Learning Effective Subgoals with Multi-Task Hierarchical Reinforcement Learning

Dagui Chen[1] , **Qi Yan**[1] , **Shangqi Guo**[1] , **Zhile Yang**[1] , **Xin Su**[1] and **Feng Chen**[1]
[1] Tsinghua University
{cdg16, q_yan15, gsq15, yzl18, suxin16}@mails.tsinghua.edu.cn, chenfeng@mail.tsinghua.edu.cn

Abstract

Hierarchical reinforcement learning (HRL) is a promising direction of solving complex tasks, and the success of it is determined by the acquisition of effective and semantically distinct subgoals. Most existing methods try to achieve this by introducing manual task-specific design and prior knowledge, which limits the generality of methods and the transferability of well-learned policies. We argue that this challenge is mainly because the single-task optimization cannot provide enough motivation for learning effective subgoals, and it is necessary to incorporate a multi-task setting into the HRL framework. In this paper, we propose a novel general multi-task HRL framework which is shared among different tasks and can automatically extract meaningful and transferable subgoals from multiple related tasks without requiring prior knowledge. Our framework consists of a task decomposition network which decomposes complex tasks into sequential abstract subgoals in a latent embedding space, and a parameterized skill network which manipulates atomic actions to yield the agent's abstract state close to the matching subgoal. Extensive experiments on several challenging environments with extremely sparse reward demonstrates that our method does learn effective subgoals and provides superior performance over strong baselines.

1 Introduction

Hierarchical reinforcement learning (HRL) has proven capable of extending traditional reinforcement learning (RL) to complex tasks with long-term credit assignment [Sutton *et al.*, 1999]. Nevertheless, whether HRL works depends on *whether effective subgoals can be obtained*. An **effective subgoal** should contain the following attributes: (i) **Temporally abstracted**. The subgoal should represent a temporal abstraction of the agent's behavior in a certain period. (ii) **Shared**. The effective subgoal must be shared by other related tasks or by different episodes of the same task. The hierarchical policies with such subgoals enable RL agents to plan or explore at different time scales, which improves the agent's exploration efficiency [Nachum *et al.*, 2018]. Besides, The well-learned low-level policies that are used to handle subgoals is independent of entire tasks and can be transferred to previously unseen tasks [Barto and Mahadevan, 2003].

Most previous HRL approaches [Dayan and Hinton, 1993; Parr and Russell, 1998; McGovern and Barto, 2001] attempt to obtain effective subgoals by introducing some degree of manual task-specific design, including domain knowledge for architecture and assumptions about effective subgoals. Although these artificial designs provide encouraging success, they also limit the generality of methods and the transferability of well-learned policies. For example, the "bottleneck" states, i.e., states that are frequently visited on system trajectory, are used as subgoals in [Stolle and Precup, 2002]. Such a setting assumes that all bottleneck states are critical states, but it is not applicable in many cases. While a recent work [Bacon *et al.*, 2017] has shown the potential for automatically learning subgoals in an end-to-end fashion, it requires the regularisers [Vezhnevets *et al.*, 2016] to prevent degradation into a trivial solution.

In this paper, we argue that one critical reason why it is difficult to design an automatic HRL learning framework is that the single-task optimization that most prior HRL works focus on cannot provide enough motivation for learning effective subgoals. In contrast, we propose a novel perspective for designing HRLs: it is necessary to incorporate multi-task learning (MTL) [Zhang and Yang, 2017] into the HRL framework. The core idea is inspired by the fact that discovering meaningful and effective subgoals can facilitate improving the overall multi-task performance. Therefore, aiming to solve multiple related tasks simultaneously can motivate the agent to learn effective subgoals. Besides, such the multi-task setting can reduce the dependence on task-specific prior knowledge, enabling us to design a general framework scaling to various tasks.

