories are annotated with natural language instructions to learn an LLP. To achieve this, previous studies [12–15]

propose utilizing unannotated trajectories through *pseudo-label generation* [16], which augments unannotated trajectories with pseudo-instruction labels. Specifically, it labels unannotated trajectories with language instructions predicted by a model trained for labeling. However, as the scenes of the environment and tasks assigned to the agent are diverse, the inference of language instructions is sometimes incorrect. In this case, the policy learns to ground incorrect instructions to actions, which deteriorates the performance of the LLP; this problem is exacerbated as the amount of available annotation decreases. This problem of overfitting to incorrect pseudo-labels is called *confirmation bias* [17].

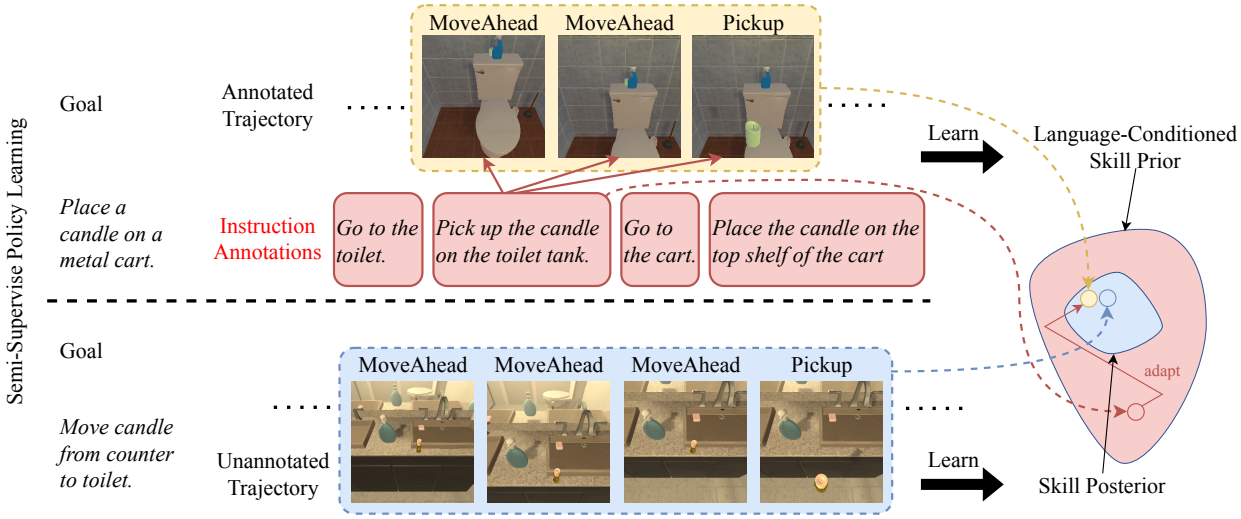


Figure 1. Semi-supervised LLP learning with sparse language annotations.

To alleviate the problem of confirmation bias, we propose a **s**elf-supervised **l**anguage-conditioned **h**ierarchical **s**kill **p**olicy (SLHSP). The SLHSP exploits unannotated trajectories by learning a generative model for reusable and task-general action segments called "skills" through self-supervised learning. The learned skill representation facilitates the grounding of instructions into action sequences. Instead of labeling skill trajectories (action segments) with pseudo-labels (instructions) and learning a mapping between the two symbolic spaces (the space of word sequence and the space of action sequence), the SLHSP grounds language instructions directly to the internal skill space of the generative model. This internal skill space is structurally organized and can be seen as a *continuous* and *regularized* version of the symbolic action sequence space. These two properties make the grounding of instruction efficient, bypassing the need for augmentation to the annotated trajectories and avoiding the error introduced by incorrect labeling.

Specifically, Figure 1 illustrates the use of unannotated data to learn a latent skill space for effective grounding of language instruction. As mentioned previously, a sampled skill in the latent skill space corresponds to an action sequence. Grounding language instructions into symbolic action sequences is equivalent to first aligning language instructions with latent skill variables and then decoding the latent skill variables into action sequences by reusing the learned generative model. The continuous and regularized nature of the latent skill space makes it data-efficient in aligning instruction features with latent skill variables. More concretely, this aligning process encodes natural language instruction features into the same vector space as latent skills. The target feature vector for a given language instruction is obtained by encoding the corresponding annotated trajectory using the learned generative model, and this process can be seen as "pseudo-latent label generation". The major challenge in learning the aforementioned latent skill space is determining reusable and general task-related actions' segments as skills to handle the combinatorial complexity of the task. To address this challenge, the SLHSP learns a novel high-level separator policy, supervised by annotated data, to preprocess the unannotated trajectories, and extracts action sequences that meet the above requirements.

In the experiments, we show that grounding language to the learned skill space through the aforementioned skill requires less than 10% of language annotations. The SLHSP outperforms semi-supervised baselines and shows a performance comparable to that of existing state-of-the-art methods that utilize additional inputs or prior knowledge in both training and evaluation. Experiments visualizing the latent skill variable demonstrate that the learned skills are structurally organized.

The main contributions of this work are summarized as follows.

- We propose a self-supervised latent skill space with reusable and general task-related skills that can be effectively applied to a wide range of instruction following tasks.
- We propose an annotation-efficient method for the learning of the LLP.
- We propose an efficient grounding method that aligns instructions and learned skills in a latent space.

2. Related Work

The scarcity of annotated data presents a challenge in handling long-horizon instruction following tasks. Existing works have extensively studied this issue, either by exploiting external knowledge (e.g., large language models, human feedback, or pre-defined templates) or by designing efficient policies to utilize unannotated data. In this section, we first review previous approaches that solve the scarcity problem by utilizing external knowledge, and then compare the SLHSP with existing approaches that utilize unannotated data.

2.1. Utilizing External Knowledge

Additional supervisions for training and *additional agent observations* are two types of external knowledge. For example, several studies have applied rule-based symbolic plans as an additional supervision for training [3,4,6]. Existing LLP methods typically provide step-by-step natural language instructions to actions as additional language observations to the agent [18–20]. In addition to language observations, [21] uses multiple visual observations around the agent for better sensing of the environmental state. Depth and semantic visual information is also utilized to learn a persistent spatial representation (e.g., a voxel map) [3,6], which provides the agent with a holistic view of the environmental state. Contrary to these works, the SLHSP assumes no additional supervisions for training and additional observations. The SLHSP utilizes unannotated data to reduce the cost of data annotation.

Large language models (LLMs) have demonstrated success in few-shot learning on vision-language-related tasks. In vision-language navigation and manipulation tasks, LLMs enable the agent to reason in novel scenarios using language instructions [10,22]. The inferred instruction plan can be semantically translated to admissible actions conditioned on existing demonstrations [7]. Li et al. [8] investigated how LLMs adapt to sequential decision-making problems, while Huang et al. [9] avoid relying on expert demonstrations by leveraging language feedbacks. Unlike these works, which ground language instruction into actions relying on LLMs, the SLHSP utilizes knowledge extracted from unannotated data.

Rule-based semantic parsing converts task goals into symbolic logic expressions which incorporate prior knowledge from designed rules. For example, the task goal is mapped to lambda calculus expressions [23] or linear temporal logic [24], and then these template expressions are interpreted into agent actions through fixed rules. However, instead of parsing task goals through predefined rules, the SLHSP reasons through natural language instructions, making it more feasible in environments with complex state and action spaces.

2.2. Utilizing Unannotated Data

Generation *pseudo-labels* for unannotated trajectories is commonly used to augment them so that the labeled data can be utilized for supervised learning of the policy. Sharma et al. [12] proposed learning a model from sparse annotation in order to segment unannotated trajectories into semantically meaningful segments and label the segments with language instructions. Combined with the annotated data, the labeled data are leveraged in the

imitation learning of the language-conditioned policy. Similarly, Xiao et al. [13] analyzed the generation of pseudo-labels through a pre-trained vision-language model. In the context of reinforcement learning, existing works relabel collected experiences with either a hard-coded [15] or a learned label generator [14]. Instead of labeling unannotated trajectories with pseudo-language instruction labels, the SLHSP utilizes unannotated trajectories through self-supervised learning to avoid the introduction of errors caused by incorrect labeling.

