Learning Multi-Objective Curricula for Robotic Policy Learning

Jikun Kang
McGill University
Mila - Québec AI Institute
jikun.kang@mail.mcgill.ca

Miao Liu
IBM Research
Miao.Liu1@ibm.com

Abhinav Gupta
Université de Montréal
Mila - Québec AI Institute
abhinavg@nyu.edu

Christopher Pal
Polytechnique Montréal
Mila - Québec AI Institute
christopher.pal@polymtl.ca

Xue Liu
McGill University
Mila - Québec AI Institute
xueliu@cs.mcgill.ca

Jie Fu*
Beijing Academy of AI
fujie@baai.ac.cn

Abstract: Various automatic curriculum learning (ACL) methods have been proposed to improve the sample efficiency and final performance of robots' policies learning. They are designed to control how a robotic agent collects data, which is inspired by how humans gradually adapt their learning processes to their capabilities. In this paper, we propose a unified automatic curriculum learning framework to create multi-objective but coherent curricula that are generated by a set of parametric curriculum modules. Each curriculum module is instantiated as a neural network and is responsible for generating a particular curriculum. In order to coordinate those potentially conflicting modules in a unified parameter space, we propose a multi-task hyper-net learning framework that uses a single hyper-net to parameterize all those curriculum modules. We evaluate our method on a series of robotic manipulation tasks and demonstrate its superiority over other state-of-the-art ACL methods in terms of sample efficiency and final performance. Our code is available at https://github.com/luciferkonn/MOC_CoRL22.

Keywords: ACL, Hyper-net, Multi-objective Curricula

1 Introduction

The concept that humans frequently organize their learning into a curriculum of interdependent processes according to their capabilities was first introduced to machine learning in [1]. Over time, curriculum learning has become more widely used in machine learning to control the stream of examples provided to training algorithms [2], to adapt model capacity [3], and to organize exploration [4]. Automatic curriculum learning (ACL) for deep reinforcement learning (DRL) [5] has recently emerged as a promising tool to learn how to adapt a robot's learning tasks to its capabilities during training. ACL can be applied to robotics' policies learning in various ways, including adapting initial states [6], shaping reward functions [7], generating goals [8]. More broadly, curriculum learning can also be used to modify the environment itself. For example, it can add new entities to the world or change the behaviors of other agents [9].

Oftentimes, only a single ACL paradigm (*e.g.*, generating subgoals) is considered. It remains an open question whether different paradigms are complementary to each other and if yes, how to combine them in a more effective manner similar to how the "rainbow" approach of [10] has greatly improved DRL performance in Atari games. Multi-task learning is notoriously difficult, and Yu et al. [11] hypothesize that the optimization difficulties might be due to the gradients from different tasks

*Corresponding Author

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

conflicting with each other thus hurting the learning process. In this work, we propose a multi-task bilevel learning framework for more effective multi-objective curricula robotics' policy learning. Concretely, inspired by neural modular systems [12] and multi-task RL [13], we utilize a set of neural modules and train each of them to output a sequence of tasks with different desired goals, rewards, initial states, etc. To coordinate potentially conflicting gradients from modules in a unified parameter space, we use a single hyper-net [14] to parameterize neural modules so that these modules generate a diverse and cooperative set of curricula. *Multi-task learning provides a natural curriculum for the hyper-net itself* since learning easier curriculum modules can be beneficial for learning more difficult curriculum modules with parameters generated by the hyper-net.

Furthermore, existing ACL methods usually rely on manually-designed paradigms of which the target and mechanism have to be clearly defined and it is therefore challenging to create a very diverse set of curriculum paradigms. Consider goal-based ACL for example, where the algorithm is tasked with learning how to rank goals to form the curriculum [15]. Many of these curriculum paradigms are based on simple intuitions that are inspired by learning in humans, but they usually take too simple forms (*e.g.*, generating subgoals) to apply to neural models. Instead, we propose to augment the hand-designed curricula introduced above with an *abstract* curriculum of which paradigm is learned from scratch. More concretely, we take the idea from memory-augmented meta-DRL [16] and equip the hyper-net with a non-parametric memory module, which is also directly connected to the DRL agent. The hyper-net can write entries to and update items in the memory, through which the DRL agent can interact with the environment under the guidance of the abstract curriculum maintained in the memory. The *write-only* permission given to the hyper-net over the memory is distinct from the common use of memory modules in meta-DRL literature, where the memories are both readable and writable. We point out that the hyper-net is instantiated as a recurrent neural network [17] which has its internal memory mechanism and thus a write-only extra memory module is enough. Another key perspective is that *such a write-only memory module suffices to capture the essence of many curriculum paradigms*. For instance, the subgoal-based curriculum can take the form of a sequence of coordinates in a game which can be easily generated as a hyper-net and stored in the memory module. The contributions of this work are as follows:

1. We introduce multi-objective curricula learning approach for improving the sample efficiency of solving challenging deep reinforcement learning tasks, which is an important problem that has not been adequately addressed before.
2. We further propose a unified automatic curriculum learning framework to create multi-objective but coherent curricula that are generated by a set of parametric curriculum modules. Each curriculum module is instantiated as a neural network and is responsible for generating a particular curriculum. To coordinate those potentially conflicting modules in unified parameter space, we propose a multi-task hyper-net learning framework that uses a single hyper-net to **parameterize** all those curriculum modules.

2 Related Work

Curriculum learning. Automatic curriculum learning (ACL) for deep reinforcement learning (DRL) [18, 5, 19, 20, 21] has recently emerged as a promising tool to learn how to adapt an agent's learning tasks based on its capacity during training. ACL [22] can be applied to DRL in a variety of ways, including adapting initial states [6, 23], shaping reward functions [24, 25], or generating goals [8, 15, 26, 27]. In a closely related work [28], a series of related environments of increasing difficulty have been created to form curricula. There are other works related to curriculum reinforcement learning (CRL).

Multi-task and neural modules. Learning with multiple objectives is shown to be beneficial in DRL tasks [29, 30, 31, 32]. Sharing parameters across tasks [33, 34, 35] usually results in conflicting gradients from different tasks. One way to mitigate this is to explicitly model the similarity between gradients obtained from different tasks [11, 36, 37, 38, 39, 40, 41]. On the other hand, researchers propose to utilize different modules for different tasks, thus reducing the interference of gradients

from different tasks [42, 43, 44, 45, 46, 47, 48]. Most of these methods rely on pre-defined modules that make them not flexible in practice. One exception is [12], which utilizes soft combinations of neural modules for multi-task robotics manipulation. However, there is still redundancy in the modules in [12], and those modules cannot be modified during inference. Instead, we use a hyper-net to dynamically update complementary modules on the fly conditional on the environments.

Memory-augmented meta DRL. Our approach is also related to episodic memory-based meta DRL [49, 16, 50, 51]. Different from memory-augmented meta DRL methods, the DRL agent in our case is not allowed to modify the memory. This design prevents the agent from writing its information into the memory module, which can interfere with the generated curriculum. Note that it is straightforward to augment the DRL agent with both readable and writable neural memory just like [16, 49], which is different from our read-only memory module designed for ACL.

Dynamic neural networks. Dynamic neural networks [52] can change their structures or parameters based on different environments. Dynamic filter networks [53] and hyper-nets [14] can both generate parameters.

Our proposed framework generalizes previous work by unifying the aforementioned key concepts with a focus on automatically learning multi-objective curricula from scratch for DRL-based robot learning.

3 Preliminaries

Reinforcement learning (RL) is used to train an agent policy with the goal of maximizing the (discounted) cumulative rewards through trial and error. A basic RL setting is modelled as a Markov decision process (MDP) with the following elements: $\mathcal{S}$ is the set of environment states. In this paper, goal $g \in \mathcal{G}$ corresponds to a set of states $S^g \subset S$. The goal is considered to be achieved when the agent is in any state $s_t \in S^g$; $\mathcal{A}$ is the set of actions; δ is the state transition probability function, where $\delta(s^{t+1}|s^t, a^t)$ maps a state–action pair at time-step t to a probability distribution over states at time $t+1$; R is the immediate reward after a transition from s to s' ; $\pi(\cdot;\phi_\pi)$ is the policy function parameterized by ϕ_π, and $\pi(a|s;\phi_\pi)$ denotes the probability of choosing action a given an observation s.

Automatic curriculum learning (ACL) is a learning paradigm where an agent is trained iteratively following a curriculum to ease learning and exploration in a multi-task problem. Since it is not feasible to manually design a curriculum for each task, recent work has proposed to create an implicit curriculum directly from the task objective. Concretely, it aims to maximize a metric P computed over a set of target tasks $T \sim \mathcal{T}_{target}$ after some episodes t'. Following the notation in [5], the objective is set to: $\max_{\mathcal{D}} \int_{T\sim\mathcal{T}_{target}} P_T^{t'} dT$, where $\mathcal{D} : \mathcal{H} \rightarrow \mathcal{T}_{target}$ is a task selection function. The input $\mathcal{H}$ can be consist of any information about past interactions. For example, in our experiments, the history consists of the last state of the episode. , and the output of $\mathcal{D}$ is a sequence of generated episodes, which contain different desired goals, initial states, rewards, etc.

