LEARNING MULTI-OBJECTIVE CURRICULA FOR DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Various automatic curriculum learning (ACL) methods have been proposed to improve the sample efficiency and final performance of deep reinforcement learning (DRL). They are designed to control how a DRL agent collects data, which is inspired by how humans gradually adapt their learning processes to their capabilities. For example, ACL can be used for subgoal generation, reward shaping, environment generation, or initial state generation. However, prior work only considers curriculum learning following one of the aforementioned predefined paradigms. It is unclear which of these paradigms are complementary, and how the combination of them can be learned from interactions with the environment. Therefore, 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 unified parameter space, we propose a multi-task hyper-net learning framework that uses a single hyper-net to parameterize all those curriculum modules. In addition to existing hand-designed curricula paradigms, we further design a flexible memory mechanism to learn an abstract curriculum, which may otherwise be difficult to design manually. 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.[1]

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 (Selfridge et al., 1985). Over time, curriculum learning has become more widely used in machine learning to control the stream of examples provided to training algorithms (Elman, 1993; Bengio et al., 2009), to adapt model capacity (Krueger & Dayan, 2009), and to organize exploration (Schmidhuber, 1991). Automatic curriculum learning (ACL) for deep reinforcement learning (DRL) (Portelas et al., 2020a) has recently emerged as a promising tool to learn how to adapt an agent's learning tasks to its capabilities during training. ACL can be applied to DRL in various ways, including adapting initial states (Florensa et al., 2017b; Ivanovic et al., 2019), shaping reward functions (Bellemare et al., 2016; Shyam et al., 2019), and generating goals (Lair et al., 2019; Sukhbaatar et al., 2017).

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 Hessel et al. (2018) has greatly improve DRL performance in Atari games. Multi-task learning is notoriously difficult (Caruana, 1997; Ruder, 2017) and Yu et al. (2020) hypothesize that the optimization difficulties might be due to the gradients from different tasks confliciting 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 DRL learning. Concretely, inspired by neural modular systems (Andreas et al., 2016; Yang et al., 2020) and multi-task RL (Wang et al., 2020; Yang et al., 2020), we utilize a set of neural modules and train each of them to output a curriculum. In order to coordinate potentially conflicting gradients from modules in a unified parameter space, we use a single hyper-net

[1]The code is available in the supplementary material.

(Schmidhuber, 1992; Ha et al., 2017) to parameterize neural modules so that these modules generate a diverse and cooperative set of curricula. *Multi-task learning provides a natural curriculum*[2] *for the hyper-net itself* since learning easier curriculum modules can be beneficial for learning of 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 (Sukhbaatar et al., 2017). Many of these curriculum paradigms are based on simple intuitions 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 (Blundell et al., 2016; Lengyel & Dayan, 2007) 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*[3] 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 (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) which has its own 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 essences 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 a hyper-net and stored in the memory module.

In summary, our proposed framework helps in learning coordinated curricula rather than naïvely combining multiple curricula that are otherwise trained independently. The combination of the curriculum modules and the memory module further boosts the performance in terms of sample-efficiency and unifies memory-augmented meta-DRL and ACL. We demonstrate our approach in a rich robotic manipulation environment, and show that it substantially outperforms state-of-the-art baselines and naïve ACL ensemble methods.

2 PRELIMINARIES

Reinforcement learning (RL) (Sutton & Barto, 2018) 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 modeled as a Markov decision process (MDP) with the following elements: $\mathcal{S}$ is the set of environment states; $\mathcal{A}$ 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 and every 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 (Portelas et al., 2020a), the objective is set to: $\max_{\mathcal{D}} \int_{T\sim\mathcal{T}_{target}} P_T^{t'} dT$, where $\mathcal{D} : \mathcal{H} \to \mathcal{T}_{target}$ is a task selection function. The input of $\mathcal{D}$ is $\mathcal{H}$ the history, and the output of $\mathcal{D}$ is a curriculum such as an initial state.

Hyper-networks were proposed in (Schmidhuber, 1992; Ha et al., 2017) 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 (Galanti & Wolf, 2020) and suppose that we aim to model a target function $y : \mathcal{X} \times \mathcal{I} \to \mathbb{R}$, 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

[2]This curriculum is not targeted at the DRL agent.

[3]To the DRL agent, the memory module is *read-only*.

composite function, where $f_b : \mathcal{X} \to \mathbb{R}$ and $f_h : \mathcal{I} \to \Theta_b$. 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.

3 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 (Ha et al., 2017), 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. 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.

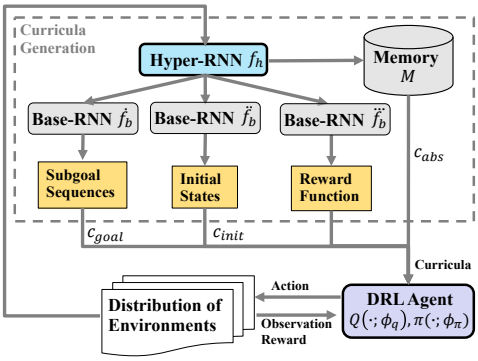


Figure 1: Illustration of MOC-DRL with two loops. Curricula generation corresponds to the outer-level loop. The DRL agent interacts with the environment in the inner-level loop.

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 (Grefenstette et al., 2019) 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. 3.3.

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 $[\theta_b^{t'}, \mathbf{h}_h^{t'}] = f_h(I^{t'}; \theta_h)$.

In Sec. 3.1, we describe the details of generating manually designed curricula while the process of updating the abstract curriculum is described in Sec. 3.2. We describe how to train them in Sec. 3.3.

3.1 MANUALLY DESIGNED CURRICULA

In this work, we use three curriculum modules responsible for generating pre-defined curricula (Portelas et al., 2020a): *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 (Matiisen et al., 2019)) 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 sequences $\gamma^{1:t}$ 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 (Lair et al., 2019) or a continuous goal space (Sukhbaatar et al., 2017). A suitable subgoal state sequence $\gamma_{1:t}$ can ease the learning procedures by guiding the agent 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 (Schaul et al., 2015) and modify the action-value function, $Q(\cdot;\phi_q)$, to combine the generated subgoal sequences with other information $Q := Q(s^t, a^t, \gamma^{1:t}; \phi_q) = Q(s^t, a^t, \mathbf{c}_{goal}; \phi_q)$, where s^t is the state, a^t is the action, and $\gamma^{1:t}$ is the generated subgoal state sequences. The loss is defined as $\mathcal{J}_{goal} = \mathbb{E}_{(s^t,a^t,r^t,s^{t+1},\gamma^t)\sim H_{buf}}[(Q(s^t, a^t, \mathbf{c}_{goal}; \phi_q) - \dot{y})^2]$, where $\dot{y}$ is the one-step look-ahead:

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

H_{buf} is the replay buffer and λ is the discount factor. Moreover, we use one subgoal state as the input by averaging the subgoal sequences $\gamma^{1:t}$.