Learning modules that generate reusable actions segments are also a promising approach to utilizing unannotated trajectories, and these modules are typically called *skills*. These skills can be learned from unannotated trajectories collected from an expert policy [20], a suboptimal behavior policy [25,26], or even a random policy [25]. Thus, the rich source of trajectory data makes this type of method widely applicable. As skills in LLPs are required to execute language instructions, we categorize these methods according to how the language instructions are associated with the learned skills. In the context of *reinforcement learning*, associating instructions with skills has been studied with either a teacher policy (e.g., a human teacher) or a rule-based policy that provides instruction labels as supervision [2,14,27,28], short-horizon tasks [25] or synthetic languages [29]. Contrary to these works, the SLHSP assumes long-horizon tasks and natural language instructions. In the context of *imitation learning*, Lynch et al. [30] associate learned skills with language instructions through a goal space defined by goal images, and only the image of the final goal state is used to represent each sampled skill. In contrast, the SLHSP defines skills in a space wherein each sample is conditioned on the entire trajectory sequence so that it is applicable for complex tasks whose goal state cannot be fully expressed by a single image.

3. Method

The aim of our work is to learn a latent language policy, which is a hierarchical policy that applies languages as the intermediate representation, for long-horizon compositional tasks using only sparsely annotated trajectories. The essential idea through which the SLHSP will reach this target is the utilization of unannotated trajectories to facilitate the grounding of language instructions into actions and reduce the number of required language annotations. To alleviate the problem of confirmation bias [17], the SLHSP avoids augmenting unannotated trajectories through pseudo-labels, and utilizes these trajectories by learning reusable skills through self-supervised learning.

Learning reusable skills and *grounding language instructions to learned skills* are the two essential processes of the SLHSP. Using only unannotated dataset $\mathcal{D}$, the learning of reusable skills is formulated by learning general and reusable action segments that can be temporally composed to solve tasks. The extracted skills are represented by a continuous latent variable $z \in \mathcal{Z}$, where $\mathcal{Z}$ is the latent skill space. Given a small annotated dataset $\mathcal{D}^{\text{ann}}$, the grounding of language instructions to learned skills is defined as mapping a language instruction to a skill distribution over the latent skill space $\mathcal{Z}$ so that the sampled latent skill is decoded into an action sequence that follows the instruction.

The SLHSP has three components, which we will describe in this section: (1) reusable skill extraction from unannotated data through self-supervised learning, (2) prior skill adaptation from sparsely annotated data for grounding natural language instructions to the learned skills, and (3) language plan generation for composing repeated subtasks into a long-horizon task through natural language instructions. Figure 2 provides a graphical overview of the SLHSP.

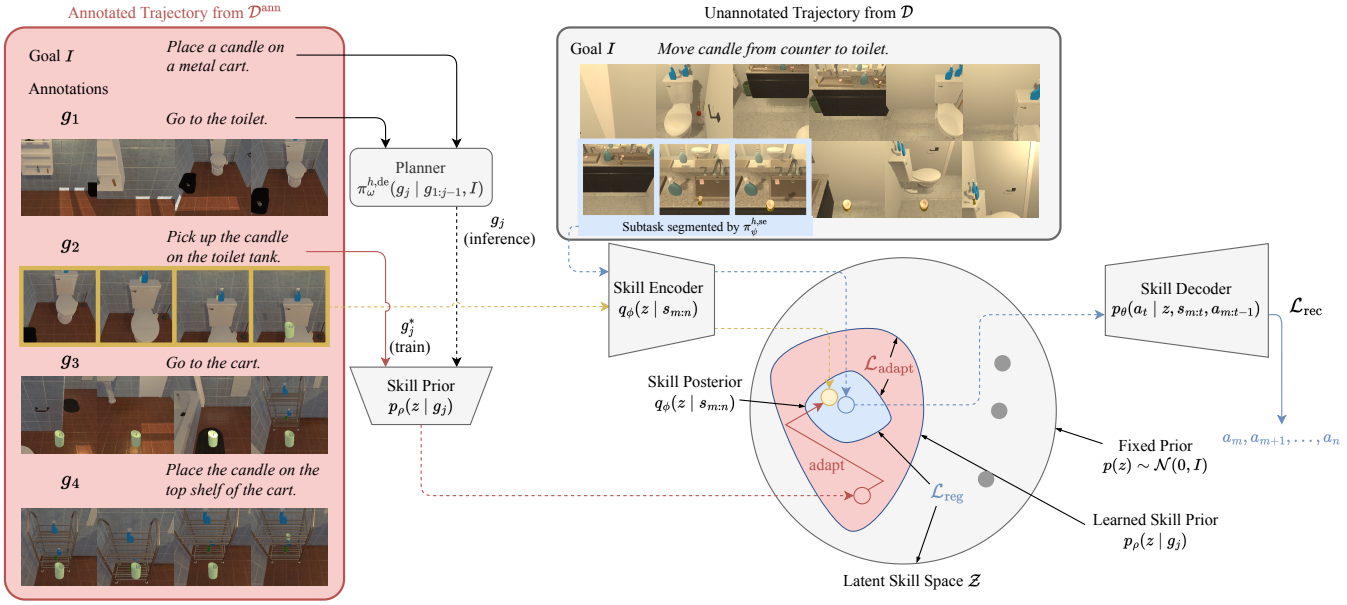


Figure 2. Architecture of the SLHSP. The skill encoder and decoder are self-supervised on $\mathcal{D}$, while the skill prior, planner and separator are supervised using annotated trajectories $\mathcal{D}^{\text{ann}}$.

3.1. Problem Formulation

The SLHSP aims to solve long-horizon instruction following tasks which are formulated as a Markov decision process (MDP) defined by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma\}$ of states, actions, transition probabilities, reward functions, and discount factors. At each time step, the agent obtains a state s_t and outputs an action a_t, which is executed in the environment, resulting in the transition to the next state s_{t+1} according to $\mathcal{T}$. The task goal for the agent is defined by a high-level description written in natural language.

We assume access to two sets of expert trajectories: a large unannotated dataset $\mathcal{D}$ and a small annotated dataset $\mathcal{D}^{\text{ann}}$. In the annotated dataset $\mathcal{D}^{\text{ann}}$, a trajectory τ_i^{ann} is annotated with multiple step-level language instructions $\{g_1, \ldots, g_j, \ldots\}$, and each state–action pair in τ_i^{ann} is aligned with an annotated instruction to form a state–action–annotation pair $\{(s_t, a_t, g_j)\}_{t=1}^{t=T_i}$, where T_i is the length of τ_i^{ann}. Here, we define the state–action sequence aligned with a language instruction as a *subtask*. Thus, the trajectory τ_i^{ann} is segmented into several subtasks, and each subtask corresponds to an annotated language instruction g_j. We use $\tau^{\text{ann},j}$ to denote the jth subtask trajectory in the annotated trajectory τ_i^{ann}. For example, a task j with goal I_j = *Examine a clock with a lamp* in $\mathcal{D}^{\text{ann}}$ has a set of language annotations {g_1 = *Go to the dresser*, g_2 = *Pick up the clock that is on the dresser*, g_3 = *Hold it up to the lamp*, g_4 = *Turn the lamp on*}, and the trajectory τ_j^{ann} is segmented into subtasks by sequentially aligning each step with a language annotation $\{(s_t, a_t, g_j)\}_{t=1}^{t=T_j}$. However, in the unannotated dataset $\mathcal{D}$, both the language annotations and segmentations are unavailable. As an example, a task i with goal I_i = *Turn lamp on to look at clock* in $\mathcal{D}$ has only a state–action trajectory $\tau_i = \{(s_t, a_t)\}_{t=1}^{t=T_i}$.

3.2. Reusable Skill Extraction

The basic idea of reusable skill extraction from unannotated trajectories is to learn a generative model of action sequences (skills) that can be applied across tasks. A skill $a_{m:n}$ is defined as a sequence of actions $\{a_m, \ldots, a_n\}$ of variable length. As the learned skills should be reusable and general, the boundary m and n of a skill in an unannotated trajectory determines the quality of the extracted skill. As there is no information in $\mathcal{D}$ to indicate the appropriate boundary for skills, the SLHSP resorts to the information in $\mathcal{D}^{\text{ann}}$. Supervised by the annotated subtasks, the SLHSP learns a subtask *separator policy* $\pi_\psi^{h,\text{se}}$ (discussed in Section 3.4). The learned separator policy is then applied to preprocess

the unannotated trajectories, and each task trajectory τ is segmented into multiple subtask trajectories $\tau = \{\tau^1, \ldots, \tau^j, \ldots\}$ to help extract reusable and task-general skills.