To enable HRL to benefit from multi-task optimization, we further propose a novel HRL architecture (see Figure 2a) with two levels of hierarchies, i.e., a high-level Task Decomposition Network (TDN) and a low-level Parameterized Skill Network (PSN). Different from previous methods focusing on a single task, our TDN is required to be able to generalize over tasks. It decomposes complex tasks into sequential subgoals in a latent embedding space conditioned on the current state and task information. The TDN's action (i.e., subgoal) is associated with the target state that the PSN is expected to achieve, and the TDN's optimization is motivated by the

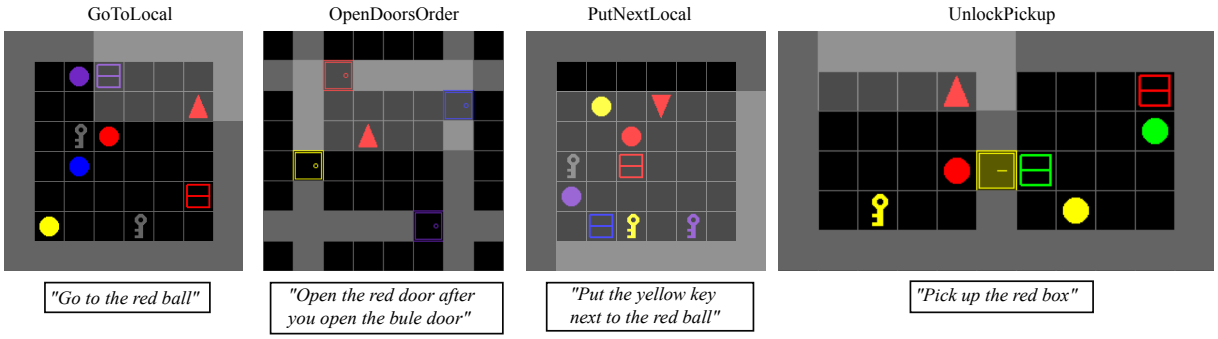

Figure 1: Four environments in BabyAI to evaluate our algorithm. Each environment contains a set of tasks specified by instructions. The text in each rectangle shows an example of instructions. The shaded grids represent the agent's current observations. The details refer to Section 5.

overall performance over a distribution of tasks. As for PSN, it is responsible for executing subgoals parameterized by the TDN's action. Through the intrinsic reward function, the PSN is encouraged to yield a state close to the desired subgoal. In our framework, the PSN is shared by all subgoals, and it does not need to know the number of subgoals and meaning of each subgoal in advance.

Concretely, our contributions are three-fold. (i) We propose a new perspective for the HRL design, i.e., learning a shared hierarchy of policy through multi-task setting, which enables the HRL to automatically acquire effective subgoals. (ii) We present a general HRL architecture, namely GMHRL (i.e., General and Multi-task), that is suitable for different tasks and requires little manual task-specific design. The comparative experiments in several environments verify our method's superiority, and the visualization results show the well-learned effective subgoals. (iii) Considering that learning the PSN with high-dimension subgoals requires extensive training, we propose a novel *reachable subgoals training* method, in which only subgoals that can be achieved are used for training. The method avoids massive invalid training, thereby improving the learning efficiency significantly.

2 Multi-task Optimization

The optimization problem we would like to solve is conditioned on the multiple tasks with instructions. Each instruction I encodes a specific task and is sampled from a distribution over tasks $\rho(I)$. Formally, we formulate such multi-task setting as a Multi-task Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, which consists of an instruction space $\mathcal{I}$, a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{P}(s'|s,a)$ and a reward function $\mathcal{R}(r|s,a,I)$, where (s,a,s',r) are state, action, next state and reward, respectively.

The agent receives an instruction I describing the current task at the beginning of each episode and then interacts with the task for $T(I)$ timesteps (i.e., the predefined maximum episode length) according to its policy π_θ which is parameterized by θ. A policy is a mapping from an instruction-following history $(I, s_0, a_0, r_0, s_1, \ldots, s_{t-1})$ to the next action a_t. Different from the traditional RL policy, the policy π_θ in multi-task MDP is shared among multiple tasks. Moreover, the optimization objective is to find a shared parameter vector θ^* to maximize the expected return over the distribution of instructions

$$\theta^* = \arg\max_\theta \mathbb{E}_{I\sim\rho(I)}\left[\sum_{t=0}^{T(I)-1} r_t \middle| \pi_\theta\right]. \quad (1)$$

For a hierarchy of policy, targeting Equation 1 as its optimization objective can facilitate learning effective subgoals. On the other hand, by comparing the overall performance on multiple related tasks that are previously unseen, we can better evaluate the transferability and effectiveness of the well-learned subgoals.