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 initial state and the end-goal state. This method has been shown effective in control tasks with sparse rewards (Florensa et al., 2017b; Ivanovic et al., 2019). 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.

To evaluate the generated initial states, we incorporate it into the action-value (Q) function and estimate the expect return. 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 success of using reward shaping for scaling RL methods to handle complex domains (Ng et al., 1999), we introduce the reward shaping as the third manually selected curriculum. The reward shaping function can take the form of: $\ddot{f}'_b(s^t, a^t, s^{t+1}) = \mu \cdot \ddot{f}_b(s^{t+1}) - \ddot{f}_b(s^t)$, where μ is a hyper-parameter and $\ddot{f}_b()$ is base-RNN that maps the current state with a reward. In this paper, we add the shaping reward $\ddot{f}'_b(s^t, a^t, s^{t+1})$ 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 (Ng et al., 1999), we modify the look-ahead function: $\ddot{y} = r^t + \ddot{f}_b(s^t, a^t, s^{t+1} + \lambda \mathbb{E}_{a^{t+1}\sim\pi_\theta(s^{t+1})}[Q(s^t, a^t, \mathbf{c}_{rew}; \phi_q) - \log(\pi(a^{t+1}|s^{t+1}; \phi_\pi))]$. Thus the loss is defined as: $\mathcal{J}_{reward} = \mathbb{E}_{s^t,a^t,s^{t+1},a^{t+1}\sim H_{buf}}[(Q(s^t, a^t, \mathbf{c}_{rew}; \phi_q) - \ddot{y})^2]$.

3.2 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.

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. 4.3): 1) The hyper-net can receive reliable training signals from the manually designed curriculum learning objectives[4]; 2) Using the memory module alone would result in unstable training; 3) Utilizing both the memory and manually 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 the overall performance.

Specifically, an external memory is updated by the Hyper-RNN. To capture the latent curriculum information, we design a neural memory mechanism similar to (Graves et al., 2014; Weston et al., 2014; Sukhbaatar et al., 2015). 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:

$$^{\mathrm{t}}\left[\mathbf{m}_e^{t'}, \mathbf{m}_a^{t'}\right] = {}^{\mathrm{t}}[\sigma, \texttt{tanh}](\mathbf{W}_h^{t'}\mathbf{h}_h^{t'}) \tag{2}$$

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 the policy learning. We incorporate the abstract curriculum into the value function by $Q(s^t, a^t, \gamma^t, c_{abs}^{t'}; \phi_q)$. Similar to manually designed currricula, 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.

3.3 Bilevel Training of Hyper-RNN with Hyper-gradients

After introducing the manually designed curricula in Sec. 3.1 and the abstract curriculum in Sec. 3.2, 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 (Grefenstette et al., 2019) 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 (Grefenstette et al., 2019), we obtain $\theta_h^* = \mathrm{argmin}\,(\theta_h; \mathcal{J}_{outer}\,(\mathrm{argmin}\,(\phi; \mathcal{J}_{inner}(\theta_h, \phi))))$.

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) (Schulman et al., 2017) which is a popular policy gradient algorithm that learns a stochastic policy.

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

4 EXPERIMENTS

We evaluate and analyze our proposed MOC DRL on the CausalWorld (Ahmed et al., 2020), as this environment enables us to easily design and test different types of curricula in a fine-grained manner. 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. We start by demonstrating how multiple curricula can benefit from Hyper-RNN. Then we evaluate the effectiveness of the memory component as well as the abstract curriculum, and conduct an ablation study over different components of MOC DRL. Finally, we plot the state visitation density graph to analyze how each component can affect the agent's behavior. The results shown in this section are obtained during the training phase. Specifically, we evaluate the trained policy performance every 10000 steps with fixed curricula. This is in line with the evaluation procedures used in [5]. 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.

4.1 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.

The total number of training steps is 10 million steps. Similar to (Clavera et al., 2018; Nagabandi et al., 2018), we unroll the inner loop for one step to compute the approximate hyper-gradients to update the Hyper-RNN.

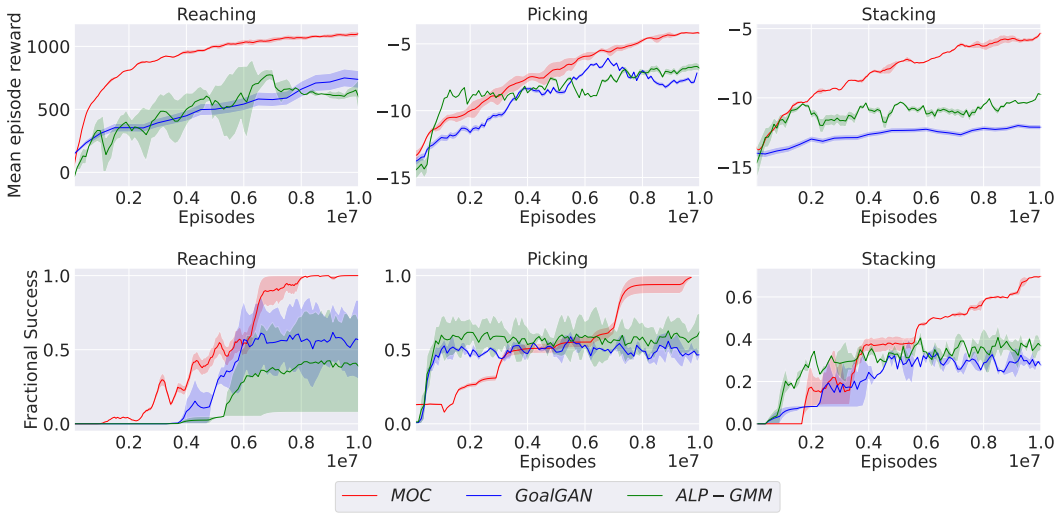


Figure 2: Comparisons with state-of-the-art ACL algorithms. Each learning curve is computed in three runs with different random seeds.

4.2 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 (Florensa et al., 2018), which uses a generative adversarial neural network (GAN) to propose tasks for

[5] https://stable-baselines3.readthedocs.io/en/master/

the agent to finish; (2) ALP-GMM (Portelas et al., 2020a), 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, ractional 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.