Inspired by [31], the SLHSP learns a low-dimensional skill space $\mathcal{Z}$ through a stochastic latent variable model (see Figure 2) from the segmented trajectories. Formally, assuming that m and n are starting and ending points of a subtask trajectory τ^j determined by the separator policy $\pi_\psi^{h,\text{se}}$, encoding the subtask information requires learning an encoder $q(z \mid s_{m:n})$ and a decoder $p(s_{m:n} \mid z)$, where $s_{m:n}$ is the state sequence from step m to step n. These two models are learned by using the amortized variational inference [32], and the objective is to maximize the evidence lower bound (ELBO):

$$\log p(s_{m:n}) \geq \underbrace{\log p(s_{m:n} \mid z)}_{\mathcal{L}_{\text{rec}}} + \beta \underbrace{(\log p(z) - \log q(z \mid s_{m:n}))}_{\mathcal{L}_{\text{reg}}}. \tag{1}$$

Here, $\mathcal{L}_{\text{rec}}$ is the reconstruction term and β is a hyperparameter used to weight the regularization term [33], which is denoted as $\mathcal{L}_{\text{reg}}$. Note that the reconstruction of the state's sequence $s_{m:n}$ depends on the transition probability $\mathcal{T}$ of the environment in which the agent acts. This is because the agent cannot directly determine what the upcoming states in the future are, but instead affects future states through predicted actions. Considering the transition probability $\mathcal{T}(s_{i+1} \mid s_i, a_i)$, the reconstruction term $\mathcal{L}_{\text{rec}}$ in Equation (1) can be rewritten as follows:

$$\log p(s_{m:n} \mid z) = \log p(s_m) \prod_{i=m}^{n} \underbrace{\mathcal{T}(s_{i+1} \mid s_i, a_i)}_{\text{transition probability}} \; p(a_i \mid s_i, z). \tag{2}$$

On the right-hand side of Equation (2), the transition probabilities are determined by the environment. This indicates that the skill decoder should reconstruct the subtask trajectory by predicting actions that result in $s_{m:n}$. Therefore, optimizing the agent's parameters according to the reconstruction loss in Equation (1) is equivalent to optimizing the following term:

$$\mathcal{L}_{\text{rec}} = \log p(s_m) \prod_{i=m}^{n} p(a_i \mid s_i, z). \tag{3}$$

Replacing the reconstruction loss in Equation (1) with Equation (3), the evidence lower bound is formulated:

$$\log p(s_{m:n}) \geq \log p(s_m) \prod_{i=m}^{n} p(a_i \mid s_i, z) + \beta \log p(z) - \log q(z \mid s_{m:n}). \tag{4}$$

With subtask trajectories from the preprocessed unannotated dataset $\mathcal{D}$, the learning of the skill encoder q_ϕ, parameterized by ϕ, and the skill decoder p_θ, parameterized by θ, is carried out to maximize the following lower bound:

$$\log p(s_{m:n}) \geq \mathbb{E}_{\tau_j \sim \mathcal{D}, z \sim q_\phi(z \mid s_{m:n})} \left[\log \prod_{i=m}^{n} p_\theta(a_i \mid s_i, z) + \beta \big(\log p(z) - \log q_\phi(z \mid s_{m:n}) \big) \right], \tag{5}$$

where p_θ and q_ϕ are modeled as deep neural networks. The prior $p(z)$ is set to be a unit Gaussian $\mathcal{N}(0, I)$. To support the prediction of actions with historical decision information, the skill decoder $p_\theta(a_t \mid s_t, a_{t-1}, z)$ is augmented with previous action a_{t-1} as an input when decoding each action. More details about the implementation of the skill encoder and decoder are discussed in Section 3.5.

As this learning process requires no language annotations, the dataset $\mathcal{D}$ can be collected via automatically generated trajectories through other learned or rule-based agents. The overall procedure of latent skill space learning is presented in Algorithm 1.

Algorithm 1: Latent Skill Space Learning

Input: Parameters θ and ϕ, learning rates α_θ and α_ϕ, number of epochs N, pre-trained $\pi_\psi^{h,\text{se}}$, and unannotated dataset $\mathcal{D}$
Output: Learned parameters θ and ϕ
Initialize the skill encoder ϕ and the skill decoder θ.
for $n \leftarrow 0$ **to** N **do**
  repeat
    $\tau \sim \mathcal{D}$
    $(\tau^1, \tau^2, \dots) \leftarrow \pi_\psi^{h,\text{se}}(\tau)$ # *segment trajectory into subtasks trajectories*
    for τ^j *in* $(\tau^1, \tau^2, \dots)$ **do**
      $(s_{m:n}, a_{m:n}) \leftarrow \tau^j$
      $z \sim q_\phi(z \mid s_{m:n})$
      $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \beta\big(\log p(z) - \log q_\phi(z \mid s_{m:n})\big)$
      $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \log \prod_{i=m}^{n} p_\theta(a_i \mid s_i, a_{i-1}, z)$
    end
  until *the last episode* τ *in* $\mathcal{D}$;
end

3.3. Skill Prior Adaptation

Mapping language instructions into action sequences is extremely challenging due to the complex dynamics and states of the environment. Moreover, the search space of action sequences increases exponentially with the length of the trajectory. Thus, in such a large space, grounding language instructions into action sequences is challenging and requires numerous annotated trajectories to supervise the learning process. To learn from only sparse annotations, the SLHSP applies the learned skill space $\mathcal{Z}$ to the grounding of language instructions. A language instruction has conditioned the prior skill $p_\rho(z \mid g_j)$, parameterized by ρ, it is learned to estimate the distribution of skills that accomplish the instruction, and the generation of the action sequence is handled by the learned skill decoder p_θ. As illustrated in Figure 2, the target of the skill prior is the posterior distribution, computed by the skill encoder by encoding the corresponding state trajectory. Thus, the process of learning the skill prior p_ρ can be seen as the adaptation of the skill prior to the posterior q_ϕ. The overall adaptation process is presented in Algorithm 2.

Algorithm 2: Skill Prior Adaptation

Input: Parameters ρ, number of epochs N, learned skill encoder q_ϕ, and annotated dataset $\mathcal{D}^{\text{ann}}$
Output: Learned parameters ρ
Initialize the skill prior ρ.
for $n \leftarrow 0$ **to** N **do**
  repeat
    $\tau^{\text{ann}} \sim \mathcal{D}^{\text{ann}}$
    $(\tau^{\text{ann},1}, \tau^{\text{ann},2}, \dots) \leftarrow \tau^{\text{ann}}$
    for $\tau^{\text{ann},j}$ *in* $(\tau^{\text{ann},1}, \tau^{\text{ann},2}, \dots)$ **do**
      $(s_{m:n}, g_j) \leftarrow \tau^{\text{ann},j}$
      $\rho \leftarrow \rho - \alpha_\rho \nabla_\rho D_{\text{KL}}\Big(q_\phi(z \mid s_{m:n}) \mid p_\rho(z \mid g_j)\Big)$
    end
  until *the last episode* τ^{ann} *in* $\mathcal{D}^{\text{ann}}$;
end

Specifically, the policy minimizes the Kullback–Leibler divergence between the posterior $q_\phi(z \mid s_{m:n})$ and the language-conditioned skill prior $p_\rho(z \mid g_j)$ on the following annotated trajectories:

$$\mathcal{L}_{\text{adapt}}(\tau^{\text{ann},j}) = D_{\text{KL}}\big(q_\phi(z \mid s_{m:n}) \mid p_\rho(z \mid g_j)\big), \tag{6}$$

where the parameters ϕ of the skill encoder are fixed during the adaptation. Thus, the skill posterior $q_\phi(z \mid s_{m:n})$ learned from unannotated trajectories stays unchanged, and the language conditioned skill prior $p_\rho(z \mid g_j)$ optimizes its parameters ρ to minimize $\mathcal{L}_{\text{adapt}}$:

$$\underset{\rho}{\operatorname{argmin}}\, \mathbb{E}_{\tau^{\text{ann},j} \sim \mathcal{D}^{\text{ann}}} \mathcal{L}_{\text{adapt}}(\tau^{\text{ann},j}). \tag{7}$$

During the inference, the skill prior $p_\rho(z \mid g_j)$ and the skill decoder $p_\theta(a_t \mid s_t, a_{t-1}, z)$ together serve as a language-conditioned policy for the SLHSP. As an LLP, the SLHSP uses this language conditioned policy as a low-level policy that executes planned language instructions. Formally, this low-level policy is formulated as

$$\pi^l_{\rho,\theta} = p_\rho(z \mid g_j) \prod_{t=m}^{n} p_\theta(a_t \mid z, s_t, a_{t-1}), \tag{8}$$

With the low-level policy $\pi^l_{\rho,\theta}$, the SLHSP learns a high-level policy that breaks down the task goal into multiple language instructions; this is introduced in the next section.