In practice, we select *BabyAI* platform [Chevalier-Boisvert *et al.*, 2018] as the testbed which comprises an extensive suite of instruction-following tasks with different difficulty levels, as shown in Figure 1. The platform is based on a partially observable 2D grid-world environment which is populated with various entities of different colors, such as the agent, balls, boxes, doors, and keys [Chevalier-Boisvert *et al.*, 2018]. The agent can not only navigate the world but also manipulate the world state, such as pick up or move around the objects, toggle the boxes and use keys with matching color to unlock the doors. Each environment contains various randomly generated related tasks which are specified by instructions.

Here, the instruction takes a simple and natural form, i.e., the language phrase that describes only the ultimate objective of the task. Such instruction coding is generic and readily available. All tasks in our setting are characterized by sparse and extremely delayed rewards, that is, the agent receives a positive reward signal only when the agent executes the instruction correctly within the time step limit, otherwise zero. Such tasks have proven to be very challenging [Andreas *et al.*, 2017; Oh *et al.*, 2017] and require the ability to perform exploratory navigation as well as complex sequences of object manipulation. Therefore, HRL becomes a natural solution.

3 GMHRL Framework

3.1 Main Design

As shown in Figure 2a, our overall architecture consists of an observation embedding module, a high-level TDN π^H, a low-level PSN π^L. The embedding module maps the original form of image observation input o_t to the abstract state

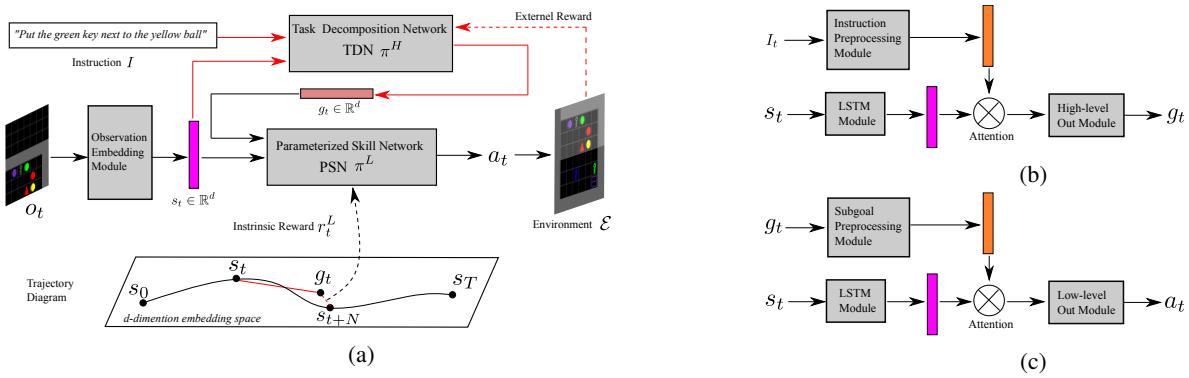

Figure 2: (a) The overall architecture of GMHRL. The below trajectory diagram in d-dimension embedding space illustrates our intrinsic reward setting. The subgoal g_t is generated by the TDN and remain fixed for N steps. The PSN aims to make the state after N steps s_{t+N} closer to g_t. (b) The network structure of TDN. (c) The network structure of PSN.

vector s_t in a d-dimension embedding space, where standard metrics can be applied to measure the distance between two states. At each time step t, the PSN observes current state s_t and produces atomic actions to interact with the environment, conditioned on the subgoal g_t that it receives from the TDN. The intrinsic reward function encourages the PSN to take actions to bring the future state closer to the desired subgoal. As for TDN, it outputs a target state (i.e., subgoal g_t) at the latent embedding spaces according to analyzing current state s_t and the task coding I. In this way, the TDN breaks up the complex task into a sequence of subgoals, which compels the agent to execute suitable sequential behaviors to fulfill the entire task. Furthermore, following the common setting in goal-conditioned HRL [Nachum *et al.*, 2018; Vezhnevets *et al.*, 2017], the TDN make decisions at a slower and fixed timescale. In this work, the subgoal g_t is generated by the TDN when $t = 0, N, 2N, \cdots$, otherwise it remain fixed $g_t = g_{t-1}$.