4.3 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) $\textbf{MOC}_{Base^-}$: MOC has the Hyper-RNN and the memory module but does not have the Base-RNNs for manually designed curricula. (2) $\textbf{MOC}_{Memory^-}$: MOC has the Hyper-RNN to generate three curriculum modules but does not have the memory module. (3) $\textbf{MOC}_{Memory^-,Hyper^-}$: MOC has Base-RNNs but does not have memory and Hyper-RNN components. It independently generates manually designed curricula. (4) $\textbf{MOC}_{Memory^-,Goal^+}$: 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. Here we investigate the importance of the Hyper-RNN for the MOC framework. We keep those three manually designed curriculum modules but remove the Hyper-RNN. Without the Hyper-RNN, these Base-RNNs' parameters are directly updated by loss gradients and learn to generate curricula independently. Note that neither of them have the abstract memory module. By comparing MOC_{Memory^-} with $\text{MOC}_{Memory^-,Hyper^-}$ 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 $\text{MOC}_{Memory^-,Hyper^-}$ 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.2, Fig. 4.

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 agent. By comparing $\text{MOC}_{Memory^-,Goal^+}$ 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.

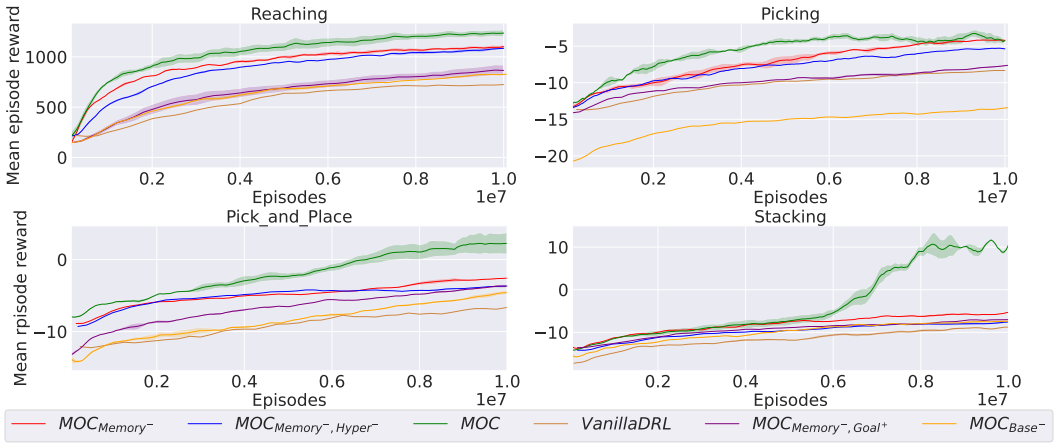


Figure 3: Comparison of algorithms with and without memory component on all four tasks. Each learning curve is obtained by three independent runs with different random seeds.

In summary, we have the following key observations: (1) The proposed three key components, *i.ei.*, Hyper-RNN, multi-objectrive curriculum modules, and memory module, can improve the policy performance and sample efficiency. (2) Without the centralized control from the Hyper-RNN, multiple independent curricula sometimes are not as useful as one curriculum alone. In `stacking`, we can see that one-curriculum can slightly outperform multiple curricula, which may be caused by the conflicts among independent curricula. (3) Curriculum learning is a powerful idea to help agent achieve a better performance. Other experiments with similar results are shown in the supplementary.

4.4 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 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 1a. 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.

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

(a) Analysis of initial state curriculum

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

(b) Analysis of subgoal curriculum

Table 1: Analysis of initial state curriculum and subgoal state curriculum.

In Sec. 4.2, 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. 1b. In this table, we see that MOC Goal State

($\text{MOC}_{Memory^-,Goal^+}$), which is MOC has goal curriculum but doesn't have memory component, slightly outperform other two baseline schemes.

5 Related Work

Curriculum learning. Automatic curriculum learning (ACL) for deep reinforcement learning (DRL) (Narvekar et al., 2017; Portelas et al., 2020a; Svetlik et al., 2017; Narvekar & Stone, 2018; Campero et al., 2020; Narvekar et al., 2020; Fang et al., 2021b; Czarnecki et al., 2018; Peng et al., 2018) has recently emerged as a promising tool to learn how to adapt an agent's learning tasks based on its capacity during training. ACL (Graves et al., 2017) can be applied to DRL in a variety of ways, including adapting initial states (Florensa et al., 2017b; Salimans & Chen, 2018; Ivanovic et al., 2019), shaping reward functions (Bellemare et al., 2016; Pathak et al., 2017; Shyam et al., 2019), or generating goals (Lair et al., 2019; Sukhbaatar et al., 2017; Florensa et al., 2018; Long et al., 2020). In a closely related work (Portelas et al., 2020b; Fang et al., 2021a; Narvekar et al., 2016), a series of related environments of increasing difficulty have been created to form curricula. Unfortunately, the curricula strongly rely on the capability to modify the environments fundamentally, which poses practical difficulties for creating the tasks. In contrast, our approach only assumes a mild authority to modify the environments.

Multi-task and neural modules. Learning with multiple objectives are shown to be beneficial in DRL tasks (Wilson et al., 2007; Pinto & Gupta, 2017; Riedmiller et al., 2018; Hausman et al., 2018). Sharing parameters across tasks (Parisotto et al., 2015; Rusu et al., 2015; Teh et al., 2017) 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 (Yu et al., 2020; Zhang & Yeung, 2014; Chen et al., 2018; Kendall et al., 2018; Lin et al., 2019; Sener & Koltun, 2018; Du et al., 2018). On the other hand, researchers propose to utilize different modules for different tasks, thus reducing the interference of gradients from different tasks (Singh, 1992; Andreas et al., 2017; Rusu et al., 2016; Qureshi et al., 2019; Peng et al., 2019; Haarnoja et al., 2018; Sahni et al., 2017). Most of these methods rely on pre-defined modules that make them not attractive in practice. One exception is (Yang et al., 2020), which utilizes soft combinations of neural modules for multi-task robotics manipulation. However, there is still redundancy in the modules in (Yang et al., 2020), and those modules cannot be modified during inference. Instead, we use a hyper-net to dynamically update complementary modules on the fly conditioned on the environments.

Memory-augmented meta DRL. Our approach is also related to episodic memory-based meta DRL (Lengyel & Dayan, 2007; Blundell et al., 2016; Vinyals et al., 2016; Pritzel et al., 2017). Different from memory augmented meta DRL methods, the DRL agent in our case is not allowed to modify the memory. Note that it is straightforward to augment the DRL agent with a both readable and writable neural memory just like (Blundell et al., 2016; Lengyel & Dayan, 2007), which is different from our read-only memory module designed for ACL.