Hyper-networks were proposed in [14] where one network (hyper-net) is used to generate the weights of another network. All the parameters of both networks are trained end-to-end using backpropagation. We follow the notation in [54] and suppose that we aim to model a target function $y : \mathcal{X} \times \mathcal{I} \rightarrow \mathbb{R}$, where $x \in \mathcal{X}$ is independent of the task and $I \in \mathcal{I}$ depends on the task. A **base** neural network $f_b(x; f_h(I;\theta_h))$ can be seen as a composite function, where $f_b : \mathcal{X} \rightarrow \mathbb{R}$ and $f_h : \mathcal{I} \rightarrow \Theta_b$. Conditioned on the task information I, the small **hyper**-net $f_h(I;\theta_h)$ generates the parameters θ_b of base-net f_b. Note that θ_b is never updated using loss gradients directly.

4 Learning Multi-Objective Curricula

We use a single hyper-net to dynamically parameterize all the curriculum modules over time and modify the memory module shared with the DRL agent. We call this framework a Multi-Objective

Curricula (MOC). This novel design encourages different curriculum modules to merge and exchange information through the shared hyper-net.

Following the design of hyper-networks with recurrence [14], this hyper-net is instantiated as a recurrent neural network (RNN), which we refer to as the **Hyper-RNN**, denoted as $f_h(I;\theta_h)$, in the rest of this paper to emphasize its dynamic nature. Additionally, the Hyper-RNN can be viewed as a configurator for other modules as suggested in [55]. Our motivation for the adoption of an RNN design is its capability for producing a distinct set of curricula for every episode, which strikes a better trade-off between the number of model parameters and its expressiveness. On the other hand, each manually designed curriculum module is also instantiated as an RNN, which is referred as a **Base-RNN** $f_b(x;\theta_b)$ parameterized by $\theta_b = f_h(I;\theta_h)$. Each Base-RNN is responsible for producing a specific curriculum, *e.g.*, a series of sub-goals.

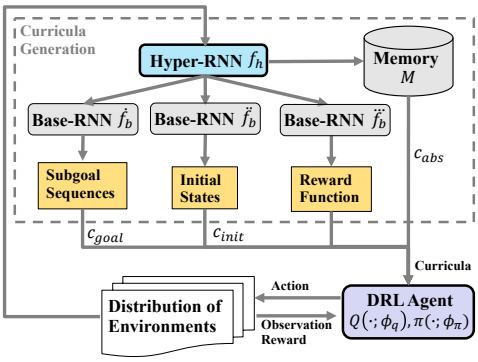


Figure 1: Illustration of MOC-DRL with two loops. Curricula generation corresponds to the outer-level loop. The DRL agent interacts with the environment in the inner-level loop.

Algorithm 1: Multi-Objective Curricula Deep Reinforcement Learning (MOC-DRL).

for *Episode* t' *in* 1 *to* T_{outer} **do**
- · Sample a new environment from the distribution of environments;
- · Hyper-RNN generates parameters for each curriculum module;
- **for** *Base-RNN in* 1 *to* 3 **do**
  - · Generate a curriculum;
- · Hyper-RNN updates the abstract curriculum in the memory;
- **for** *Training step* t *in* 1 *to* T_{inner} **do**
  - · DRL agent reads memory;
  - · Train DRL agent following curricula;
- · Update Hyper-RNN based on outer-level objective;

The architecture of MOC-DRL is depicted in Fig. 1, and its corresponding pseudo-code is given in Alg. 1. We formulate the training procedure as a bilevel optimization problem [56] where we minimize an outer-level objective that depends on the solution of the inner-level tasks.

In our case, the *outer*-level optimization comes from the curriculum generation loop where each step is an episode denoted as t'. On the other hand, the *inner*-level optimization involves a common DRL agent training loop on the interactions between the environment and the DRL agent, where each time-step at this level is denoted as t. We defer the discussion on the details to Sec. 4.3.

Inputs, I, of the Hyper-RNN, f_h, consist of: (1) the final state of the last episode, and (2) role identifier for each curriculum module (*e.g.*, for initial states generation) represented as a one-hot encoding. Ideally, we expect each Base-RNN to have its own particular role, which is specific to each curriculum.When generating the parameters for each Base-RNN, we additionally feed the role identifier representation to the Hyper-RNN.

Outputs of the Hyper-RNN at episode t' include: (1) parameters $\theta_b^{t'}$ for each Base-RNN, and (2) the abstract curriculum, $\mathbf{h}_h^{t'}$, maintained in the memory module. Here $\mathbf{h}_h^{t'}$ corresponds to the hidden states of the Hyper-RNN such that $[\theta_b^{t'}, \mathbf{h}_h^{t'}] = f_h(I^{t'};\theta_h)$.

4.1 Manually Designed Curricula

In Sec. 4.1, we describe the details of generating manually designed curricula while the process of updating the abstract curriculum is described in Sec. 4.2. We describe how to train them in Sec. 4.3.

In this work, we use three curriculum modules responsible for generating pre-defined curricula [5]: *initial state generator, sub-goal state generator,* and *reward shaping generator*. Our approach can be easily extended to include other forms of curricula (*e.g.*, selecting environments from a discrete set [57]) by adding another curriculum generator to the shared hyper-net. These Base-RNNs

simultaneously output the actual curricula for the DRL agent in a synergistic manner. It should be noted that these Base-RNNs are not directly updated by loss gradients, as their *pseudo-parameters* are generated by the Hyper-RNN.

Generating subgoal state g^t as curriculum $\mathbf{c}_{goal}$ with Base-RNN $\dot{f}_b$. As one popular choice in ACL for DRL, the subgoals can be selected from discrete sets [8] or a continuous goal space [15]. A suitable subgoal state g^t can ease the learning procedures by guiding the agent on how to achieve subgoals step by step and ultimately solving the final task.

To incorporate the subgoal state in the overall computation graph, in this paper, we adopt the idea from universal value functions [58] and modify the action-value function, $Q(\cdot;\phi_q)$, to combine the generated subgoal state with other information $Q := Q(s^t, a^t, g^t; \phi_q) = Q(s^t, a^t, \mathbf{c}_{goal}; \phi_q)$, where s^t is the state, a^t is the action, and g^t is the generated subgoal state. The loss is defined as $\mathcal{J}_{goal} = \mathbb{E}_{(s^t,a^t,r^t,s^{t+1},g^t)\sim H_{buf}}[(Q(s^t, a^t, \mathbf{c}_{goal}; \phi_q) - \dot{y})^2]$, where $\dot{y}$ is the one-step look-ahead:

$$\dot{y} = r^t + \lambda \mathbb{E}_{a^{t+1}\sim\pi_\theta(s^{t+1})}[Q(s^t, a^t, \mathbf{c}_{goal}; \phi_q) \quad - \log(\pi(a^{t+1}|s^{t+1}; \phi_\pi))], \tag{1}$$

H_{buf} is the replay buffer and λ is the discount factor.

Generating initial state $\mathbf{s}_0$ as curriculum $\mathbf{c}_{init}$ with Base-RNN $\ddot{f}_b$. Intuitively, if the starting state $\mathbf{s_0}$ for the agent is close to the end-goal state, the training would become easier, which forms a natural curriculum for training tasks whose difficulty depends on a proper distance between the initial state and the end-goal state. This method has been shown effective in control tasks with sparse rewards [6, 59]. To simplify implementation, even though we only need a single initial state $\mathbf{s}_0$ which is independent of time, we still use a Base-RNN,$\ddot{f}_b$, to output it.

The loss for this module is: $\mathcal{J}_{init} = \mathbb{E}_{(s^t,a^t)\sim H_{buf}}[(Q(s^t, a^t, c_{init}; \phi_q) - \dot{y})^2]$, where $\dot{y}$ is defined in Eqn. 1.

Generating potential-based shaping function as curriculum $\mathbf{c}_{rew}$ with Base-RNN $\dddot{f}_b$. Motivated by the success of using reward shaping for scaling RL methods to handle complex domains [60], we introduce reward shaping as the third manually selected curriculum. The reward shaping function can take the form of: $\dddot{f'}_b(s^t, a^t, s^{t+1}) = \mu \cdot \dddot{f}_b(s^{t+1}) - \dddot{f}_b(s^t)$, where μ is a hyper-parameter and $\dddot{f}_b()$ is base-RNN that maps the current state with a reward. In this paper, we add the shaping reward $f'_b(s^t, a^t, s^{t+1})$ to the original environment reward r. We further normalize the shaping reward between 0 and 1 to deal with wide ranges.

Following the optimal policy invariant theorem [60], we modify the look-ahead function: $\dddot{y} = r^t + \dddot{f}_b(s^t, a^t, s^{t+1} + \lambda \mathbb{E}_{a^{t+1}\sim\pi_\theta(s^{t+1})}[Q(s^t, a^t, \mathbf{c}_{rew}; \phi_q) - \log(\pi(a^{t+1}|s^{t+1}; \phi_\pi))]$. Thus the loss is defined as: $\mathcal{J}_{reward} = \mathbb{E}_{s^t,a^t,s^{t+1},a^{t+1}\sim H_{buf}}[(Q(s^t, a^t, \mathbf{c}_{rew}; \phi_q) - \dddot{y})^2]$.

4.2 Abstract Curriculum with Memory Mechanism

Although the aforementioned hand-designed curricula are generic enough to be applied in any environment/task, it is still limited by the number of such predefined curricula. It is reasonable to conjecture that there exist other curriculum paradigms, which might be difficult to hand-design based on human intuition. As a result, instead of solely asking the hyper-net to generate human-engineered curricula, we equip the hyper-nets with an external memory, in which the hyper-nets could read and update the memory's entries. The Hyper-RNN together with the memory can also be seen as an instantiation of shared workspaces [61] for different curricula modules, in which those modules exchange information.