3.4. Planning via Natural Language

The high-level policy of the SLHSP serves two main functions: segmentation of unannotated trajectories into subtasks for latent skill space learning, and decomposition of the task goal through language instructions into several subtasks for inference. For the trajectory segmentation, it learns a separator $\pi^{h,\text{se}}_\psi(y_t \mid s_{1:t}, a_{1:t}, I)$, parameterized by ψ, to estimate the distribution of $y_t \in \{0, 1\}$, indicating whether the step t is the end point of a subtask. For the generation of natural language instructions, it learns a subtask describer $\pi^{h,\text{de}}_\omega$, parameterized by ω. The describer policy samples an instruction $\hat{g}_j$ as

$$\hat{g}_j \sim \pi^{h,\text{de}}_\omega(g_j \mid g_{1:j-1}, I), \tag{9}$$

where j is the index of the current subtask.

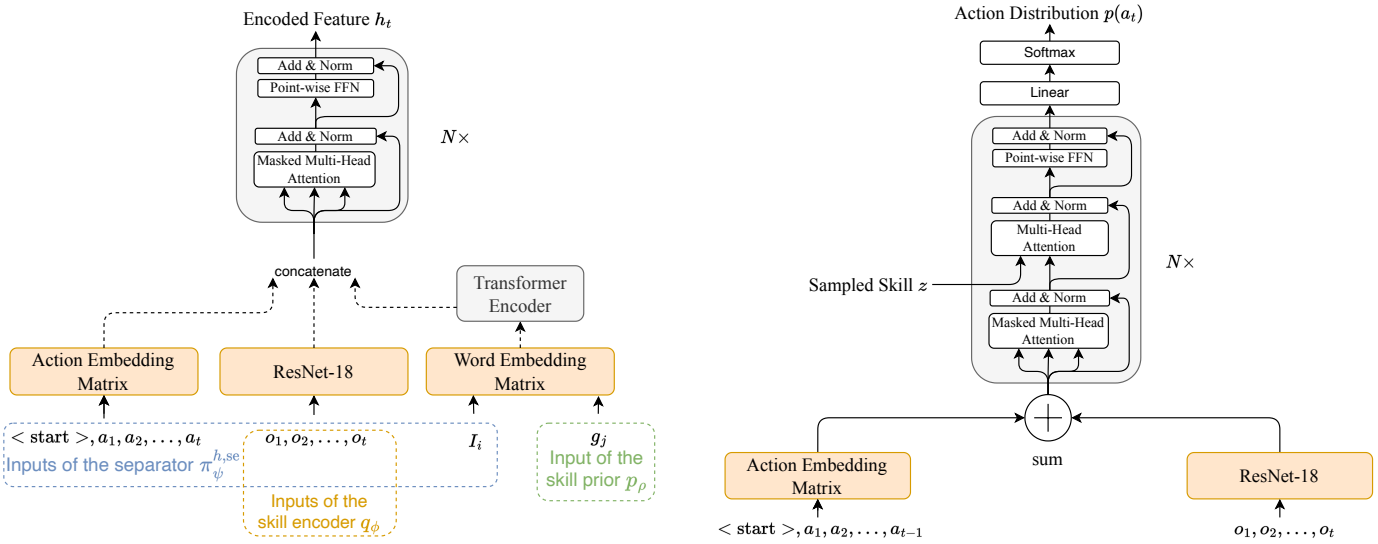
Specifically, the learning of the separator $\pi^{h,\text{se}}_\psi$ and the describer $\pi^{h,\text{de}}_\omega$ are supervised by limited annotated data in $\mathcal{D}^{\text{ann}}$. The describer $\pi^{h,\text{de}}_\omega(g_j \mid g_{1:j-1}, I)$ is initialized with a pre-trained T5-small model [34], following [12]. During the prediction of g_j, the task goal sentence I is concatenated with the previous subtask instructions $g_{1:j-1}$ to form a single sequence of words, and then the planner sequentially decodes words of g_j by attending to these input words. For the separator $\pi^{h,\text{se}}_\psi(y_t \mid s_{1:t}, a_{1:t}, I)$, a transformer model encodes the multi-modal feature sequence, which includes visual features, action features, and language features, and classifies whether step t is an end of a subtask. However, in an episode trajectory, there is only a small portion of end points of subtasks, thus the classification labels are severely unbalanced. This problem is tackled by adding the inverse class-type frequency of labels as a weight term to the classification loss.

Note that the focus of this paper is to learn and ground natural languages in a latent skill space that enables semi-supervised learning of an LLP. The specific design of the describer policy is orthogonal to our work. Consequently, our implementation of the describer is only an example; it can be replaced with a broad set of policies that decouple the task goal I into multiple language instructions.

3.5. Network Architectures

In this section, we introduce the key network architectures used by the SLHSP components: the separator $\pi^{h,\text{se}}_\psi$, the skill encoder q_ϕ, the skill prior p_ρ, the skill decoder p_θ, and the describer $\pi^{h,\text{de}}_\omega$. These components utilize transformer networks [35] as building blocks. The encoder network, shown in Figure 3a, fuses the multimodal features for the separator $\pi^{h,\text{se}}_\psi$, the skill encoder q_ϕ, and the skill prior p_ρ. The word embeddings of language inputs are first encoded by a transformer model and then the encoded language features are fused

with other features through the multimodal transformer encoder. Note that as the agent in the ALFRED benchmark has no access to the full environment states, but only to the egocentric visual observations, the skill decoder p_θ uses visual observations $o_{m:n}$ to estimate the information of $s_{m:n}$. For the skill encoder and the separator, the encoder network encodes the visual observations, actions, and task goal as inputs, while for the skill prior, the encoder network encodes only the language instruction g_j. The output feature matrix h_t of the encoder network is then used by linear layers to estimate the distribution of subtask endpoints (for the separator) or the distribution of skills (for the skill encoder and the skill prior). We found that the performance of the SLHSP benefits from longer observation and action history inputs to the decoder, and thus the skill decoder predicts each action conditioned on the sampled skills z, full histories of observations $o_{1:t}$ and actions $a_{1:t-1}$.



(**a**) Multimodal encoder network

(**b**) Skill decoder network

Figure 3. Network structures. (**a**) Multimodal encoder network structure of separator $\pi^{h,\text{se}}\psi$, skill prior p_ρ, and skill encoder q_ϕ, with different input options illustrated with colored boxes. (**b**) Skill decoder network structure with a transformer decoder.

Illustrated in Figure 3b, the skill decoder p_θ is implemented as a transformer decoder network. For each time step, the action embeddings are summed with the observation embeddings, which are computed based on a pre-trained ResNet18 model [36]. The summed features are fed as inputs to the transformer decoder and selectively attended to by the model. The latent code z is directly fed to the second layer multi-head attention of the transformer decoder. The output of the skill decoder network is the distribution over the environmental actions, which is discussed in Section 4.4. Finally, the describer $\pi_\omega^{h,\text{de}}$ is a fine-tuned T5-small model, and details can be found in [34].

4. Experiment

Through the experiment, we aim to answer the following questions. (1) Can the learned latent skill space improve performance on long-horizon tasks? (2) How many annotations are required to learn the SLHSP? (3) How are language instructions associated with skills in the latent skill space?