Concretely, our framework contains the following significant characteristics. (i) Since the number and meaning of subgoals vary among different tasks, we do not consider constructing a sub-network for each subgoal. This technique is popular in previous HRL methods [Sutton *et al.*, 1999; Stolle and Precup, 2002; Bacon *et al.*, 2017]. In contrast, we model our low-level policy as a skill network parameterized by subgoals. It is shared by all subgoals and does not require manual assumptions for effective subgoals. (ii) Compared to traditional methods with single task setting, the TDN is dedicated to optimizing the overall performance over a distribution of tasks (see Equation 1). It receives instructions as input and can handle previously unseen tasks. (iii) We represent the state and subgoal in a latent embedding space instead of in the raw form as used in [Nachum *et al.*, 2018], which means that our method can be applied to tasks with different forms of observation (e.g., visual image).

3.2 Reward Settings

In our framework, the PSN does not care about how to implement the entire task. It only needs to master the skill to reach the target state parameterized by a subgoal, which is the reason for its name (i.e., PSN). This task-independent feature allows the PSN to be transferred to other related tasks. The PSN's optimization is motivated by an intrinsic reward function that is also parameterized by subgoals. Different from the directional intrinsic reward function which has been studied previously in [Schaul *et al.*, 2015; Vezhnevets *et al.*, 2017], our intrinsic reward function directly measures the distance between s_t and g_t, which can be written as follow,

$$r^L(s_t, a_t, s_{t+1}, g_t) = -D(s_{t+1}, g_t) \tag{2}$$

where D represents a metric in the d-dimension embedding space. Here we use a simple Euclidean distance, and we also compare other distance function in the ablation experiment. This reward setting can motivate the subgoal to be associated with the target state. Besides, because the subgoal g_t remains fixed for N steps, the PSN will learn how to yield the N-step subsequent states s_{t+N} closer to the matching subgoal, as shown in the trajectory diagram in Figure 2a.

Compared with the PSN, the TDN aims to optimize the overall performance over a distribution of tasks. Therefore, the reward function of the TDN is associated with the external environment reward. Besides, considering that the TDN operates at a frequency of N steps, we set its reward function as $r_t^H = \sum_{j=t}^{t+N-1} \mathcal{R}(r|s_j, a_j, I)$ when $t = 0, N, 2N, \cdots$.

3.3 Learning

The PSN π^L can be directly trained using standard policy gradient method or TD learning. Here we choose A2C [Mnih *et al.*, 2016] algorithm with the stochastic policy gradient to optimize the network's parameters θ^L. A critical distinction from the standard RL is that our PSN generalizes not only over states but also over subgoals. Inspired by [Schaul *et al.*, 2015], we develop a *universal* stochastic policy gradient method which can be written as follow

$$\Delta\theta^L = A^L(s_t, g_t, a_t)\nabla_{\theta^L} \log \pi^L(a_t|s_t, g_t; \theta^L), \tag{3}$$

where $A^L(s_t, g_t, a_t)$ is the universal advantage function, computed from the internal critic of the PSN.

Since the TDN's action is a continuous d-dimension vector, we model the optimization of the TDN as a continuous control. One of the most natural choice to update its parameters θ^H is deterministic policy gradient (e.g., DDPG [Lillicrap *et al.*, 2015]). Every N time steps, the TDN transition

$(I, s_t, g_t, r_t^H, s_{t+N})$ is stored in a memory for training. Similarly, our TDN needs to generalize over instructions, so we also develop a *universal* deterministic gradient method for the TDN's learning. The update rule is

$$\Delta\theta^H = \nabla_g Q^H(s_t, I, g)\big|_{g=\pi^H(s_t,I)} \nabla_{\theta^H} \pi^H(s_t, I; \theta^H), \quad (4)$$

where $Q^H(s_t, I, g)$ is the universal action-value function, computed from the internal critic of the TDN.

Finally, both high-level and low-level policies share the same observation embedding module whose parameters are updated by the two policy update algorithms simultaneously. Besides, we add a pixel prediction head following this embedding module as the auxiliary task which predicts the future observation o_{t+1} conditioned on current state s_t and action a_t. This technique has proven capable of helping achieve a powerful feature representation [Jaderberg *et al.*, 2016].