By design, the content in the memory can serve as abstract curricula for the DRL agent, which is generated and adapted according to the task distribution and the agent's dynamic capacity during training. Even though there is no constraint on how exactly the hyper-net learns to use the memory, we observe that (see Sec. 5.2): 1) The hyper-net can receive reliable training signals from the manually designed curriculum learning objectives[2]; 2) Using the memory module alone would result in unstable

[2]To some extent, tasking the hyper-net to train manually designed curriculum modules can be seen as a curriculum itself for training the abstract curriculum memory module.

training; 3) Utilizing both the memory and manual curricula achieves the best performance and stable training. Thus, training this memory module with other manually designed curriculum modules contributes to the shaping of the content that can be stored in the memory and is beneficial for overall performance.

Specifically, external memory is updated by the Hyper-RNN. To capture the latent curriculum information, we design a neural memory mechanism similar to [62]. The form of memory is defined as a matrix M. At each episode t', the Hyper-RNN emits two vectors $\mathbf{m}_e^{t'}$, and $\mathbf{m}_a^{t'}$ as $[\mathbf{m}_e^{t'}, \mathbf{m}_a^{t'}]^T = [\sigma, \texttt{tanh}]^T(\mathbf{W}_h^{t'}\mathbf{h}_h^{t'})$: where $\mathbf{W}_h^{t'}$ is the weight matrix of Hyper-RNN to transform its internal state $\mathbf{h}_h^{t'}$ and ${}^{\mathrm{t}}[\cdot]$ denotes matrix transpose. Note that $\mathbf{W}_h$ are part of the Hyper-LSTM parameters θ_h.

The Hyper-RNN *writes* the abstract curriculum into the memory, and the DRL agent can *read* the abstract curriculum information freely.

Reading. The DRL agent can read the abstract curriculum $\mathbf{c}_{abs}$ from the memory $\mathbf{M}$. The read operation is defined as: $\mathbf{c}_{abs}^{t'} = \alpha^{t'}\mathbf{M}^{t'-1}$, where $\alpha^{t'} \in \mathbb{R}^K$ represents an attention distribution over the set of entries of memory $\mathbf{M}^{t'-1}$. Each scalar element $\alpha^{t',k}$ in an attention distribution $\alpha^{t'}$ can be calculated as: $\alpha^{t',k} = \texttt{softmax}(\texttt{cosine}(\mathbf{M}^{t'-1,k}, \mathbf{m}_a^{t'-1}))$, where we choose $\texttt{cosine}(\cdot,\cdot)$ as the align function, $\mathbf{M}^{t'-1,k}$ represents the k-th row memory vector, and $\mathbf{m}_a^{t'} \in \mathbb{R}^M$ is a add vector emitted by Hyper-RNN.

Updating. The Hyper-RNN can write and update abstract curriculum in the memory module. The write operation is performed as: $\mathbf{M}^{t'} = \mathbf{M}^{t'-1}(1-\alpha^{t'}\mathbf{m}_e^{t'}) + \alpha^{t'}\mathbf{m}_a^{t'}$, where $\mathbf{m}_e^{t'} \in \mathbb{R}^M$ corresponds to the extent to which the current contents in the memory should be deleted.

Equipped with the above memory mechanism, the DRL learning algorithm can read the memory and utilize the retrieved information for policy learning. We incorporate the abstract curriculum into the value function by $Q(s^t, a^t, g^t, c_{abs}^{t'}; \phi_q)$. Similar to manually designed curricula, we minimize the Bellman error and define the loss function for the abstract curriculum as: $\mathcal{J}_{abstract} = \mathbb{E}_{(s^t,a^t,r^t,s^{t+1},c_{abs}^{t'})\sim H_{buf}}[(Q(s^t, a^t, c_{abs}^{t'}; \phi_q) - \dot{y})^2]$, where $\dot{y}$ is defined in Eqn. 1.

4.3 Bilevel Training of Hyper-RNN

After introducing the manually designed curricula in Sec. 4.1 and the abstract curriculum in Sec. 4.2, here we describe how we update the Hyper-RNN's parameters θ_h, the parameters associated with the DRL agent ϕ_q and ϕ_π. Since the Hyper-RNN's objective is to serve the DRL agent, we naturally formulate this task as a bilevel problem [56] of optimizing the parameters associated with multi-objective curricula generation by nesting one inner-level loop in an outer-level training loop.

Outer-level training of Hyper-RNN. Specifically, the inner-level loop for the DRL agent learning and the outer-level loop for training Hyper-RNN with hyper-gradients. The outer-level loss is defined as :$\mathcal{J}_{outer} = \mathcal{J}_{initial} + \mathcal{J}_{goal} + \mathcal{J}_{reward} + \mathcal{J}_{abs}$.

Since the manually designed curricula and abstract curricula are all defined in terms of Q-function, for the implementation simplicity, we combine them together $\mathcal{J}_{outer} = \mathbb{E}_{s^t,a^t,s^{t+1},a^{t+1}\sim H_{buf}}[(Q(s^t, a^t, \mathbf{c}_{goal}, \mathbf{c}_{rew}, \mathbf{c}_{init}, \mathbf{c}_{abs}; \phi_q) - \ddot{y})^2]$. Following the formulation and implementation in [56], we obtain $\theta_h^* = \operatorname{argmin}(\theta_h; \mathcal{J}_{outer}(\operatorname{argmin}(\phi; \mathcal{J}_{inner}(\theta_h, \phi))))$.

Inner-level training of DRL agent. The parameters associated with the inner-level training, ϕ_q and ϕ_π, can be updated based on any RL algorithm. In this paper, we use Proximal Policy Optimization (PPO) [63] which is a popular policy gradient algorithm that learns a stochastic policy.

5 Experiments

We evaluate and analyze our proposed MOC DRL on the CausalWorld [64], as this environment enables us to easily design and test different types of curricula in a fine-grained manner. This environment also provides wrapper makes the environment to execute actions on the real robot, which

can be used in sim2real experiments. It should be noted that we do not utilize any causal elements of the environment. It is straightforward to apply our method to other DRL environments without major modification. Moreover, the training and evaluation task distributions are handled by CausalWorld. Take task "Pushing" as an example: for each outer loop, we use CausalWorld to generate a task with randomly sampled new goal shapes from a goal shape family

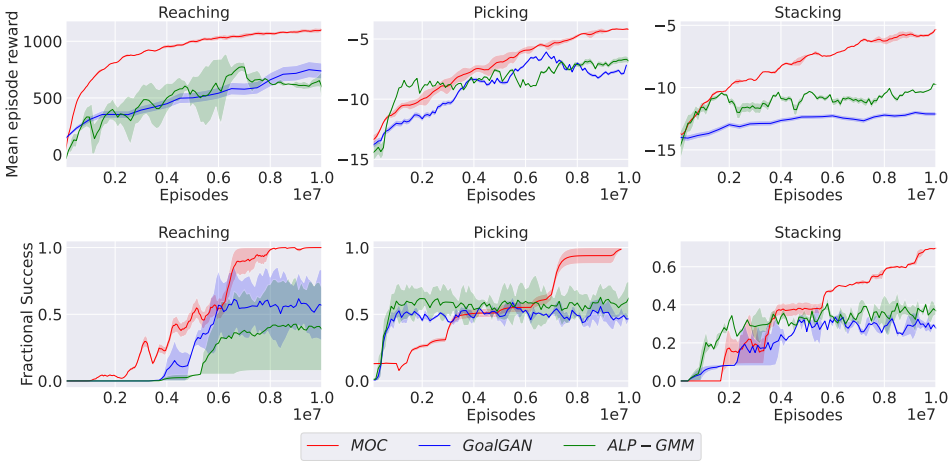


Figure 2: Comparisons with state-of-the-art ACL algorithms. Each learning curve is computed in three runs with different random seeds.

5.1 Comparing MOC with state-of-the-art ACL methods

We compare our proposed approach with the other state-of-the-art ACL methods: (1) GoalGAN [26], which uses a generative adversarial neural network (GAN) to propose tasks for the agent to finish; (2) ALP-GMM [5], which models the agent absolute learning progress with Gaussian mixture models. None of these baselines utilize multiple curricula.

Fig. 2 shows that MOC outperforms other ACL approaches in terms of mean episode reward, fractional success, and sample efficiency. Especially, MOC increases fractional success by up to 56.2% in all of three tasks, which illustrates the effectiveness of combining multiple curricula in a synergistic manner.

5.2 Ablation Study

Our proposed MOC framework consists of three key parts: the Hyper-RNN trained with hyper-gradients, multi-objective curriculum modules, and the abstract memory module. To get a better insight into MOC, we conduct an in-depth ablation study on probing these components. We first describe the MOC variants used in this section for comparison as follows: (1) **MOC**$_{Base^-}$: MOC has the Hyper-RNN and the memory module but does not have the Base-RNNs for manually designed curricula. (2) **MOC**$_{Memory^-}$: MOC has the Hyper-RNN to generate three curriculum modules but does not have the memory module. (3) **MOC**$_{Memory^-,Hyper^-}$: MOC has Base-RNNs but does not have memory and Hyper-RNN components. It independently generates manually designed curricula. (4) **MOC**$_{Memory^-,Goal^+}$: MOC with Hyper-RNN and one Base-RNN, but without the memory module. It only generates the subgoal curriculum as our pilot experiments show that it is consistently better than the other two manually designed curricula and is easier to analyze its behavior by visualizing the state visitation.