Dynamic neural networks. Dynamic neural networks (Han et al., 2021) can change their structures or parameters based on different environments. Dynamic filter networks (Jia et al., 2016) and hyper-nets (Ha et al., 2017) can both generate parameters.

Our proposed framework borrows, extends, and unifies the aforementioned key concepts with a focus on automatically learning multi-objective curricula from scratch for DRL.

6 Conclusion

This paper presents a multi-objective curricula learning approach for solving challenging deep reinforcement learning tasks. Our method trains a hyper-network for parameterizing multiple curriculum modules, which control the generation of initial states, subgoals, and shaping rewards. We further design a flexible memory mechanism to learn abstract curricula. Extensive experimental results demonstrate that our proposed approach significantly outperforms other state-of-the-art ACL methods in terms of sample efficiency and final performance.

REFERENCES

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wüthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175. PMLR, 2017.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2019.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803. PMLR, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pp. 617–629. PMLR, 2018.

Wojciech Marian Czarnecki, Siddhant M. Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Nicolas Heess, Simon Osindero, and Razvan Pascanu. Mix & match agent curricula for reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1095–1103. PMLR, 2018.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Adaptive procedural task generation for hard-exploration problems. In *ICLR*. OpenReview.net, 2021a.

Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Discovering generalizable skills via automated generation of diverse tasks. In *Robotics: Science and Systems*, 2021b.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pp. 482–495. PMLR, 2017a.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017b.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pp. 1515–1528, 2018.

Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *arXiv preprint arXiv:2002.10006*, 2020.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. PMLR, 2017.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6244–6251. IEEE, 2018.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pp. 3215–3222. AAAI Press, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 15–21. IEEE, 2019.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NIPS*, 2016.

Alex Kendall, Yarin Gal, and Roberto 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*, pp. 7482–7491, 2018.

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

Nicolas Lair, Cédric Colas, Rémy Portelas, Jean-Michel Dussoux, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language grounding through social interactions and curiosity-driven multi-goal learning. *arXiv preprint arXiv:1911.03219*, 2019.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20:889–896, 2007.

Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Qian Long, Zihan Zhou, Abhibav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*, 2020.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Sanmit Narvekar and Peter Stone. Learning curriculum policies for reinforcement learning. *arXiv preprint arXiv:1812.00285*, 2018.

Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *AAMAS*, pp. 566–574. ACM, 2016.

Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pp. 2536–2542, 2017.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.*, 21:181:1–181:50, 2020.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Ivan Bratko and Saso Dzeroski (eds.), *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pp. 278–287. Morgan Kaufmann, 1999.

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

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017.

Bei Peng, James MacGlashan, Robert Tyler Loftin, Michael L. Littman, David L. Roberts, and Matthew E. Taylor. Curriculum design for machine learners in sequential decision tasks. *IEEE Trans. Emerg. Top. Comput. Intell.*, 2(4):268–277, 2018.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020a.

Rémy Portelas, Clément Romac, Katja Hofmann, and Pierre-Yves Oudeyer. Meta automatic curriculum learning. *arXiv preprint arXiv:2011.08463*, 2020b.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. *arXiv preprint arXiv:1905.10681*, 2019.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pp. 4344–4353. PMLR, 2018.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Himanshu Sahni, Saurabh Kumar, Farhan Tejani, and Charles Isbell. Learning to compose skills. *arXiv preprint arXiv:1711.11289*, 2017.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schaul15.html.

Jürgen Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Oliver G Selfridge, Richard S Sutton, and Andrew G Barto. Training and tracking in robotics. In *Ijcai*, pp. 670–672, 1985.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pp. 5779–5788. PMLR, 2019.

Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, 1992.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, pp. 2590–2596. AAAI Press, 2017.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022, 2007.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

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

A Appendix

A.1 Environment Settings

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 uniformly 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. 2. We also provide the complete code in the supplementary material.

Table 2: 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

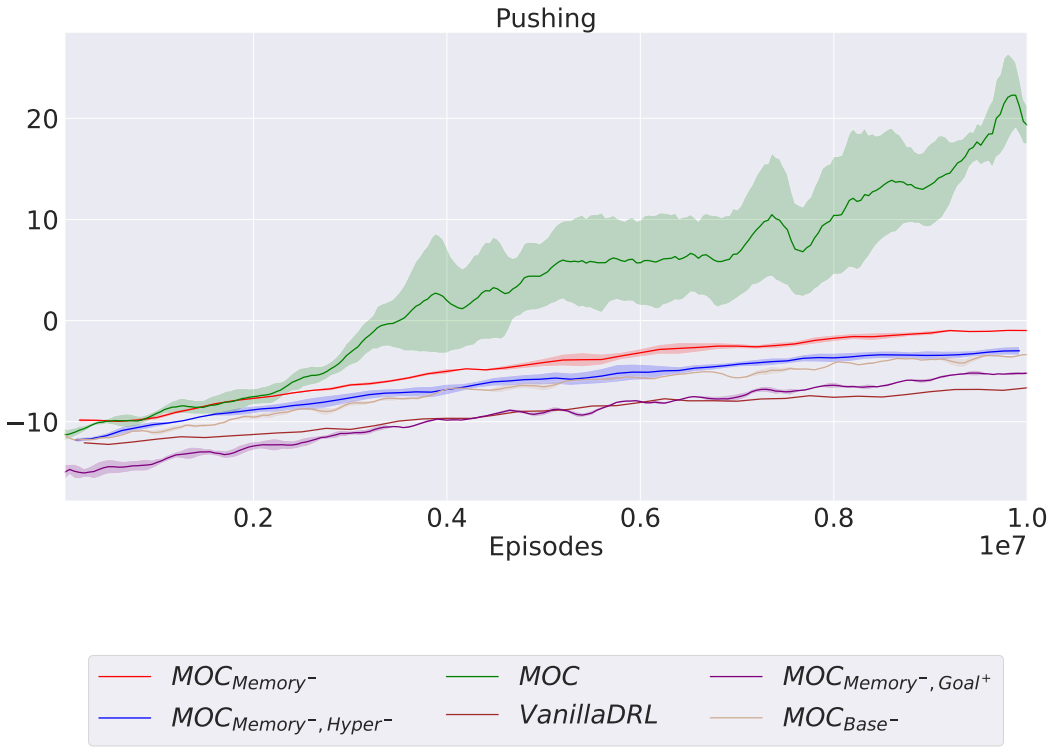


Figure 4: Comparison of algorithms with and without memory component in *pushing*. Each learning curve is computed in three runs with different random seeds.