3.4 Reachable Subgoal Training

Although our PSN is general and can generalize over subgoals, it suffers from low learning efficiency. Previous HRL methods that focus on decomposing tasks [Dayan and Hinton, 1993; Vezhnevets *et al.*, 2017] also face this challenge. One of the important reasons is that the high-level action space is much larger than the target state space that the agent can reach. If the high-level policy always set unreachable subgoals to the PSN, the PSN learning efficiency is significantly reduced. To overcome this challenge, we propose a novel *reachable subgoal training* method. In practice, we modify the original TDN exploratory policy which adds noise to the deterministic action in DDPG to output a reachable subgoal with a high probability[1]. This method avoids a large amount of invalid low-level training, which in turn speeds up the overall learning, which is verified in our ablation experiment.

The ideal way to generate reachable subgoal is to learn a generative network. Here we adopt a simpler form. When the agent faces a new environment, a memory is created to store all states that the agent has seen in this environment. All states in this memory can be treated as reachable target states and sampled for the high-level action.

3.5 Network Detail

Because our TDN and PSN essentially deal with multiple tasks which are specified by subgoal I and instruction g, respectively, the two networks have similar network structures, as shown in Figure 2b and Figure 2c. The LSTM [Hochreiter and Schmidhuber, 1997] module is applied to process the state sequence. Because the subgoal and the instruction have a lower update frequency, i.e., every N time steps and every episode, we use an attention module to integrate the mixed input with different time resolution, which is inspired by [Chaplot *et al.*, 2018]. Besides, in order to ensure the state s and subgoal g are comparable, we transform both to the same scale (e.g., apply normalization).

4 Related Work

HRL. How to discover meaningful and effective hierarchies of policies is a long-standing research topic. The most popular formulation of HRL, Options [Sutton *et al.*, 1999], incorporate a terminate policy into each sub-policy. Previous options framework either relies on the artificial prior knowledge for designing options [Precup, 2000] or requires learning regularizers [Bacon *et al.*, 2017; Vezhnevets *et al.*, 2016]. Other methods [Dayan and Hinton, 1993; Vezhnevets *et al.*, 2017; Ghazanfari and Taylor, 2017] focus on how to decompose complicated tasks into subgoals. However, the prior work is mostly devoted to the single-task setting and assumes the optimal sequence of sub-tasks is fixed during evaluation [Oh *et al.*, 2017]. In this paper, the multi-task setting is considered as part of the algorithm to facilitates learning effective subgoals. Moreover, our evaluation is conditioned on previously unseen tasks, which makes it better to evaluate the transferability of the well-learned policies.

MTL. Multi-task learning is inspired by the fact that the knowledge contained in a task can be leveraged by other tasks [Zhang and Yang, 2017]. In RL, the MTL is often used as an auxiliary technology to help boost the agent's performance in terms of feature extraction [Jaderberg *et al.*, 2016], sample efficiency [Brunskill and Li, 2013] and knowledge transfer [Parisotto *et al.*, 2015]. Besides, the multi-task setting are sometimes used as experimental evaluation [Riemer *et al.*, 2018]. In this paper, we directly define the multi-task control optimization as the objective of HRL to learn effective and transferable subgoals. Our work is closely related to [Andreas *et al.*, 2017] and [Oh *et al.*, 2017]. However, they both require some prior knowledge during training, such as *policy sketch* (i.e., the number, name, and order of sub-tasks) [Andreas *et al.*, 2017] and *analogy-making* (i.e., the similarities between different sub-tasks) [Oh *et al.*, 2017]. In contrast, our work requires merely the task coding for the ultimate task, which leads to the generality of our method.

Environment name	Max step	Reward rate
GoToLocal	72	0.2717
OpenDoorsOrder	72	0.0425
PutNextLocal	144	0.0101
UnlockPickup	288	0.0029

Table 1: Properties of different environments. The reward rate column reports the probabilities that a random policy can receive any reward signal within the max step (calculated from 10,000 repeated trails).

5 Experiments

Our experiments consist of three parts[2]: (i) The comparative experiments with the previous HRL techniques verify our method's superiority in multi-task RL problems. (ii) The ablation analyses reveal the importance of various components. (iii) The visualization results show that the GMHRL does learn non-trivial, effective and interpretable subgoals. Our experiments are conducted in a set of challenging environments as follows. Visualization of these environments are shown in Figure 1, and their properties are reported in Table 1. All environments are partially observable, that is, the agent can

[1] The probability gradually decreases with training.