Ablations of Hyper-RNN. By comparing MOC$_{Memory^-}$ with MOC$_{Memory^-,Hyper^-}$ as shown in Fig. 3, we can observe that letting a Hyper-RNN generate the parameters of different curriculum modules indeed helps in improving the sample efficiency and final performance. The advantage is even more obvious in the harder tasks `pick and place` and `stacking`. The poor performance of

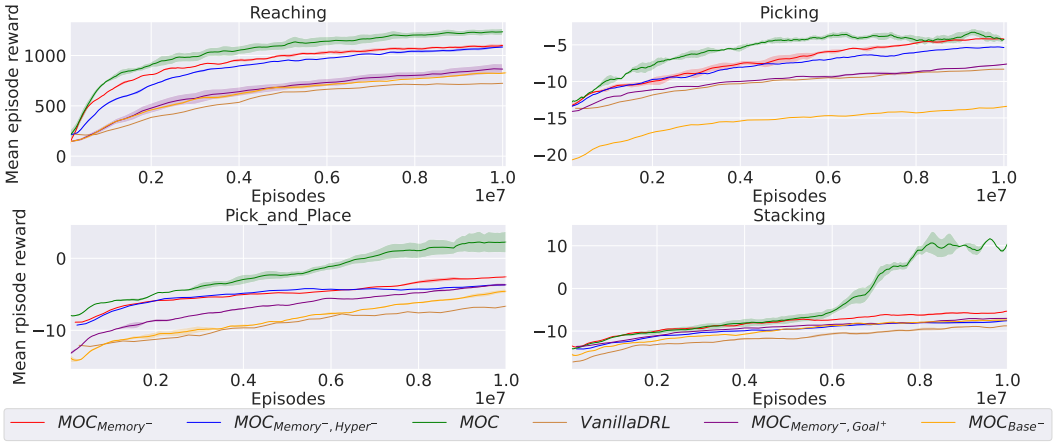


Figure 3: Comparison of algorithms with and without memory component on all four tasks. Each learning curve is obtained by three independent runs with different random seeds.

MOC$_{Memory^-,Hyper^-}$ may be caused by the potential conflicts among the designed curricula. For example, without coordination between the initial state curriculum and the goal curriculum, the initial state generator may set an initial state close to the goal state, which is easy to achieve by an agent but too trivial to provide useful training information to achieve the final goal. In sharp contrast, the Hyper-RNN can solve the potential conflicts from different curricula. All the curriculum modules are dynamically generated by the same hyper-net, and there exists an implicit information sharing between the initial state and the goal state curriculum generator. We put extra experimental results in Appendix Sec. A.3, Fig. 5.

Ablations of the memory module. We aim to provide an empirical justification for the use of the memory module and its associated abstract curriculum. By comparing MOC with MOC$_{Memory^-}$ as shown in Fig. 3, we can see that the memory module is crucial for MOC to improve sample efficiency and final performance. Noticeably, in `pick and place` and `stacking`, we see that MOC gains a significant improvement due to the incorporation of the abstract curriculum. We expect that the abstract curriculum could provide the agent with an extra implicit curriculum that is complementary to the manually designed curricula. We also find that it is better for the Hyper-RNN to learn the abstract curriculum while generating other manually designed curricula. Learning multiple manually designed curricula provides a natural curriculum for the Hyper-RNN itself since learning easier curriculum modules can be beneficial for learning of more difficult curriculum modules with parameters generated by the Hyper-RNN.

Ablations of individual curricula. We now investigate how gradually adding more curricula affects the training of DRL agents. By comparing MOC$_{Memory^-,Goal^+}$ and MOC$_{Memory^-}$ as shown in Fig. 3, we observe that training an agent with a single curriculum receives less environmental rewards as compared to the ones based on multiple curricula. This suggests that the set of generated curricula indeed helps the agent to reach intermediate states that are aligned with each other and also guides the agent to the final goal state.

6 Limitations

This paper presents a multi-objective curricula learning approach for solving challenging deep robotics tasks. However, it may be interesting to improve the sample efficiency of policy learning. Our method utilizes bi-level optimization, which elongates the training efficiency in terms of interactions between robots and environments. We argue that this limitation could be largely alleviated by replacing the model-free policy learning method with the model-based policy learning method.

References

[1] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *Ijcai*, pages 670–672, 1985.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 41–48. ACM, 2009.

[3] K. A. Krueger and P. Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

[4] J. Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991.

[5] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[6] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

[7] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, pages 1471–1479, 2016.

[8] N. Lair, C. Colas, R. Portelas, J.-M. Dussoux, P. F. Dominey, and P.-Y. Oudeyer. Language grounding through social interactions and curiosity-driven multi-goal learning. *arXiv preprint arXiv:1911.03219*, 2019.

[9] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone. Source task creation for curriculum learning. In *AAMAS*, pages 566–574. ACM, 2016.

[10] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pages 3215–3222. AAAI Press, 2018.

[11] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[12] R. Yang, H. Xu, Y. Wu, and X. Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

[13] T. Wang, H. Dong, V. Lesser, and C. Zhang. Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020.

[14] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *ICLR (Poster)*. OpenReview.net, 2017.

[15] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

[16] C. Blundell, B. Uria, A. Pritzel, Y. Li, A. Ruderman, J. Z. Leibo, J. Rae, D. Wierstra, and D. Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[18] S. Narvekar, J. Sinapov, and P. Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pages 2536–2542, 2017.

[19] M. Svetlik, M. Leonetti, J. Sinapov, R. Shah, N. Walker, and P. Stone. Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, pages 2590–2596. AAAI Press, 2017.

[20] S. Narvekar and P. Stone. Learning curriculum policies for reinforcement learning. *arXiv preprint arXiv:1812.00285*, 2018.

[21] A. Campero, R. Raileanu, H. Küttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

[22] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[23] T. Salimans and R. Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[24] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[25] P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788. PMLR, 2019.

[26] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528, 2018.

[27] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*, 2020.

[28] R. Portelas, C. Romac, K. Hofmann, and P.-Y. Oudeyer. Meta automatic curriculum learning. *arXiv preprint arXiv:2011.08463*, 2020.

[29] A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

[30] L. Pinto and A. Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2161–2168. IEEE, 2017.

[31] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[32] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[33] E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[34] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[35] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

[36] Y. Zhang and D.-Y. Yeung. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):1–31, 2014.

[37] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR, 2018.

[38] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[39] X. Lin, H. S. Baweja, G. Kantor, and D. Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[40] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

[41] Y. Du, W. M. Czarnecki, S. M. Jayakumar, M. Farajtabar, R. Pascanu, and B. Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

[42] S. P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, 1992.

[43] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

[44] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[45] A. H. Qureshi, J. J. Johnson, Y. Qin, T. Henderson, B. Boots, and M. C. Yip. Composing task-agnostic policies with deep reinforcement learning. *arXiv preprint arXiv:1905.10681*, 2019.

[46] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

[47] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6251. IEEE, 2018.

[48] H. Sahni, S. Kumar, F. Tejani, and C. Isbell. Learning to compose skills. *arXiv preprint arXiv:1711.11289*, 2017.

[49] M. Lengyel and P. Dayan. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20:889–896, 2007.

[50] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[51] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. In *International Conference on Machine Learning*, pages 2827–2836. PMLR, 2017.

[52] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021.

[53] X. Jia, B. De Brabandere, T. Tuytelaars, and L. Van Gool. Dynamic filter networks. In *NIPS*, 2016.

[54] T. Galanti and L. Wolf. On the modularity of hypernetworks. *arXiv preprint arXiv:2002.10006*, 2020.

[55] Y. LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *OpenReview.net*, 2022.

[56] E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[57] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[58] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1312–1320. JMLR.org, 2015.

[59] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *ICRA*, pages 15–21. IEEE, 2019.

[60] A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278–287. Morgan Kaufmann, 1999.

[61] A. Goyal, A. R. Didolkar, A. Lamb, K. Badola, N. R. Ke, N. Rahaman, J. Binas, C. Blundell, M. C. Mozer, and Y. Bengio. Coordination among neural modules through a shared global workspace. In *ICLR*. OpenReview.net, 2022.

[62] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

[63] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[64] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, M. Wuthrich, Y. Bengio, B. Schölkopf, and S. Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *ICLR*. OpenReview.net, 2021.

[65] O. Chang, L. Flokas, and H. Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2019.

A Appendix

A.1 Environment Settings

We choose five out of the nine tasks introduced in CausalWorld since the other four tasks have limited support for configuring the initial and goal states. Specifically, we enumerate these five tasks here: (1) `Reaching` requires moving a robotic arm to a goal position and reach a goal block; (2) `Pushing` requires pushing one block towards a goal position with a specific orientation (restricted to goals on the floor level); (3) `Picking` requires picking one block at a goal height above the center of the arena (restricted to goals above the floor level); (4) `Pick And Place` is an arena is divided by a fixed long block and the goal is to pick one block from one side of the arena to a goal position with a variable orientation on the other side of the fixed block; (5) `Stacking` requires stacking two blocks above each other in a specific goal position and orientation.