A.2 Additional Experimental Results

This section serves as a supplementary results for Sec. 4.

Fig. 4 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.

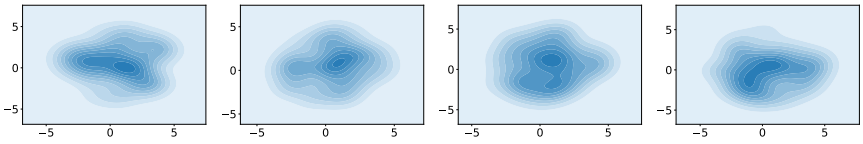
Fig. 4 clearly validate the effectiveness of our proposed method in achieving both the best final performance and improving sample efficiency.

A.3 Additional Visualizations of States

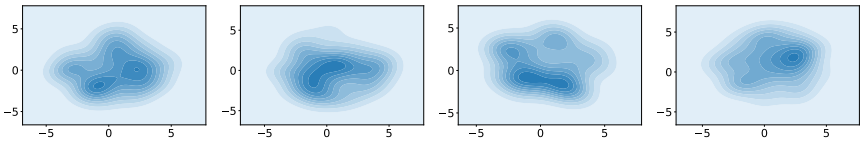
Figs. 5, 6, 7, 8 visualize the state visitation density in task `reaching`, `picking`, `pushing` and `pick and place`, respectively.

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. 6 and Fig. 7. (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. 6 and Fig. 7.

A.4 Additional Experiment Results

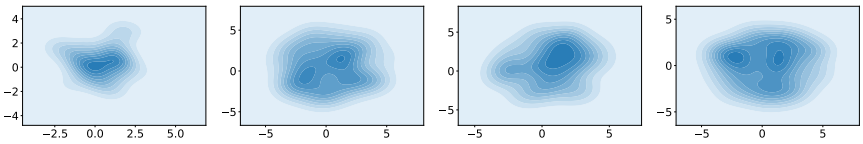


(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)

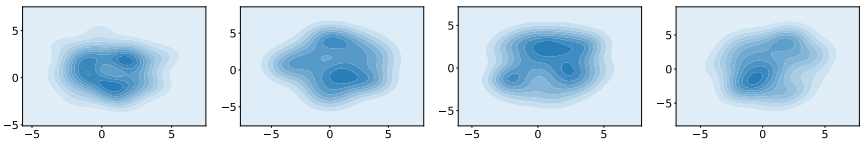


(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

Figure 5: 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 6: Visualizations of state visitation density in early and late stages in *picking*

In Sec. 4.2, we compared MOC with state-of-the art ACL algorithms. Here, we add two more baselines algorithms. The results are shown in Fig. 12:

- InitailGAN (Florensa et al., 2017a): 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.5 PPO Modifications

In Sec. 3, 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:

$$\theta_{k+1} = argmax_{\theta}\mathbb{E}_{s,a\sim\pi_{\theta_k}}[min(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a))], \tag{3}$$

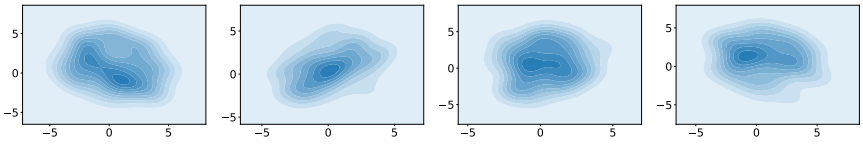
where

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

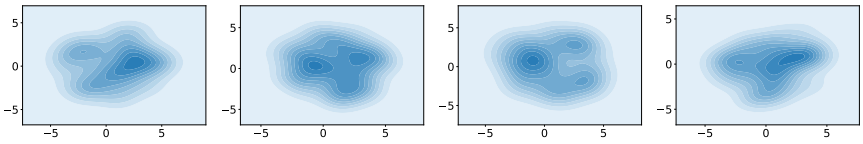
where θ is the parameter of policy π, θ_k is the updated k step parameter by taking the objective above, A is the advantage function that we define as:

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

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

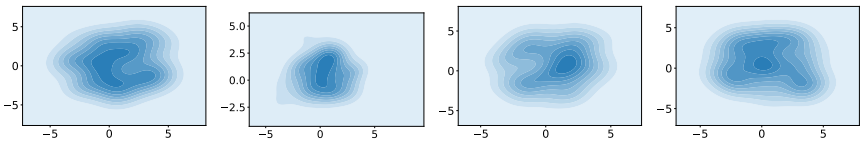


(a) no-mem (early) (b) mem-only (early) (c) independent (early) (d) with-mem (early)

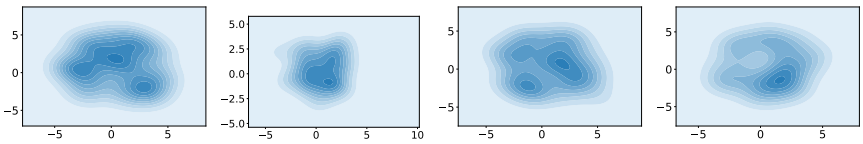


(e) no-mem (late) (f) mem-only (late) (g) independent (late) (h) with-mem (late)

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



(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 *pick* and *place*

A.6 BILEVEL TRAINING

Here we provide more details regarding the bilevel training of Hyper-RNN introduced in Sec. 3.3. 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. 3

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.7 HYPER-NET

(Ha et al., 2017) 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 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

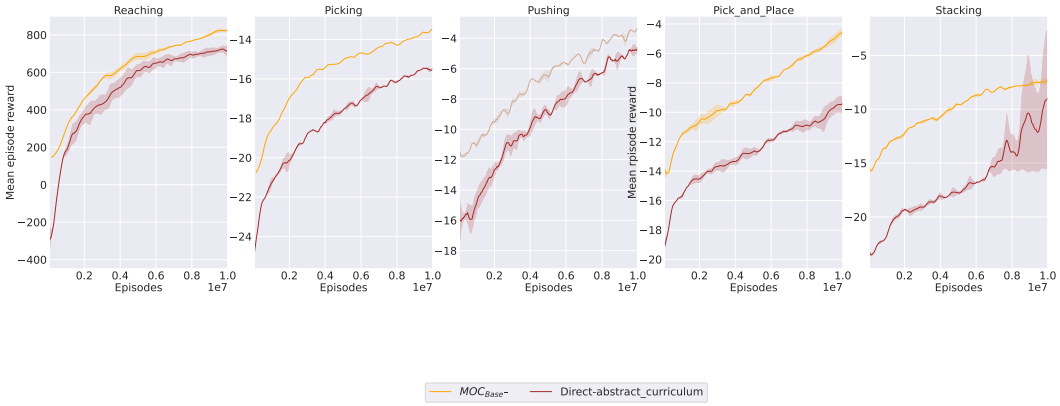


Figure 9: Comparison between read memory from memory and direct generate abstract curriculum