[2] Some supplementary details are available in https://www.dropbox.com/s/nugxeq4u1o0pnf7/appendix.pdf?dl=0.

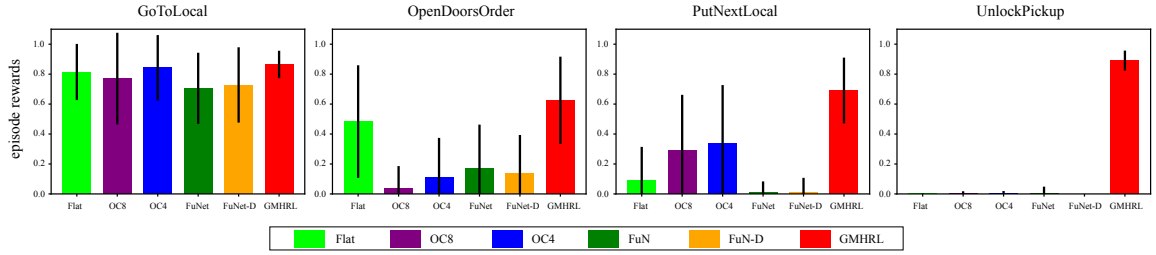

Figure 3: Performance after convergence compared with baselines. For each environment, we report the average episode rewards and standard deviation of each method in 100 random evaluate tasks.

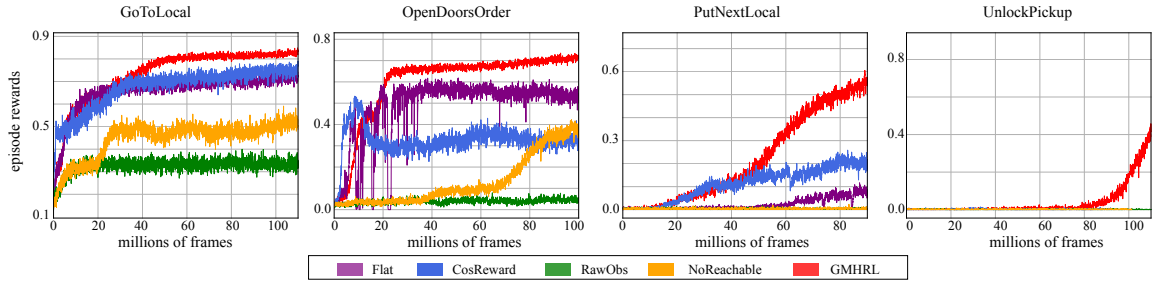

Figure 4: Learning curves of our GMHRL and its variants in the ablation experiment. Each curve has been smoothed.

only observe the grid cells in front of the agent (i.e., a 7×7 representation). There are seven available actions: turn left, turn right, forward, toggle, pickup, drop, done. Besides, the position of objects and the task switch after each episode.

GoToLocal. The agent is randomly placed in a single room without any door, where there are various objects of different colors. In each episode, the agent receives a "go to the [color] [object]" instruction, such as "go to the red ball". The instruction is completed only when the agent is located in the grid adjacent to the target object and faces the object. This setting only requires the ability for multi-task without considering complex sequences of object manipulation.

OpenDoorsOrder. There are four doors in a room. The agent can open the door and navigate the adjacent room. The instructions include three forms: "open the [X] door", "open the [X] door and then open the [Y] door" and "open the [X] door after you open the [Y] door". The agent should open the specified doors in a specific order.

PutNextLocal. The room setting is similar to GoToLocal, but the instruction is "put the [color A] [object A] next to the [color B] [object B]", such as "put the yellow key next to the red box". To complete the instruction, the agent must first go to the yellow key, take "pickup" action, then go near the red box, and finally place the yellow key on the grid adjacent to the red box.

UnlockPickup. The agent is placed in a room with a key and a locked door, and the door opens onto a room with a box. The instruction is "pick up the [color] box". To solve this task, the agent is required to execute the following sequential behaviors: go to the key with matching color, take "pickup" action, go to the door, take "toggle" action, go to the target box and finally take "pickup" action.