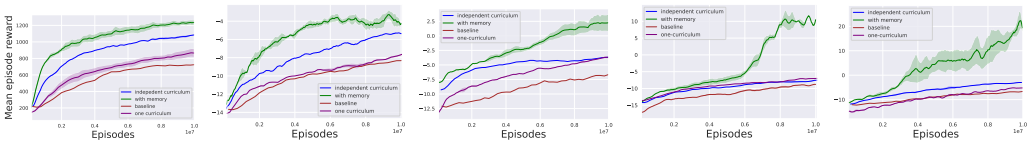
CausalWorld allows us to easily modify the initial states and goal states. In general, the initial state is the cylindrical position and Euler orientation of the block and goal state is the position variables of the goal block. These two control variables are both three-dimensional vectors with a fixed manipulation range. To match the range of each vector, we re-scale the generated initial states.

The reward function defined in CausalWorld is uniform across all possible goal shapes as the fractional volumetric overlap of the blocks with the goal shape, which ranges between 0 (no overlap) and 1 (complete overlap). We also re-scale the shaping reward to match this range.

We choose the PPO algorithm as our vanilla DRL policy learning method. We list the important hyper-parameters in Table. 1. We also provide the complete code in the supplementary material.

Table 1: Hyper-parameter values for PPO training

Parameter	Value
Discount factor (γ)	0.9995
n_steps	5000
Entropy coefficiency	0
Learning rate	0.00025
Maximum gradient norm	10
Value coefficiency	0.5
Experience buffer size	1e6
Minibatch size	128
clip parameter (ϵ)	0.3
Activation function	ReLU
Optimizer	Adam



(a) (b) (c) (d) (e)

Figure 4: Compare algorithms with different baselines in all of five tasks. Each learning curve is computed in three runs with different random seed.

A.2 Curricula Analysis and Visualization

In this section, we analyze the initial state curriculum and goal state curriculum. First, we replace the initial state curriculum with two different alternatives: (1) $MOC_{RandInitState}$, in which we replace the initial state curriculum in MOC with a uniformly chosen state. Other MOC components remains the same; (2) $MOC_{FixInitState}$, in which we replace the initial state curriculum in MOC with a fixed

	Mean Episode Reward	Success Ratio
$MOC_{RandInitState}$	936.9 (±35)	91% (±0.5%)
$MOC_{FixInitState}$	879.3 (±9)	89% (±1.1%)
$MOC_{RandGoalState}$	921.0 (±46)	91% (±0.5%)
MOC (Initial State)	1273 (±11)	100% (±0%)

(a) Analysis of initial state curriculum

	Mean Episode Reward	Success Ratio
GoalGAN	609 (±23)	56% (±18%)
ALP-GMM	568 (±26)	39% (±28%)
MOC (Goal State)	714 (±14)	68% (±15%)

(b) Analysis of subgoal curriculum

Table 2: Analysis of initial state curriculum and subgoal state curriculum.

initial state. The other MOC components remains the same. (3) $MOC_{RandGoalState}$, in which we replace the goal state curriculum in MOC with a uniformly chosen state. The other MOC components remains the same. The evaluations are conducted on the `reaching` task and the results are shown in Table 2a. From this table, we observe that MOC with initial state curriculum outperforms other two baseline schemes in terms of mean episode rewards and success ratio. This demonstrates the effectiveness of providing initial state curriculum. Besides, since "random sampling" outperforms "fixed initial state", we conjecture that it is better to provide different initial states, which might be beneficial for exploration.

In Sec. 5.1, we show that providing multi-objective curricula can improve the training of DRL agents. To further evaluate the advantages of hyper-RNN base-RNN framework, we conduct an experiment with GoalGAN, ALP-GMM and MOC with goal curriculum only. We evaluate on `reaching` task and the results are shown in Tab. 2b. In this table, we see that MOC Goal State (MOC$_{Memory^-,Goal^+}$), which is MOC has goal curriculum but doesn't have memory component, slightly outperform other two baseline schemes.

A.3 Additional Experimental Results

This section serves as a supplementary results for Sec. 5.

Fig. 5 shows the results of with and without Hyper-RNN in `pushing` tasks. The results validate the effectiveness of using Hyper-RNN. It is clear that, the incorporation of memory module consistently helps the DRL agent outperform other strong baselines in all scenarios. More importantly, in `pushing` task, we can observe a 5-fold improvement compared to the method with only the Hyper-RNN component.

Fig. 5 clearly validate the effectiveness of our proposed method in achieving both the best final performance and improving sample efficiency.

A.4 Additional Visualizations of States

Figs. 6, 7, 8, 9 visualize the state visitation density in task `reaching`, `picking`, `pushing` and `pick and place`, respectively.

From these results, we summarize the following observations: (1) The proposed architecture can help the agent explore different state spaces, which can be seen in the top row and bottom row. (2) The ablation study with three independent curricula often leads to exploring three different state space, as shown in Fig. 7 and Fig. 8. (3) By adding a memory component, the proposed MOC DRL can effectively utilize all curricula and help the agent focus on one specific state space. This is the reason why the proposed MOC DRL outperforms the other baselines in all tasks. (4) Comparing with Hyper-RNN ("no-mem") and without Hyper-RNN ("independent"), we can see that one of the benefits of using Hyper-RNN is aggregating different curricula. These can also be found in Fig. 7 and Fig. 8.

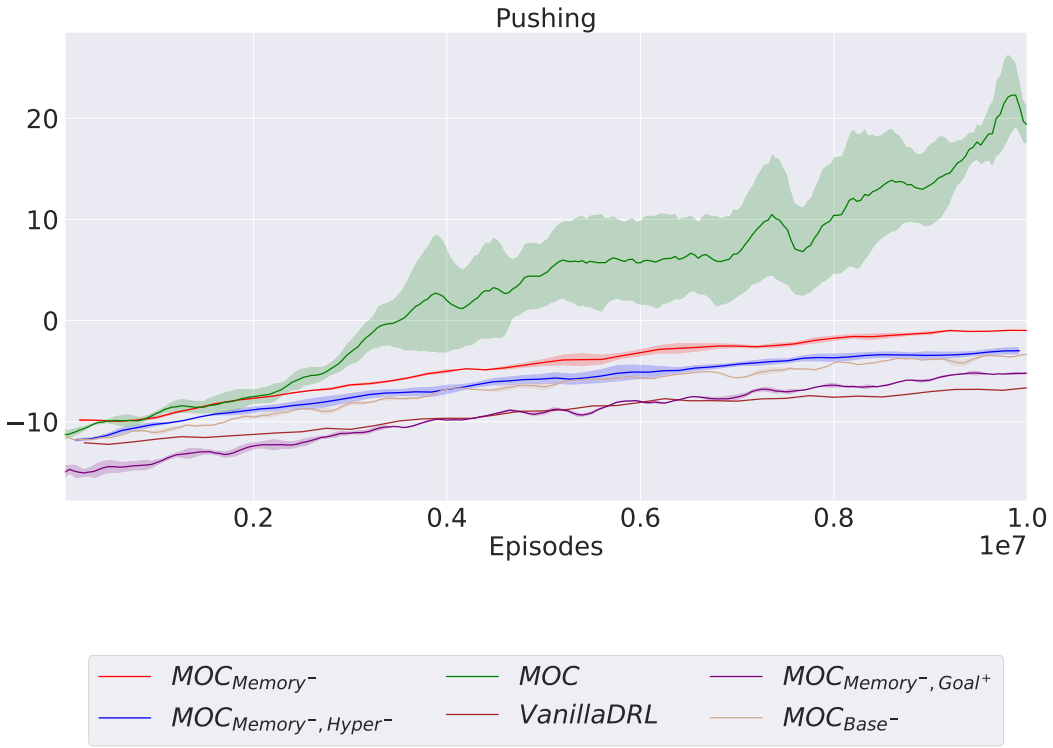
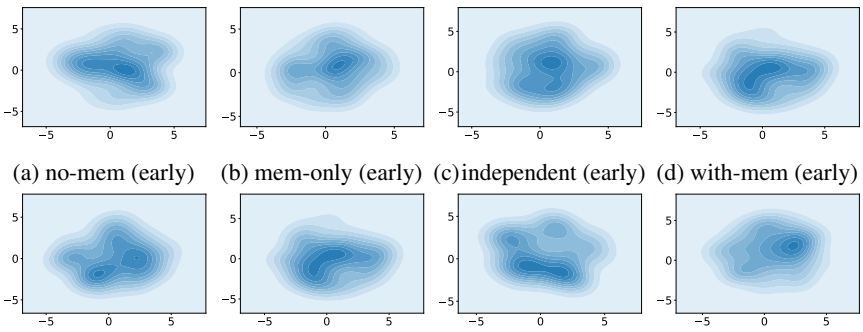


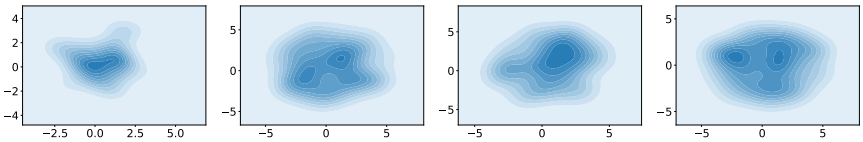
Figure 5: Comparison of algorithms with and without memory component in *pushing*. Each learning curve is computed in three runs with different random seeds.