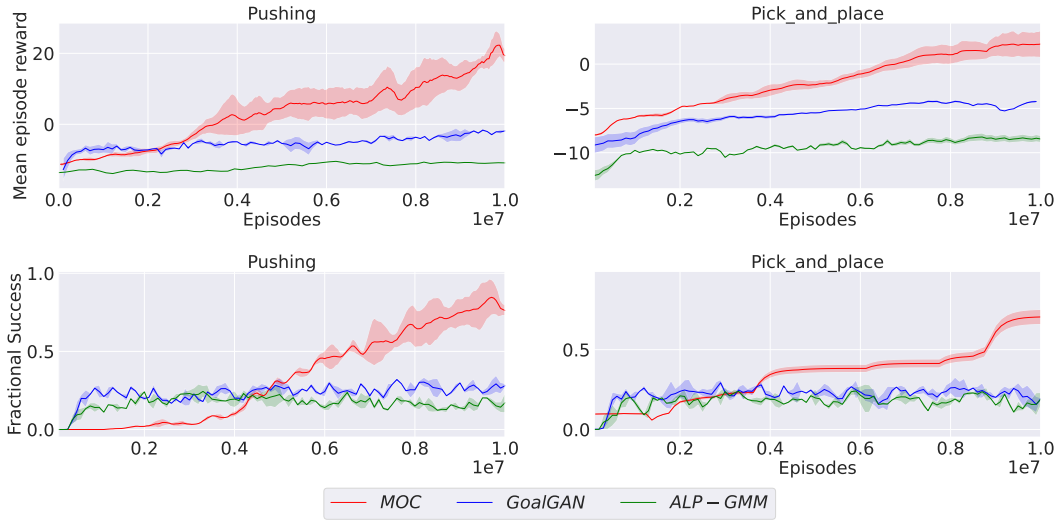


Figure 10: Comparison with ACL algorithms. Each learning curve is computed in three runs with different random seeds.

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:

$$\begin{aligned} h_t &= \phi(W_h(z_h)h_{t-1} + W_x(z_x) + b(z_b)), where \\ W_h(z_h) &=< W_{hz}, z_h > \\ W_x(z_x) &=< W_{xx}, z_x > \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned} \tag{4}$$

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}:

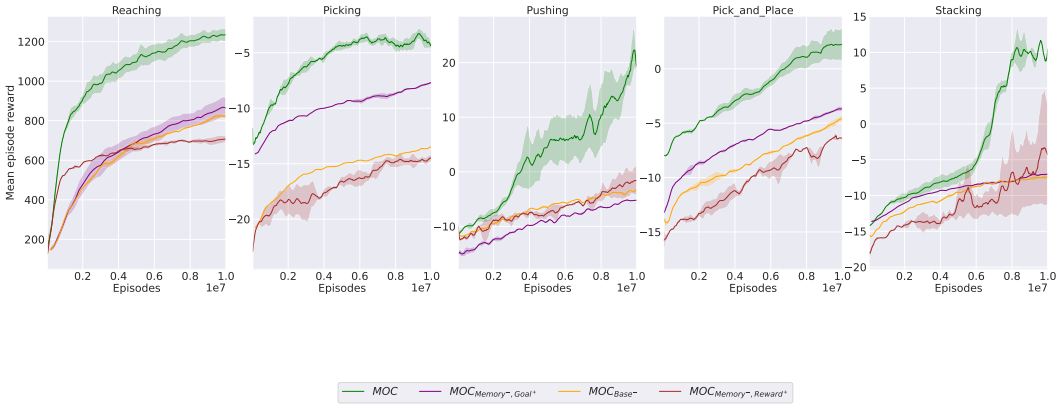


Figure 11: Comparison with reward curriculum only.

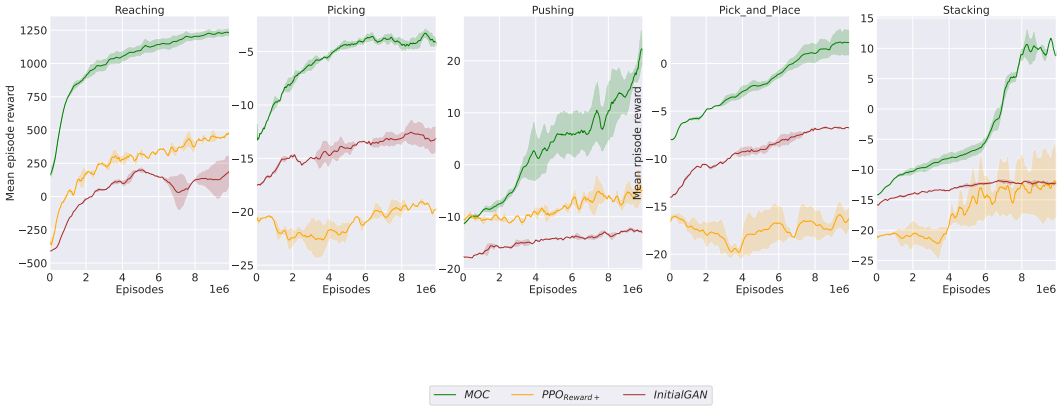


Figure 12: Comparison with Initial GAN and PPO with reward shaping only.

$$
\begin{aligned}
\hat{x}_t &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
\hat{h}_t &= \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b}) \\
z_h &= W_{\hat{h}h} \hat{h}_{t-1} + \hat{b}_{\hat{h}h} \\
z_x &= W_{\hat{h}x} \hat{h}_{t-1} + \hat{b}_{\hat{h}x} \\
z_b &= W_{\hat{h}b} \hat{h}_{t-1}
\end{aligned}
\tag{5}
$$

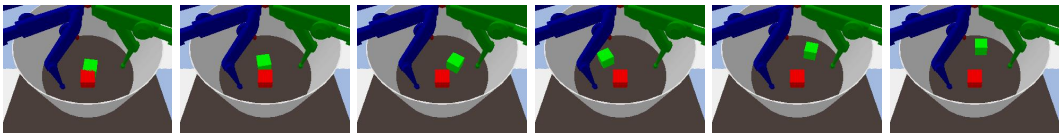
Where $W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, b \in \mathbb{R}^{N_{\hat{h}}}$, and $W_{\hat{h}h}, W_{\hat{h}x}, W_{\hat{h}b} \in \mathbb{R}^{N_z \times N_{\hat{h}}}$ and $\hat{b}_{\hat{h}h}, \hat{b}_{\hat{h}x} \in \mathbb{R}^{N_z}$.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

A.8 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[6].