For each environment, we give a positive reward only when the agent fully completes the task. To encourage faster completion, the magnitude of the reward value is set to $1 - 0.9n/n_{\max}$ (similar to [Chevalier-Boisvert *et al.*, 2018]), where n represents the number of steps of a successful episode, and $n_{\max}$ is the maximum number of steps predefined for each environment.

5.1 Comparative Experiments

With the same evaluation flow, the following HRL methods are compared with our GMHRL framework. Considering that these baselines do not support multi-task learning in their original forms, we modify each for a fairer comparison.

Option-Critic. This method [Bacon *et al.*, 2017] is the first end-to-end approach for learning options without the need to provide additional rewards. Through fusing the instruction and observation in a way similar to our TDN (see Figure 2b), we extend the original option-critic architecture to the multi-task setting. As suggested in [Bacon *et al.*, 2017], the option number is an important super-parameter. Therefore, we test two baselines with 4 options and 8 options, denoted as **OC4** and **OC8**, respectively.

FeUdal Network (FuN). FeUdal Network [Vezhnevets *et al.*, 2017] is related to our GMHRL framework and originally motivated by feudal RL [Dayan and Hinton, 1993]. FuN also designs a hierarchy of two policies, where the high-level policy sets the subgoals for the low-level policy in a learned latent embedding space. We replicate the key design of the original FuN with two critical modifications. (i) We modify the Precept module in original FuN to support additional instruction input, which is similar to our option-critic modification. (ii) Considering the episode length in our evaluate environment is limited, we use the standard LSTM instead of the dilated LSTM proposed in [Vezhnevets *et al.*, 2017] which allows gradient flow through large hops in time.

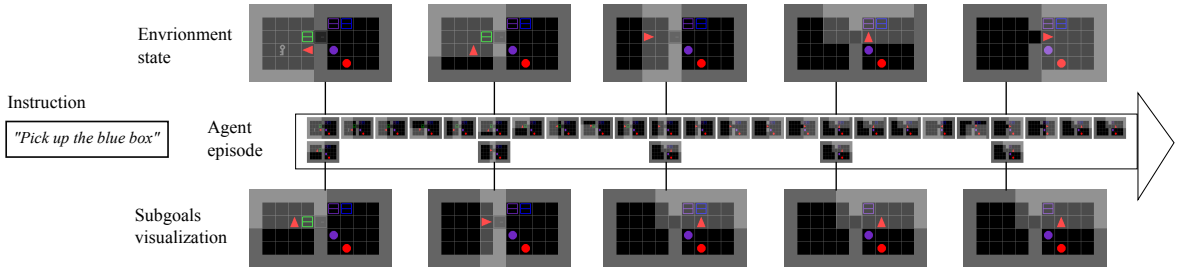

Figure 5: Visualization results on a task in UnlockPickup. The subgoals visualization identify the target state that the TDN expects the agent to reach after N steps.

FuN-D. The intrinsic reward function used in FuN is set as the direction cosine similarity. For a fairer comparison, we also explore the use of the same intrinsic reward function (see Equation 2) as in our framework.

Flat. This is a variant of our method without a hierarchy of policy. The PSN is used to directly handle the instruction, and its optimization is motivated by the external reward instead of our proposed intrinsic reward.

As shown in Figure 3, our GMHRL framework outperform these strong baselines in all environments. Although the performances of all algorithms are similar in the simple environment (i.e., GoToLocal), our approach has an absolute advantage in the more difficult environments, such as PutNextLocal and UnlockPickup. This result not only verifies the superiority of GMHRL in multi-task RL problems but also indicates that effective subgoals are more helpful in the complicated tasks which require sequential object manipulation.

5.2 Ablation Analyses

In our ablation experiment, our proposed method is compared to the following variants, as well as **Flat**.

CosReward. This variant uses the direction cosine similarity reward as the PSN's intrinsic reward function as suggested in [Vezhnevets *et al.*, 2017]. Concretely, the form of intrinsic reward is $\cos(s_{t+1} - s_t, g_t)$.

NoReachable. This variant uses the conventional training method without our proposed reachable subgoal training to update the parameters.

RawObs. This variant defines the subgoal and state in the raw observation space, as suggested in [Nachum *et al.*, 2018]. Intuitively, since the observations of the agent are pixel image, the distance between pixels cannot measure the relationship between states, which makes it difficult to obtain effective subgoals.