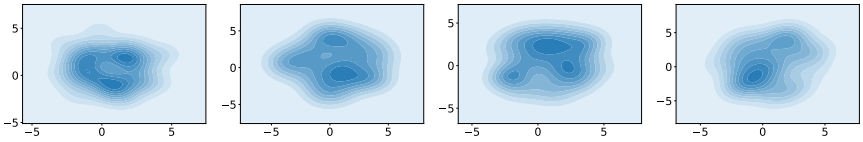
(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)

(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

Figure 6: Visualizations of state visitation density in early and late stages in *reaching*

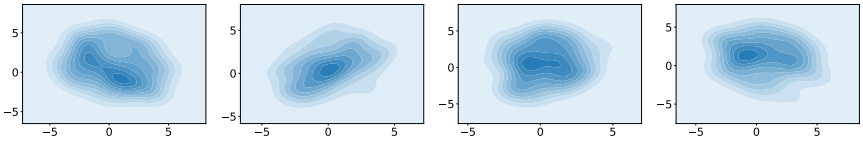


(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)

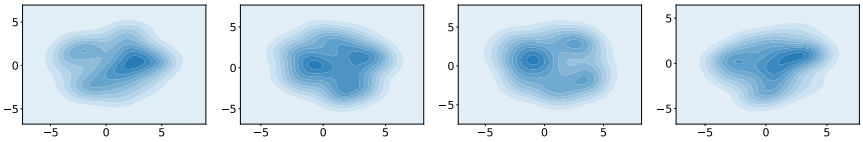


(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

Figure 7: Visualizations of state visitation density in early and late stages in *picking*



(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)



(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

Figure 8: Visualizations of state visitation density in early and late stages in *pushing*

A.5 Additional Experiment Results

In Sec. 5.1, we compared MOC with state-of-the art ACL algorithms. Here, we add two more baselines algorithms. The results are shown in Fig. 13:

- InitailGAN [6]: which generates adapting initial states for the agent to start with.
- PPO_{Reward^+}: which is a DRL agent trained with PPO algorithm and reward shaping. The shaping function is instantiated as a deep neural network.

A.6 PPO Modifications

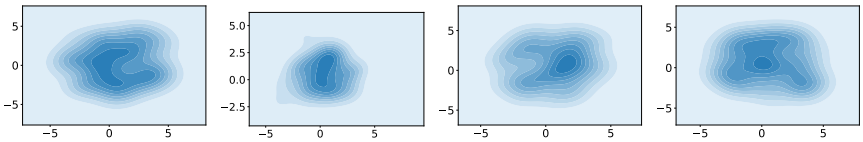
In Sec. 4, we propose a MOC-DRL framework for actor-critic algorithms. Since we adopt PPO in this paper, we now describe how we modify the PPO to cope with the learned curricula. We aim to maximize the PPO-clip objective:

$$\theta_{k+1} = argmax_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}} [min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \\ g(\epsilon, A^{\pi_{\theta_k}}(s,a))], \tag{2}$$

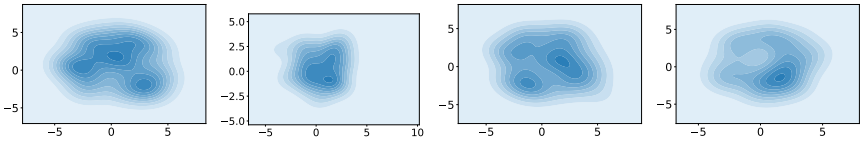
where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0, \end{cases}$$

where θ is the parameter of policy π, θ_k is the updated k step parameter by taking the objective above, A is the advantage function that we define as:



(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)



(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

Figure 9: Visualizations of state visitation density in early and late stages in *pick* and *place*

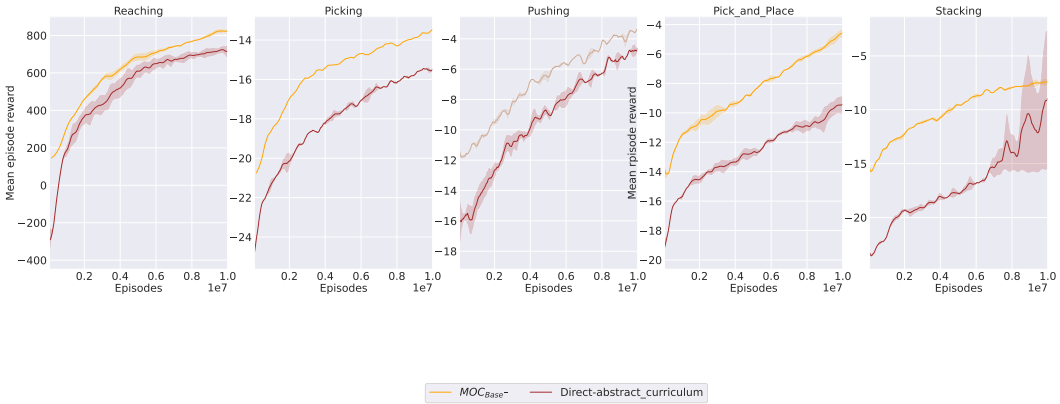


Figure 10: Comparison between read memory from memory and direct generate abstract curriculum

$$A(s,a) = Q(s,a) - V(s)$$

For the Hyper-RNN training, we modify the Q function as $Q(s, a, \mathbf{c}_{goal}, \mathbf{c}_{rew}, \mathbf{c}_{ini}, \mathbf{c}_{abs})$.

A.7 Bilevel Training

Here we provide more details regarding the bilevel training of Hyper-RNN introduced in Sec. 4.3. The optimal parameters θ_h^* are obtained by minimizing the loss function $\mathcal{J}_{outer}$. The key steps can be summarized as:

Step 1 Update PPO agent parameters θ on one sampled task by Eqn. 2

Step 2 With updated parameters θ, we train model parameters θ_h via SGD by minimizing the outer loss function $\theta_h^* = argmin_{\theta_h} \mathcal{J}_{outer}$.

Step 3 With θ_h, we generate manually designed curricula and abstract curriculum.

Step 4 We give the generate curriculum to the Q function and environment hyper-parameters.

Step 5 We go back to **Step 1** for agent training until converge.

A.8 Hyper-net

[14] introduce to generate parameters of Recurrent Networks using another neural networks. This approach is to put a small RNN cell (called the Hyper-RNN cell) inside a large RNN cell (the main RNN). The Hyper-RNN cell will have its own hidden units and its own input sequence. The input sequence for the Hyper-RNN cell will be constructed from 2 sources: the previous hidden

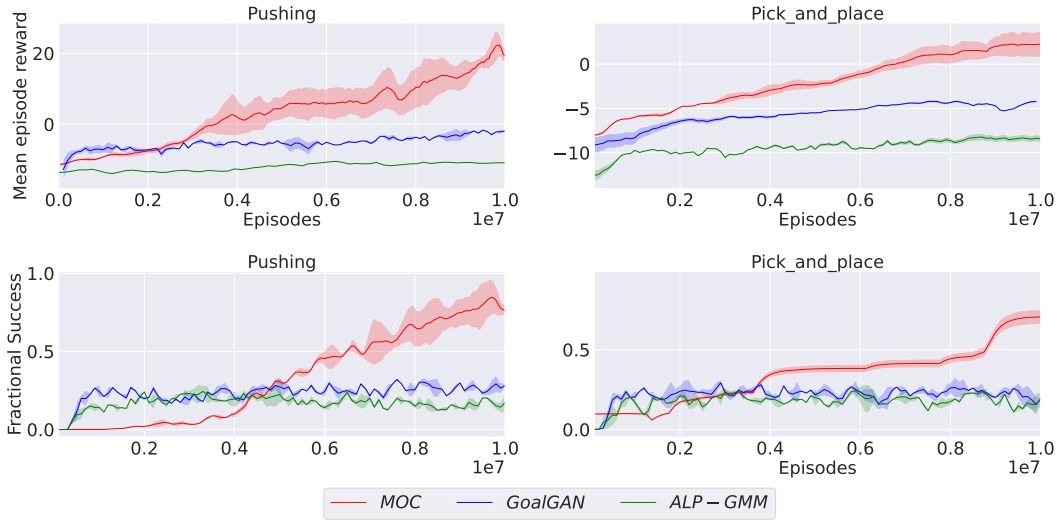


Figure 11: Comparison with ACL algorithms. Each learning curve is computed in three runs with different random seeds.