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. The details can be found in our source code.

[6]The weight initialization approach (Chang et al., 2019) 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 13: Visualization of generated subgoals

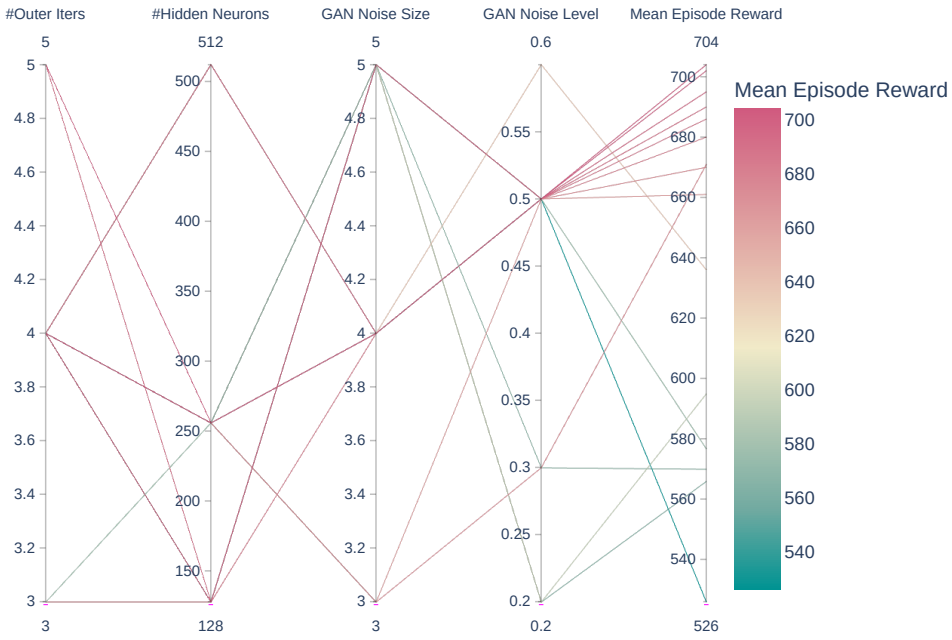
A.9 THE VISUALIZATION OF GENERATED SUB-GOAL

The visualization of generated sub-goal state is shown in Fig. 13. 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.

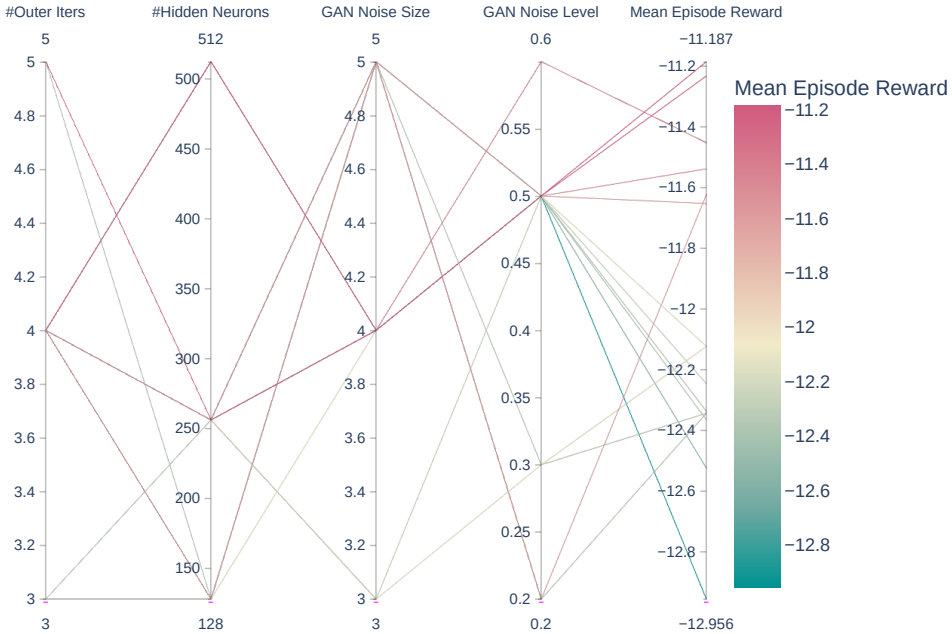
A.10 HYPERPARAMETERS

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. 14, 15, 16. For example, in Fig. 14-(a), the first column represents the 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 (Raffin et al., 2019) 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.

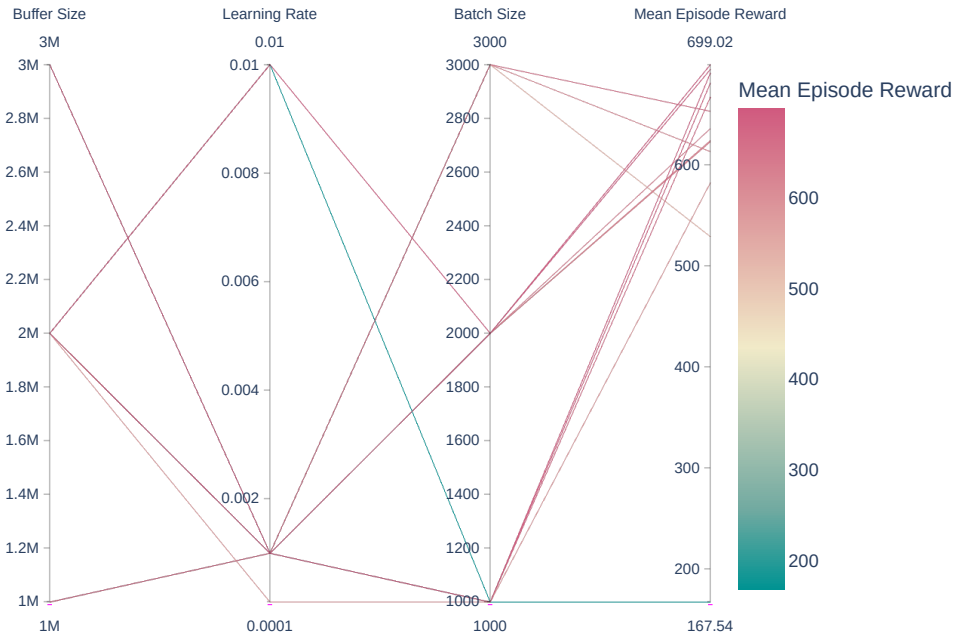


(a) Hyperparameter tuning in reaching task.