4.1. Dataset

We evaluated the SLHSP and baseline models using the ALFRED benchmark [18], which has a set of simulated environments with language-instructed long-horizon compositional indoor household tasks. The benchmark provides a dataset of 25 K expert trajectories (21 K for training), and each expert trajectory is segmented into a composition of several

subtasks $\tau = \{\tau^1, \ldots, \tau^j, \ldots\}$ (trajectories in ALFRED dataset contain on average 7.5 subtasks), each of which is annotated with a language instruction g_j. To construct a sparsely annotated dataset, the annotated trajectories are sampled from the ALFRED dataset with subtask segmentations and language instructions, while the unannotated trajectories are sampled task trajectories only. Typically, the number of annotated trajectories is significantly fewer than the number of unannotated trajectories. For instance, 10% of annotated trajectories contain 2000 annotated trajectories ($|\mathcal{D}^{\text{ann}}| = 2000$) and 19,000 unannotated trajectories ($|\mathcal{D}| = 19{,}000$). Specifically, the segmented subtasks in the dataset are categorized into eight different types: a navigation subtask (GoToLocation) and several object manipulation subtasks (Pickup, Put, Cool, Heat, Clean, Slice, Toggle). The training and validation splits of the dataset in [18] are followed in the experiments, and the validation split is further divided into *seen* and *unseen* folds based on whether the scenes or objects are shown to the agent during training. The number of trajectories in validation seen and unseen splits are 820 and 821, respectively.

As mentioned by [12], the annotated instruction for the navigation subtask is collected post hoc, after the agent has completed the subtask. For instance, it is impossible for the planner to generate the instruction "*Turn right, take a step, turn right again, move to the microwave*". before the agent has explored the layout of the room. To plan with instructions, we follow [12], modifying this type of instruction with the template "*Go to the [object]*". After the modification, the above example instruction is written as "*Go to the microwave*".

4.2. Baselines

We compared the SLHSP with models that are also designed to learn from sparse language annotations and inference with only task goals. **(SL)**^{3} [12] is a semi-supervised method that learns a language labeling policy from a small amount of annotated data and augments unannotated data with language instructions. It is similar to the SLHSP in terms of learning from sparse annotations. However, the SLHSP bypasses the data augmentation step by directly grounding the language to a learned latent skill space. A variant of (SL)^3 that uses ground truth segmentations (see [12] for more details), which is denoted as **(SL)**^{3}**(GT** α**)**, is also included as a baseline model. To analyze the improvement of the data efficiency of language annotation that is caused by our method, the constructed flat model **seq2seq** and hierarchical model **seq2seq2seq** [12] are also adopted as baselines. The SLHSP and all these baselines apply the same input and output information during training and inference.

Various state-of-the-art models (including those utilizing extra input information) for long-horizon instruction following tasks in ALFRED are also included as baselines. Although these methods have different inputs or training supervisions compared to SLHSP, they are compared with SLHSP to show the relative performance that can be achieved with only limited supervision. **MOCA** [19] and **ET** [37] use ground truth segmentation and subtask instruction in both training and inference. Instead of using an egocentric view of the environment, **LWIT** [21] takes multiple egocentric views as input to augment the observation at each step. **HiTUT** [4] designs unified transformers with self-monitoring and backtracking mechanisms to achieve strong performance in unseen scenarios. **HLSM** [3] and **FILM** [6] utilize the depth and semantic information of visual observation to construct a persistent semantic map, which guides the agent with holistic information. **EmBERT** [5] fine-tunes a BERT model [38] for language-conditioned long-horizon task completion. Most of these methods use additional information (e.g., step-by-step instruction, rich visual information that includes depth and visual semantics) to either supervise the learning process or facilitate the inference process. A detailed comparison of these models is reported in Table 1.

Table 1. Evaluated models in the experiments. ✓ denotes the use of corresponding attribute, and ✗ refers to the absence of corresponding attribute.

Model	During Training					During Inference			
	Goals	Instructions	Segmentations	Multi-View	Depth	Goals	Instructions	Multi-View	Depth
SLHSP (ours)	✓	✓ (10%)	✓ (10%)	✗	✗	✓	✗	✗	✗
$(SL)^3$ [12]	✓	✓ (10%)	✗	✗	✗	✓	✗	✗	✗
seq2seq [12]	✓	✗	✗	✗	✗	✓	✗	✗	✗
seq2seq2seq [12]	✓	✓	✗	✗	✗	✓	✗	✗	✗
E.T. [37]	✓	✓	✓	✗	✗	✓	✓	✗	✗
MOCA [19]	✓	✓	✓	✗	✗	✓	✓	✗	✗
LWIT (Multi view) [21]	✓	✓	✓	✓	✗	✓	✓	✓	✗
LWIT (Single view) [21]	✓	✓	✓	✗	✗	✓	✓	✗	✗
HiTUT [4]	✓	✗	✓	✗	✓	✓	✗	✗	✓
HLSM [3]	✓	✗	✓	✗	✓	✓	✗	✗	✓
FILM [6]	✓	✗	✓	✗	✓	✓	✗	✗	✓
EmBERT [5]	✓	✓	✓	✓	✗	✓	✓	✓	✗

4.3. Model Variants

To analyze the data efficiency and the performance of the SLHSP, several variants of the SLHSP are proposed for comparison. As all the components are indispensable for the SLHSP to work, instead of analyzing it through ablation studies, these variants are constructed from the perspective of available training data. In the experiments, different numbers of annotations are used in the learning of SLHSP. The annotation of a trajectory comprises two parts: the *segmentation of the task trajectory* and the *annotation of the language instruction* to segmented trajectory segments. The annotated segmentations are for the learning of the separator policy, while the annotated instructions are for the grounding of instructions to skills. The annotation setting **10% seg + 1% ann** in the experiments denotes that the fractions of annotated segmentations and instructions are 10% and 1%, respectively. By default, the SLHSP uses the same number of segmentation and instruction annotations, and is denoted as **SLHSP(*x*%)**, where x is the fraction of annotated trajectories. For long-horizon task evaluation, the language as scaffold between the hierarchy of the SLHSP allows it to be augmented with other language-conditioned models in a modular way. This way, some planned instructions can be executed by other language-conditioned models to improve the overall performance. Similar to [12], the SLHSP augments its navigation abilities with a rule-based navigator policy with ground-truth environment information to assist the low-level policy in executing navigation-related subtasks, and the augmented model is denoted as **SLHSP + nav**.

4.4. Implementation Details

In ALFRED, the agent's action space is discrete and consists of 13 different actions, including five navigation-related actions (*MoveAhead*, *RotateRight*, *RotateLeft*, *LookUp*, *LookDown*) and seven interaction-related actions (e.g., *Pickup*, *Close*, *Put*, *Open*, *Slice*, *ToggleOn*, *ToggleOff*). To perform interaction-related actions, an interaction mask is required to select the object in the agent's view with which to interact. We follow MOCA in obtaining the object mask by first predicting the object class, and then convert the object class to a mask in view, through the mechanism of Object-Centric Localization [19]; there are 82 different object classes.

For the observation space, the visual observation of the agent consists of 300×300 RGB images. These images are then sent to a frozen ResNet18 model [36] to obtain feature maps of size $512 \times 7 \times 7$. The language and action inputs to the agent are encoded through trainable embedding matrices, whose dimensions are set to 768.

The number of layers of all transformer networks in SLHSP is set to 2, and the number of heads in all multi-head attention mechanisms is set to 8. For the learning process, both the skill extraction and skill prior adaptation processes are conducted for 20 epochs with a

batch size of 8. The optimizer adopted for learning is AdamW [39], and the learning rate is set to 1×10^{-4}, which is decayed to 1×10^{-5} after 10 epochs of learning. The regularization weight term β is set to 1×10^{-2}.

4.5. Evaluation Metrics

Models are evaluated through both online interaction with the environment simulator and offline comparison with expert trajectories. For the online evaluation, a task goal is achieved if object positions and state changes meet the requirements of the task goal, and a subtask is achieved if the object positions and state changes of the subtask are met, conditioned on the preceding expert sequence [18]. Specifically, the end-to-end task success rate

$$v_{\text{task}} = \frac{\text{\# achieved task goals}}{|\mathcal{D}_{\text{eval}}|}, \tag{10}$$

where $\mathcal{D}_{\text{eval}}$ is the dataset of tasks used for evaluation, and the subtask success rate

$$v_{\text{subtask}} = \frac{\text{\# achieved subtasks}}{\left|\{\tau_i^j | \tau_i^j \in \tau_i, \tau_i \in \mathcal{D}_{\text{eval}}\}\right|}, \tag{11}$$

are evaluated. The environment is initialized and tasks are defined with the settings defined in the dataset. As the information used by the evaluated models is different, it is unfair to directly compare the absolute value of their corresponding success rates. In Tables 2 and 3, the results of the evaluation are grouped into models that are trained with sparse annotations and inference with only the task goal, and models that use additional inputs in either training or inference. The latter group refers to the use of extra inputs, such as multi-view observation and depth observation, or the use of full set of annotations and segmentations to supervise the training. For the offline evaluation, the data efficiency is evaluated following [12], in which the offline success rate score v_{exact} is computed as the fraction of *exact match* of the predicted actions

$$v_{\text{exact}} = \frac{|\{\hat{a}_t \mid \hat{a}_t = a_t, a_t \in \tau_i, \tau_i \in \mathcal{D}_{\text{eval}}\}|}{|\{a_t \mid a_t \in \tau_i, \tau_i \in \mathcal{D}_{\text{eval}}\}|}, \tag{12}$$

where $\hat{a}_t$ is the predicted action.