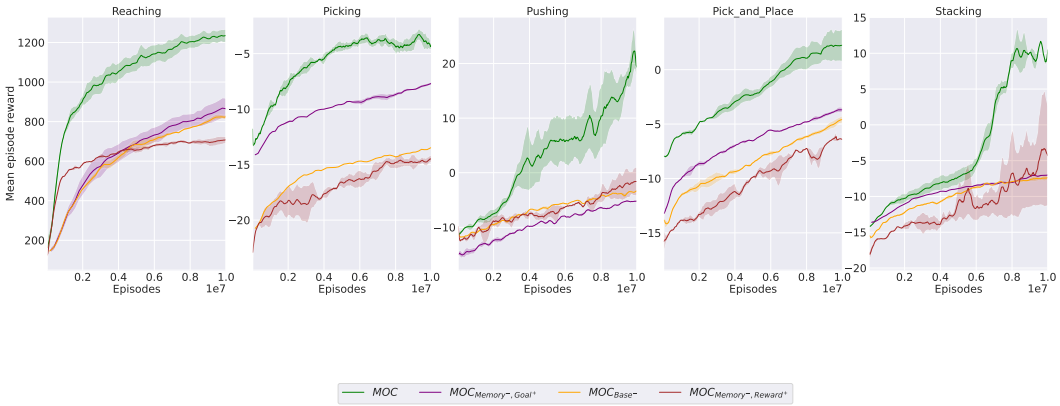


Figure 12: Comparison with reward curriculum only.

states of the main LSTM concatenated with the actual input sequence of the main LSTM. The outputs of the Hyper-RNN cell will be the embedding vector Z that will then be used to generate the weight matrix for the main LSTM. Unlike generating weights for convolutional neural networks, the weight-generating embedding vectors are not kept constant, but will be dynamically generated by the HyperLSTM cell. This allows the model to generate a new set of weights at each time step and for each input example. The standard formulation of a basic RNN is defined as:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b),$$

where h_t is the hidden state, ϕ is a non-linear operation such as $tanh$ or $relu$, and the weight matrics and bias $W_h \in \mathbb{R}^{N_h \times N_h}, W_x \in \mathbb{R}^{N_h \times N_x}, b \in \mathbb{R}^{N_h}$ is fixed each timestep for an input sequence $X = (x_1, \ldots, x_T)$. More conceretly, the parameters W_h, W_x, b of the main RNN are different at different time steps, so that h_t can now be computed as:

$$\begin{aligned} h_t &= \phi(W_h(z_h)h_{t-1} + W_x(z_x) + b(z_b)), where \\ W_h(z_h) &=< W_{hz}, z_h > \\ W_x(z_x) &=< W_{xx}, z_x > \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned} \tag{3}$$

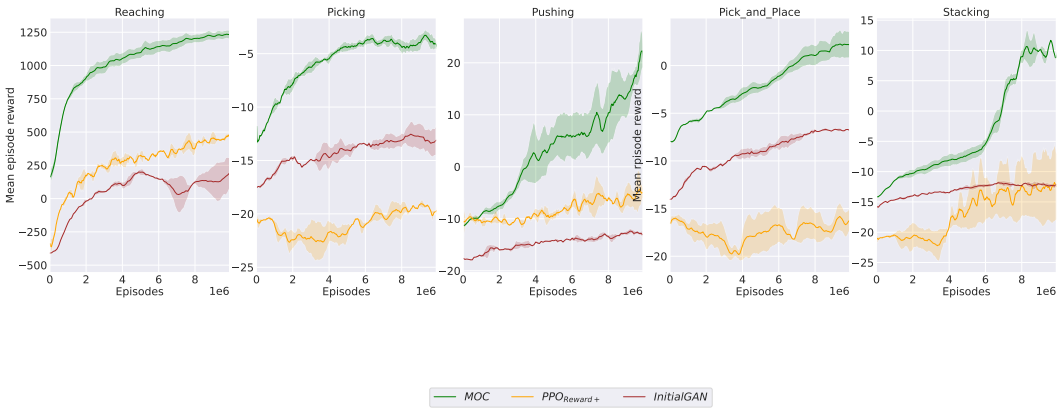


Figure 13: Comparison with Initial GAN and PPO with reward shaping only.

where $W_{hz} \in \mathbb{R}^{N_h \times N_h \times N_z}$, $W_{xz} \in \mathbb{R}^{N_h \times N_x \times N_z}$, $W_{bz} \in \mathbb{R}^{N_h \times N_z}$, $b_o \in \mathbb{R}^{N_h}$ and $z_h, z_x, z_z \in \mathbb{R}^{N_z}$. Moreover, z_h, z_x and z_b can be computed as a function of x_t and h_{t-1}:

$$
\begin{aligned}
\hat{x}_t &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
\hat{h}_t &= \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b}) \\
z_h &= W_{\hat{h}h} \hat{h}_{t-1} + \hat{b}_{\hat{h}h} \\
z_x &= W_{\hat{h}x} \hat{h}_{t-1} + \hat{b}_{\hat{h}x} \\
z_b &= W_{\hat{h}b} \hat{h}_{t-1}
\end{aligned}
\tag{4}
$$

Where $W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, b \in \mathbb{R}^{N_{\hat{h}}}$, and $W_{\hat{h}h}, W_{\hat{h}x}, W_{\hat{h}b} \in \mathbb{R}^{N_z \times N_{\hat{h}}}$ and $\hat{b}_{\hat{h}h}, \hat{b}_{\hat{h}x} \in \mathbb{R}^{N_z}$. The Hyper-RNN cell has $N_{\hat{h}}$ hidden units.

A.9 The abstract curriculum training

For some difficult tasks, we find that it is difficult to train a policy with small variances if the Hyper-RNN is initialized with random parameters[3].

As a simple workaround, we propose to pre-train the Hyper-RNN and memory components in a slightly unrelated task. In particular, when solving task T_x, we pre-train the abstract memory module on tasks other than T_x.

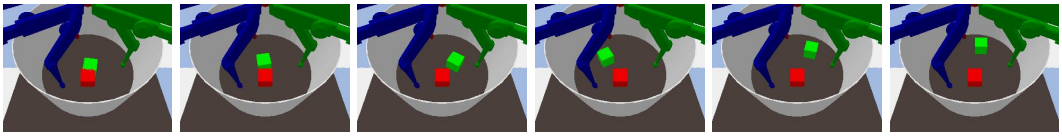
A.10 The visualization of generated sub-goal

The visualization of generated sub-goal state is shown in Fig. 14. Specifically, the arm is tasked to manipulate the red cube to the position shown as a green cube. As we can see, MOC generates subgoals that gradually change from "easy" (which are close to the initial state) to "hard" (which are close to the goal state). The generated subgoals have different configurations (*e.g.*, the green cube is headed north-west in 7000k steps but is headed north-east in 9000k steps), which requires the agent to learn to delicately manipulate robot arm.

A.11 Hyperparameters

In this section, we extensively evaluate the influence of different hyperparameters for the baselines and MOC, where the search is done with random search. We choose the reaching and stacking tasks, which are shown in Fig. 15, 16, 17. For example, in Fig. 15-(a), the first column represents the

[3]The weight initialization approach [65] designed for hyper-net does not help too much in our case.

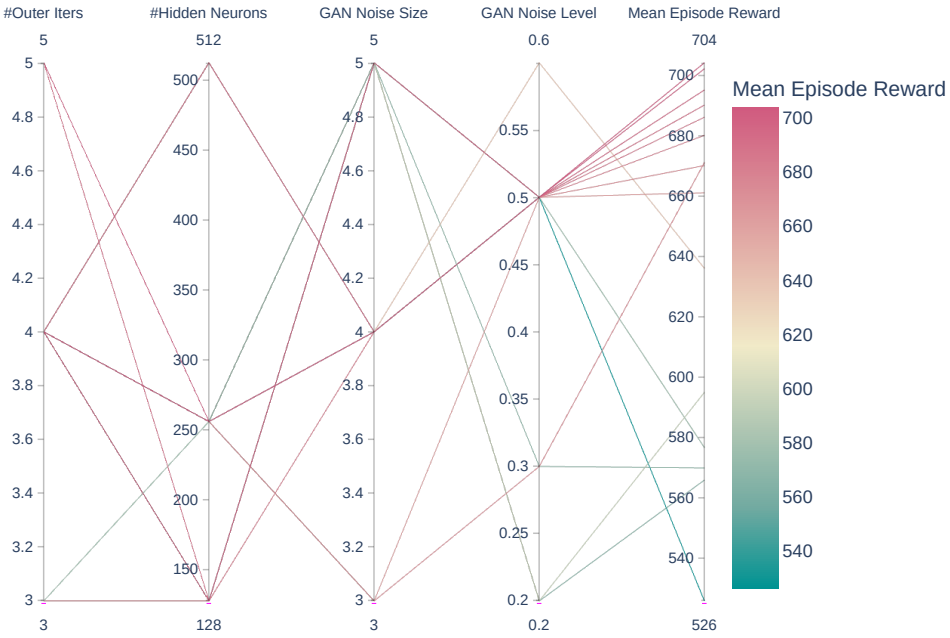


(a) 1000k steps (b) 3000k steps (c) 5000k steps (d) 7000k steps (e) 9000k steps (f) Goal state

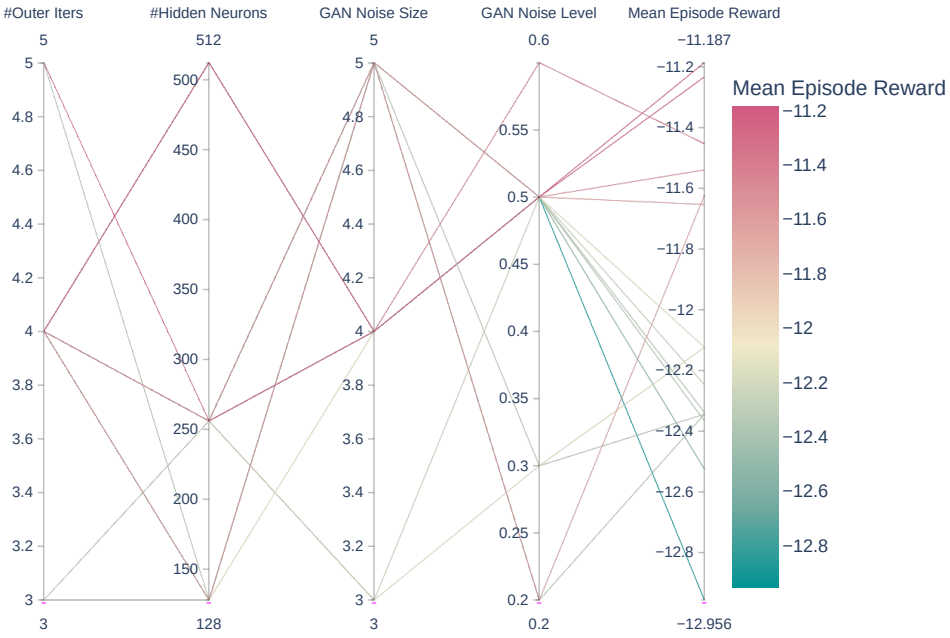
Figure 14: Visualization of generated subgoals

different values for outer iterations. A particular horizontal line, *e.g.*, $\{4, 512, 5, 0.5\}$, indicates a particular set of hyperparameters for one experiment. Besides, during the training phase, we adopt hyperparameters of PPO from stable-baselines3 and search two hyperparameters to test the MOC sensitivity.