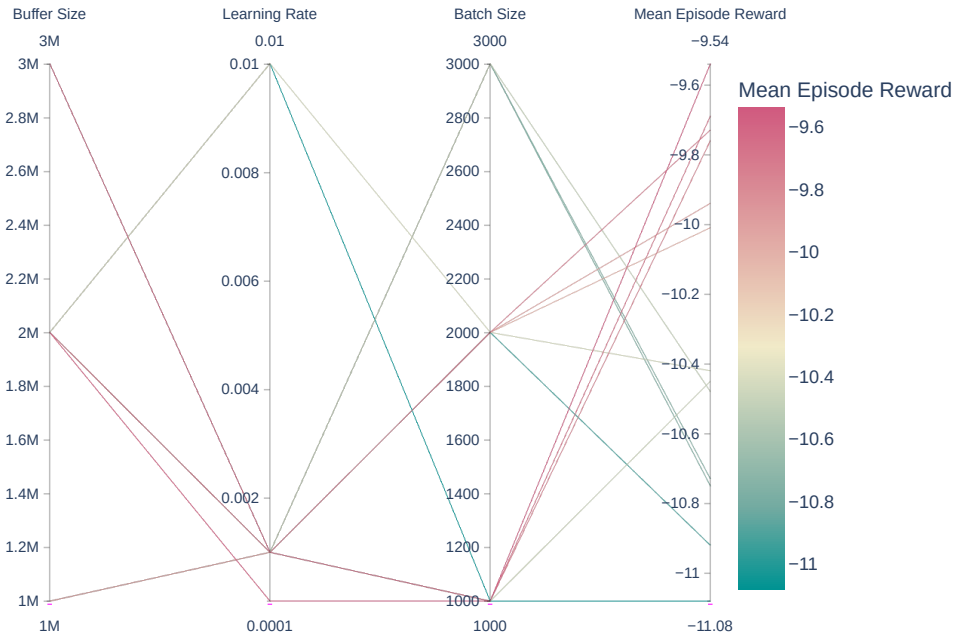


(b) Hyperparameter tuning in stacking task.

Figure 14: Hyperparameter tuning results for GoalGAN

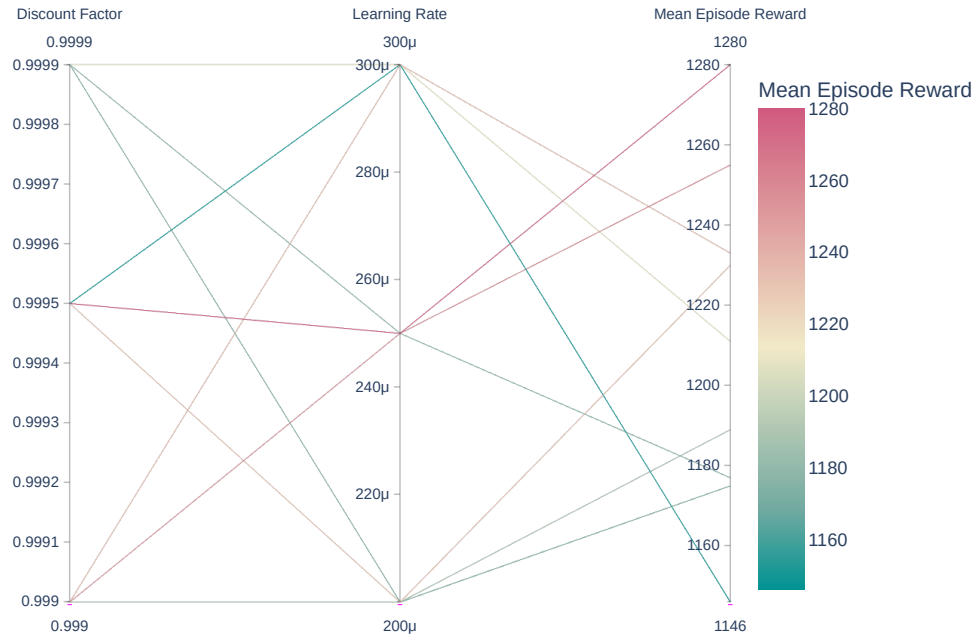


(a) Hyperparameter tuning in reaching task.

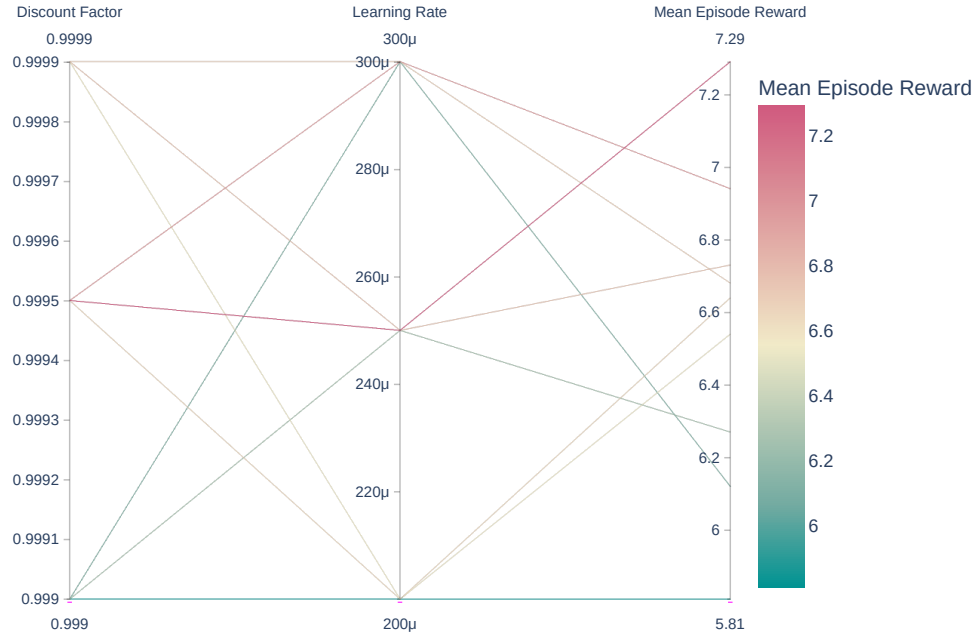


(b) Hyperparameter tuning in stacking task.

Figure 15: Hyperparameter tuning results for ALP-GMM



(a) Hyperparameter tuning in reaching task.



(b) Hyperparameter tuning in stacking task.

Figure 16: Hyperparameter tuning results for MOC