In order to evaluate the effectiveness of an agent in completing a task, the path length weighted (PLW) score [18] is applied to discount the success rate, according to the relative path length to the expert trajectory. Formally, the PLW score p is computed as

$$p = \frac{\hat{w}}{\max(w^*, \hat{w})}, \tag{13}$$

where w^* is the path length (the number of action) of the expert trajectory, and $\hat{w}$ is the path length of the predicted path. In Table 2, the path length weighted success rate is obtained by simply multiplying the PLW score p with the corresponding task success rate v_{subtask}.

5. Results

5.1. Overall Performance

In this section, we analyze the question of whether the learned latent skill space improves performance on long-horizon tasks. Table 2 compares the SLHSP with baseline models that use only sparse annotations, and models that use either rich observations, such as multiple visual observations and depth information, or ground-truth segmentations and instructions. In Table 2a, both *(SL)*^{3} and *SLHSP* are trained with only 10% of annotations ($|\mathcal{D}|$ = 19,000, $|\mathcal{D}^{\text{ann}}|$ = 2000). The *seq2seq* and *seq2seq2seq* models are trained with 100% of annotations ($|\mathcal{D}| = 0$, $|\mathcal{D}^{\text{ann}}|$ = 21,000). We can see that none of the baseline methods complete an entire long-horizon task, but the SLHSP is able to successfully accomplish some tasks in this condition. Notably, the performance of (SL)^3 on v_{task} is mainly affected

by its navigation ability, as $(SL)^3$ has a relatively low success rate on the *GoTo* subtask (in Table 3), and 87% of subtasks in ALFRED are *GoTo*. $(SL)^3$ improves its performance on v_{task} if it is augmented with a rule-based policy for navigation, as reported in Table 2b. Similarly, when augmenting the SLHSP with a rule-based navigator for navigation, the SLHSP is on par with some state-of-the-art approaches. Note that as the implementation details of $(SL)^3$ + planner and $(SL)^3$ + HLSM in [12] are not available, their implementations are different from our navigator policy. The performances of these augmented models are affected by the ability of the rule-based policy to both navigate the agent and coordinate with the learned policy. By comparing "SLHSP + nav (10%)" with the other models in Table 2b, the SLHSP shows that it has the potential to be augmented to demonstrate performance that is on par with the other state-of-the-art methods.

Table 2. Task success rate v_{task} evaluation on ALFRED benchmark for models trained with sparse annotations and additional inputs, reported for seen and unseen folds. Values in parentheses are PLW success rates $p \times v_{\text{task}}$, and boldface indicates highest in column. "-" denotes unreported scores.

(a) Sparse Annotation		
Model	**Unseen v_{task}**	**Seen v_{task}**
SLHSP (10%) (ours)	**1.2 (0.5)**	**1.7 (0.7)**
$(SL)^3$ (10%) [12]	0.0 (-)	-
seq2seq [12]	0.0 (-)	-
seq2seq2seq [12]	0.0 (-)	-
(b) Additional Inputs		
Model	**Unseen v_{task}**	**Seen v_{task}**
SLHSP + nav (10%) (ours)	23.7(18.5)	30.9 (24.9)
$(SL)^3$ + planner (10%) [12]	**40.4 (-)**	-
$(SL)^3$ + HLSM (10%) [12]	15.5 (-)	-
HLSM [3]	18.3 (-)	29.6 (-)
FILM [6]	20.1 (-)	24.6 (-)
EmBERT [5]	5.7 (3.1)	37.4 (28.8)
LWIT (Multi view) [21]	9.7 (7.3)	33.7 (28.4)
E.T. [37]	7.3 (3.3)	**46.6 (32.3)**
HiTUT [4]	12.4 (6.9)	25.2 (12.2)
MOCA [19]	5.4 (3.2)	25.9 (19.0)

Table 3. Subgoal success rate v_{subtask} on an unseen validation fold. The highest values per fold and subtask are shown in boldface. "-" denotes scores that are not reported.

Model	**Goto**	**Pickup**	**Put**	**Cool**	**Heat**	**Clean**	**Slice**	**Toggle**	**Avg.**
Sparse annotation									
SLHSP (10%) (ours)	**40**	**63**	45	**95**	**98**	43	42	20	**56**
$(SL)^3$ (10%) [12]	13	50	**48**	75	74	**56**	**54**	**32**	50
seq2seq2seq [12]	15	29	42	69	58	15	50	**32**	39
seq2seq [12]	14	20	15	33	64	16	25	13	25
Additional inputs									
SLHSP (100%) (ours)	40	66	46	99	**100**	42	48	31	59
$(SL)^3$ (100%) [12]	15	50	45	82	75	68	55	32	53
LWIT (Multi view) [21]	39	**79**	66	94	95	68	**85**	66	**74**
HiTUT [4]	-	71	**69**	**100**	97	**91**	78	58	-
MOCA [19]	32	44	39	38	86	71	55	11	47
E.T. [37]	**45**	67	66	**100**	97	**91**	53	**72**	**74**

Accomplishing the entire long-horizon task is challenging. Additionally, we assess the ability of models to complete the next subtask, conditioned on the preceding expert sequence, and report the results of subtask evaluation in Table 3. The SLHSP improves on v_{subtask} by 6% compared with the $(SL)^3$. On the navigation subtask *GoTo*, the SLHSP shows a marginal improvement of 27% and 25% over the $(SL)^3$ under conditions of 10% and 100% annotations,, respectively. This indicates that navigation-related instructions can be better associated with action sequences by utilizing learned reusable skills, compared with directly mapping instructions to action sequences. Compared with the other state-of-the-art methods, the SLHSP shows comparable performances in most subtask types.

5.2. Data Efficiency

In this section, we analyze the data efficiency of the SLHSP. As illustrated in Figure 4, the performance scores of the SLHSP models learned with 10% and 100% annotations are nearly the same, which indicates that only sparse annotations are adequate for the SLHSP to obtain a reasonable performance. Compared with other methods, the SLHSP learned with only 5% of language annotations outperforms the $(SL)^3$ learned with full (100%) language annotations. Moreover, with 40% of language annotations, the SLHSP receives nearly the same performance as the $(SL)^3$(GT α) model, which has access to ground truth segmentations, and leaves the seq2seq and seq2seq2seq baselines far behind. The performance difference of SLHSP learned with 10% and 100% annotations is presented in Table 4, with less than 1.4% and 1.3% in the settings of 10% and 100% segmentations, respectively. These results shows that the learned skill decoder undertakes the majority of the work for the grounding process, so that the language instructions can be efficiently grounded to actions with a small amount of annotated data.

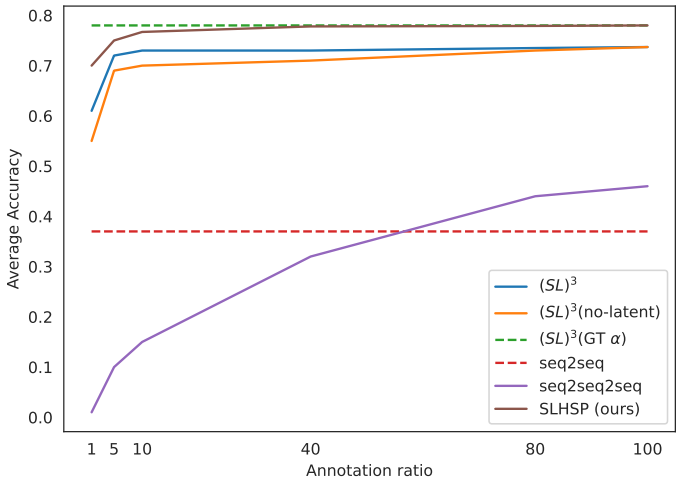


Figure 4. Offline subtask success rate v_{exact}. The amount of annotated trajectories denotes the fraction of trajectories of all annotated trajectories. The dashed lines illustrate the performance of models that use 100% of annotated trajectories.