Figure 4 reports our ablation results. All variants make almost no progress in UnlockPickup. Concretely, although Flat has achieved comparable performance in GoToLocal and OpenDoorsOrder, it struggles in more complicated environments, which confirms the role of HRL. CosReward has a high learning efficiency in the initial stage of training but achieves limited performance, which might indicate that the direction similarity reward cannot generalize to our framework. By comparing NoReachable and GMHRL, the reachable subgoal training method is demonstrated to be able to speed up the learning significantly, thereby leading to better results. Besides, RawObs performs worst as it cannot scale to the problems with pixel image input.

5.3 Visualization Result

In addition to evaluating the overall performance in multiple tasks, we also design a visualization experiment for our well-learned GMHRL model. Considering that there are few feasible actions in our environment setting, we can obtain all possible observations and states of the agent after N steps by exhaustive search. When $t = 0, N, 2N, \cdots$, the TDN outputs the subgoal g_t according to current state s_t. We denote the the possible state and the possible observation after N steps as $s^N_{t,i}$ and $o^N_{t,i}$, respectively, where $i = 1, 2, \ldots, M$ and M is the number of possible states. By maximizing the intrinsic reward function as follow[3],

$$j = \arg\max_i r^L(s_t, a_t, s^N_{t,i}, g_t), \tag{5}$$

we can identify the target state $s^N_{t,j}$ as the subgoal that the TDN expects the PSN to achieve, and the corresponding observation $o^N_{t,j}$ gives the visualization. Figure 5 reports the visualization result on a task in UnlockPickup, which shows that our GMHRL does learn non-trivial, effective and interpretable subgoals.

6 Discussion

Establishing meaningful hierarchical policies may be an important stepping stone to human-like intelligence, and the key is how to obtain effective subgoals. This paper introduces a novel perspective for HRL, i.e., incorporating multi-task optimization into the HRL framework. Moreover, we propose a solution, GMHRL, which can automatically learn effective subgoals from multiple related tasks. However, the key of the advantage of MTL is the relevance between tasks, which is satisfied here by sharing the same state transition function. Future work might focus quantifying the task relevance and theoretically study its relationship with HRL.

7 Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant 61671266, 61327902, in part by Tsinghua University Initiative Scientific Research Program under Grant 20161080084, and in part by National

[3]In practice, we consider one-step margin, i.e., $s^{N-1}_{t,i}$ and $s^{N+1}_{t,i}$ are also considered.

High-tech Research and Development Plan under Grant 2015AA042306.

References

[Andreas *et al.*, 2017] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175, 2017.

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.

[Barto and Mahadevan, 2003] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 2003.

[Brunskill and Li, 2013] Emma Brunskill and Lihong Li. Sample complexity of multi-task reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 122–131, 2013.

[Chaplot *et al.*, 2018] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *AAAI Conference on Artificial Intelligence*, pages 2819–2826, 2018.

[Chevalier-Boisvert *et al.*, 2018] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: First steps towards grounded language learning with a human in the loop. *arXiv preprint arXiv:1810.08272*, 2018.

[Dayan and Hinton, 1993] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[Ghazanfari and Taylor, 2017] Behzad Ghazanfari and Matthew E Taylor. Autonomous extracting a hierarchical structure of tasks in reinforcement learning and multi-task reinforcement learning. *arXiv preprint arXiv:1709.04579*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, pages 1735–1780, 1997.

[Jaderberg *et al.*, 2016] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[McGovern and Barto, 2001] Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *International Conference on Machine Learning*, 2001.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Nachum *et al.*, 2018] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3307–3317. 2018.

[Oh *et al.*, 2017] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, 2017.

[Parisotto *et al.*, 2015] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[Parr and Russell, 1998] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, pages 1043–1049, 1998.

[Precup, 2000] Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[Riemer *et al.*, 2018] Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018.

[Schaul *et al.*, 2015] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[Stolle and Precup, 2002] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation and Approximation*, pages 212–223, 2002.

[Sutton *et al.*, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Vezhnevets *et al.*, 2016] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems*, pages 3486–3494, 2016.

[Vezhnevets *et al.*, 2017] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, 2017.

[Zhang and Yang, 2017] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.