We can observe that: (1) It is clear that MOC outperforms all the baselines with extensive hyperparameter search. (2) MOC is not sensitive to different hyperparameters.

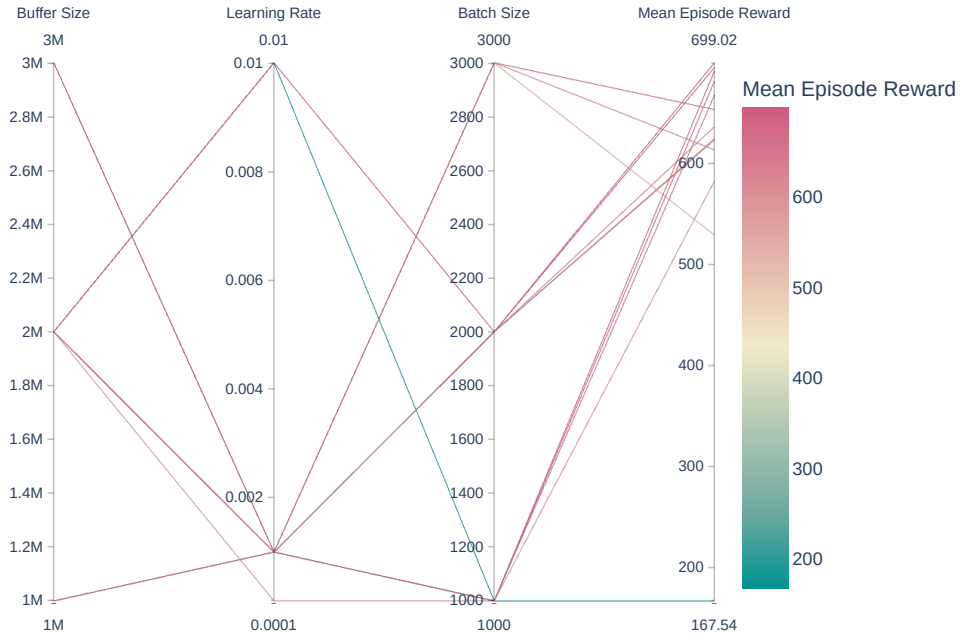


(a) Hyperparameter tuning in reaching task.

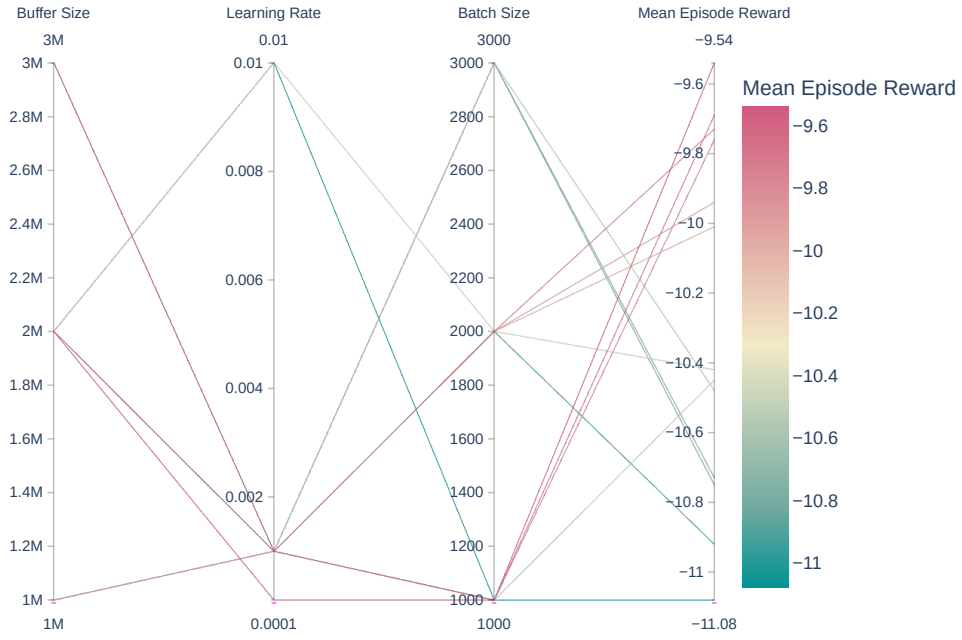


(b) Hyperparameter tuning in stacking task.

Figure 15: Hyperparameter tuning results for GoalGAN

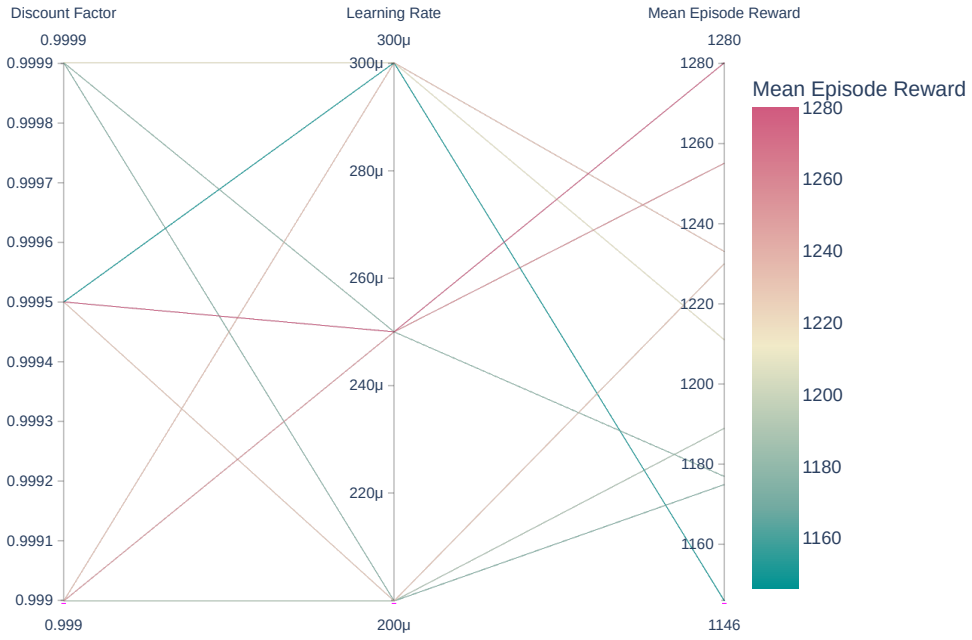


(a) Hyperparameter tuning in reaching task.

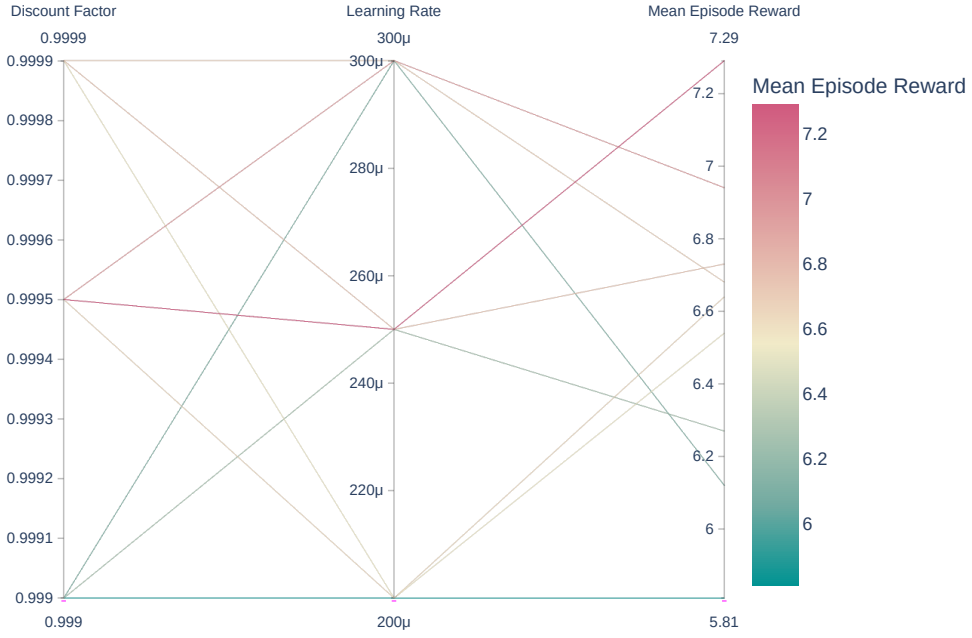


(b) Hyperparameter tuning in stacking task.

Figure 16: Hyperparameter tuning results for ALP-GMM



(a) Hyperparameter tuning in reaching task.



(b) Hyperparameter tuning in stacking task.

Figure 17: Hyperparameter tuning results for MOC