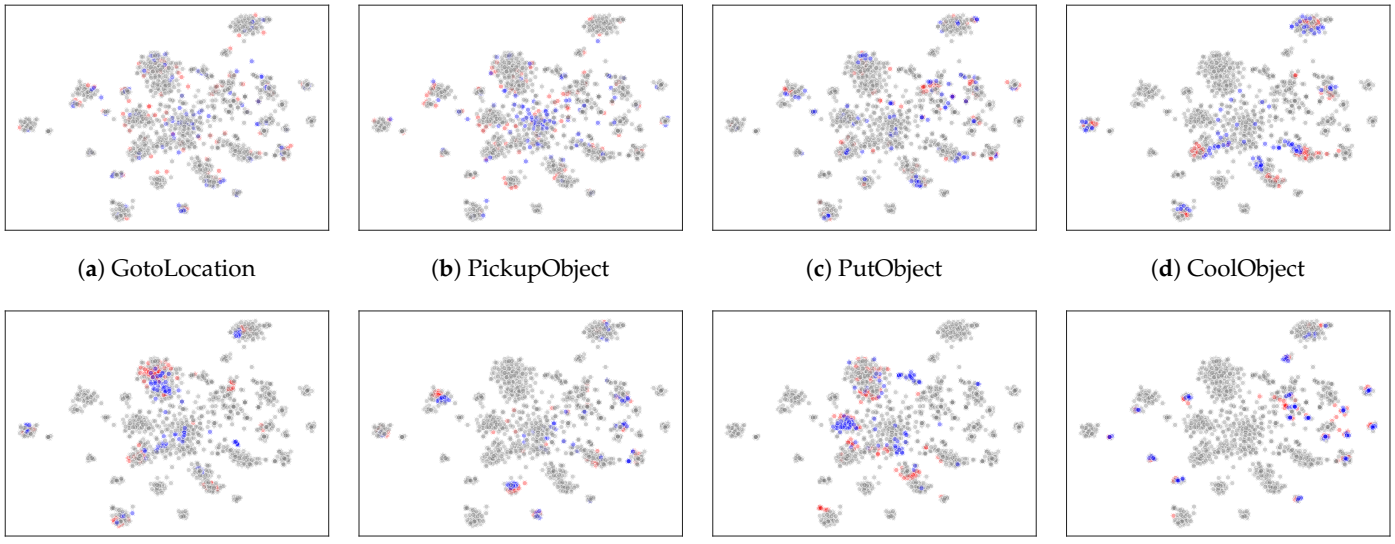
Table 4. Offline success rate v_{exact} of SLHSP learned with different training data settings.

Annotation Settings	Average Success Rate
10% seg + 1% ann	70.2
10% seg + 5% ann	75.0
10% seg + 10% ann	76.7
100% seg + 10% ann	77.0
10% seg + 100% ann	78.1
100% seg + 100% ann	78.3

As mentioned in Section 4.3, the annotation contains two parts: language instructions and trajectory segmentations. In Table 4, we analyze the influence of these two parts on the performance of SLHSP. It reports the results of offline success rate averaged over three random seeds, where the "seg" represents ground truth segmentations for learning separator policy, and "ann" represents annotations for skill prior adaptation. The results show that increasing the annotation ratio of the sparsely annotated dataset is crucial to improve performance when there are less than 10% of annotated trajectories. From the perspective of the segmentations, the improvement of success rate brought by the increase from 10% to 100% is only 0.2%, which is tiny compared with the improvement brought by the increase in the number of language annotations. This finding suggests that learning the separator policy requires fewer labels than adapting the skill prior.

5.3. Latent Skill Space

To examine the association between language instructions and skills, we first analyzed the properties of the latent skill space. The latent skill space obtained through self-supervised learning is structurally organized. In Figure 5, gray points are samples not belonging to the plotted subtask type. Each plot shows sample points based on trajectories or instructions from a subtask type, with an equal number of sample points per subtask type. The t-SNE model is learned with perplexity of 20. We found that the sample points of the latent variable z form clusters. However, these clusters are not directly related to the type of subtasks. This is because a latent skill encodes a sequence of trajectory observations, so that these clusters are formed in accordance with the features of visual observations. As shown in Figure 5c–h, most of the highlighted samples are centered in a few clusters; this indicates that these subtask trajectories have similar visual observations. In contrast, trajectories in Figure 5a (GoToLocation) and Figure 5b (PickupObject) are more diverse in terms of the observation sequence, and the corresponding skill samples are scattered in nearly all the clusters.



(**a**) GotoLocation (**b**) PickupObject (**c**) PutObject (**d**) CoolObject

(**e**) HeatObject (**f**) CleanObject (**g**) SliceObject (**h**) ToggleObject

Figure 5. t-SNE visualization of adapted skill prior (red) and skill posterior (blue) samples.

Samples from the adapted skill prior are associated with samples from the skill encoder (skill posterior). In Figure 5c–h, most of the samples from the skill prior are distributed in the same clusters with the samples from the skill posterior. This indicates that the language instructions are correlated with the skills. However, in Figure 5a (GoToLocation) and Figure 5b (PickupObject), the patterns of the distributions of skill samples in these two types of subtasks are not explicit, and there are two possible reasons for this. First, the visual observations between samples are diverse. The associations between samples

are not centered in a few clusters. Second, the information loss caused by the dimension reduction of the t-SNE visualization hides some correlations between the skill prior and the skill posterior.

5.4. Case Study

For a more intuitive view of how SLHSP works in long-horizon instruction following tasks, we visualize two qualitative examples of the task execution trajectory in Figure 6. The SLHSP predicts a sequence of subtasks $\{\hat{g}_k\}_{k=1}^{k=K}$ via natural language and executes them sequentially. Navigation trajectories are denoted with thick lines, and key frames are illustrated around a map, with positions highlighted by circles. Line and circle colors correspond to subtasks. The agent's egocentric view range is shown as a white cone at each step. In the plot on the left, the agent successfully accomplishes the task. The agent switches smoothly between the navigation and manipulation subtasks. This indicates that the high-level separator policy accurately recognizes a subtask is accomplished, and the SLHSP switches to the instruction of the next subtask during inference. The plot on the right shows a failure case. Although the describer of the SLHSP generates appropriate instructions, the SLHSP agent failed to navigate to the garbage bin in the 5th subtask ("Go to the garbage bin"). This might be due to the inaccuracy of the skill prior distribution, meaning the sampled skill from the skill prior cannot reflect the intended instruction.

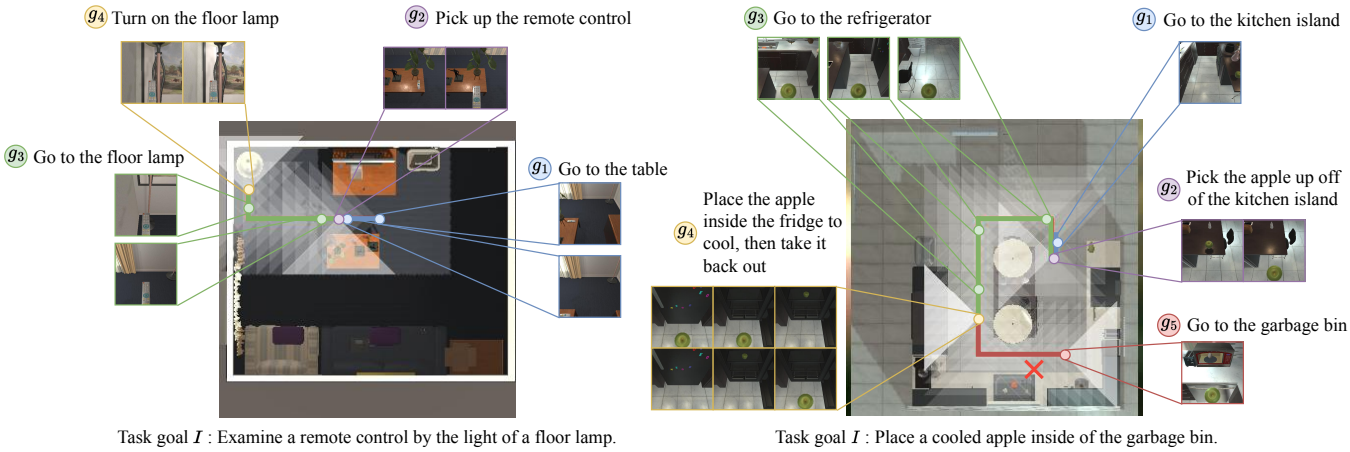


Figure 6. Qualitative results of online evaluation. Success (**left**) and failure (**right**) cases are shown for the agent navigating and manipulating objects to reach the task goal I.

6. Conclusions

We introduce an SLHSP for long-horizon compositional decision-making tasks. The SLHSP is a latent language policy learning framework for utilizing unannotated datasets to improve the data efficiency of language annotations. The SLHSP learns a latent skill space from unannotated data to regularize the action sequence space that the language grounds to, so that only sparse annotations are adequate to learn the mapping between the languages and action sequences.

While progress has been made in building embodied agents that can understand natural language and make decisions, there is still much work to be done to handle the complexity and diversity of real-world scenarios. The application of a SLHSP accelerates the accomplishment of real-life instruction-following robotics by diminishing the amount of supervisory data required for learning, which saves expensive labeling efforts. In the future, to make our approach more applicable in real-life instruction-following robotics, we intend to combine large language models with our semi-supervised method to further improve generalization ability and data efficiency, for example, through few-shot or even zero-shot grounding of language instructions.

Author Contributions: Conceptualization, B.Z. and C.Z.; Formal analysis, B.Z.; Funding acquisition, Z.H.; Investigation, B.Z.; Methodology, B.Z.; Project administration, Z.H.; Supervision, C.Z. and Z.H.; Validation, B.Z.; Visualization, B.Z.; Writing—original draft, B.Z.; Writing—review & editing, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Project grant number 2018YFE0205503.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/askforalfred/alfred.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Andreas, J.; Klein, D.; Levine, S. Learning with Latent Language. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; pp. 2166–2179. [CrossRef]
2. Jiang, Y.; Gu, S.; Murphy, K.; Finn, C. Language as an Abstraction for Hierarchical Deep Reinforcement Learning. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 9414–9426.
3. Blukis, V.; Paxton, C.; Fox, D.; Garg, A.; Artzi, Y. A Persistent Spatial Semantic Representation for High-level Natural Language Instruction Execution. In Proceedings of the Conference on Robot Learning, London, UK, 8–11 November 2021; Volume 164, pp. 706–717.
4. Zhang, Y.; Chai, J. Hierarchical Task Learning from Language Instructions with Unified Transformers and Self-Monitoring. In Proceedings of the International Joint Conference on Natural Language Processing, Virtual, 1–6 August 2021; pp. 4202–4213. [CrossRef]
5. Suglia, A.; Gao, Q.; Thomason, J.; Thattai, G.; Sukhatme, G.S. Embodied BERT: A Transformer Model for Embodied, Language-guided Visual Task Completion. *arXiv* **2021**, arXiv:2108.04927.
6. Min, S.Y.; Chaplot, D.S.; Ravikumar, P.K.; Bisk, Y.; Salakhutdinov, R. FILM: Following Instructions in Language with Modular Methods. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2022.
7. Huang, W.; Abbeel, P.; Pathak, D.; Mordatch, I. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Volume 162, pp. 9118–9147.
8. Li, S.; Puig, X.; Paxton, C.; Du, Y.; Wang, C.; Fan, L.; Chen, T.; Huang, D.; Akyürek, E.; Anandkumar, A.; et al. Pre-Trained Language Models for Interactive Decision-Making. *arXiv* **2022**, arXiv:2202.01771.
9. Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. *arXiv* **2022**, arXiv:2207.05608.
10. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv* **2022**, arXiv:2204.01691.
11. Vaezipoor, P.; Li, A.C.; Icarte, R.T.; McIlraith, S.A. LTL2Action: Generalizing LTL Instructions for Multi-Task RL. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; Volume 139, pp. 10497–10508.
12. Sharma, P.; Torralba, A.; Andreas, J. Skill Induction and Planning with Latent Language. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; pp. 1713–1726. [CrossRef]
13. Xiao, T.; Chan, H.; Sermanet, P.; Wahid, A.; Brohan, A.; Hausman, K.; Levine, S.; Tompson, J. Robotic Skill Acquisition via Instruction Augmentation with Vision-Language Models. *arXiv* **2022**, arXiv:2211.11736.
14. Cideron, G.; Seurin, M.; Strub, F.; Pietquin, O. HIGhER: Improving instruction following with Hindsight Generation for Experience Replay. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Canberra, Australia, 1–4 December 2020; pp. 225–232. [CrossRef]
15. Röder, F.; Eppe, M.; Wermter, S. Grounding Hindsight Instructions in Multi-Goal Reinforcement Learning for Robotics. *arXiv* **2022**, arXiv:2204.04308.
16. Lee, D.H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Proceedings of the Workshop on Challenges in Representation Learning, ICML, Atlanta, GA, USA, 16–21 June 2013; Voume 3, p. 896.
17. Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.E.; McGuinness, K. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. In Proceedings of the International Joint Conference on Neural Networks, Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]

18. Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; Fox, D. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10737–10746. [CrossRef]
19. Singh, K.P.; Bhambri, S.; Kim, B.; Mottaghi, R.; Choi, J. Factorizing Perception and Policy for Interactive Instruction Following. In Proceedings of the International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 1868–1877. [CrossRef]
20. Corona, R.; Fried, D.; Devin, C.; Klein, D.; Darrell, T. Modular Networks for Compositional Instruction Following. In Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 1033–1040. [CrossRef]
21. Nguyen, V.; Suganuma, M.; Okatani, T. Look Wide and Interpret Twice: Improving Performance on Interactive Instruction-following Tasks. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 19–27 August 2021; pp. 923–930. [CrossRef]
22. Jansen, P.A. Visually-Grounded Planning without Vision: Language Models Infer Detailed Plans from High-level Instructions. In Proceedings of the Empirical Methods in Natural Language Processing, Online, 16–20 November 2020; pp. 4412–4417. [CrossRef]
23. Artzi, Y.; Zettlemoyer, L. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Trans. Assoc. Comput. Linguist.* **2013**, *1*, 49–62. [CrossRef]
24. Patel, R.; Pavlick, E.; Tellex, S. Grounding Language to Non-Markovian Tasks with No Supervision of Task Specifications. In Proceedings of the Robotics: Science and Systems, Pittsburgh, PA, USA, 26–30 June 2020. [CrossRef]
25. Nair, S.; Mitchell, E.; Chen, K.; Ichter, B.; Savarese, S.; Finn, C. Learning Language-Conditioned Robot Behavior from Offline Data and Crowd-Sourced Annotation. In Proceedings of the Conference on Robot Learning, London, UK , 8–11 November 2021; Volume 164; pp. 1303–1315.
26. Mees, O.; Hermann, L.; Burgard, W. What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data. *IEEE Robot. Autom. Lett.* **2022**, *7*, 11205–11212. [CrossRef]
27. Andrychowicz, M.; Crow, D.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; Zaremba, W. Hindsight Experience Replay. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5048–5058.
28. Akakzia, A.; Colas, C.; Oudeyer, P.; Chetouani, M.; Sigaud, O. Grounding Language to Autonomously-Acquired Skills via Goal Generation. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
29. Carta, T.; Lamprier, S.; Oudeyer, P.; Sigaud, O. EAGER: Asking and Answering Questions for Automatic Reward Shaping in Language-guided RL. *arXiv* **2022**, arXiv:2206.09674.
30. Lynch, C.; Sermanet, P. Grounding Language in Play. *arXiv* **2020**, arXiv:2005.07648.
31. Pertsch, K.; Lee, Y.; Lim, J.J. Accelerating Reinforcement Learning with Learned Skill Priors. In Proceedings of the Conference on Robot Learning, Cambridge, MA, USA, 16–18 November 2020; Volume 155, pp. 188–204.
32. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
33. Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.P.; Glorot, X.; Botvinick, M.M.; Mohamed, S.; Lerchner, A. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
34. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 140:1–140:67.
35. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
37. Pashevich, A.; Schmid, C.; Sun, C. Episodic Transformer for Vision-and-Language Navigation. In Proceedings of the International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 15922–15932. [CrossRef]
38. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [CrossRef]
39. Loshchilov, I.; Hutter, F. Fixing Weight Decay Regularization in Adam. *arXiv* **2017**, arXiv:1711.05101.