Model-Based Meta Automatic Curriculum Learning

Anonymous submission

Abstract

Curriculum learning (CL) has been widely explored to facilitate the learning of hard-exploration tasks in reinforcement learning (RL) by training a sequence of easier tasks, often called *a curriculum*. While most curricula are built either manually or automatically based on heuristics, e.g. choosing a training task which is barely beyond the current abilities of the learner, the fact that similar tasks might benefit from similar curricula motivates us to explore meta-learning as a technique for curriculum generation or *teaching* for a distribution of similar tasks. This paper formulates the meta CL problem that requires a *meta-teacher* to generate the curriculum which will assist the *student* to train toward any given *target task* from a task distribution based on the similarity of these tasks to one another. We propose a model-based meta automatic curriculum learning algorithm (`MM-ACL`) that learns to predict the performance improvement on one task when the student is trained on another, given the current status of the student. This predictor can then be used to generate the curricula for different target tasks. Our empirical results demonstrate that `MM-ACL` outperforms the state-of-the-art CL algorithms in a grid-world domain and a more complex visual-based navigation domain in terms of sample efficiency.

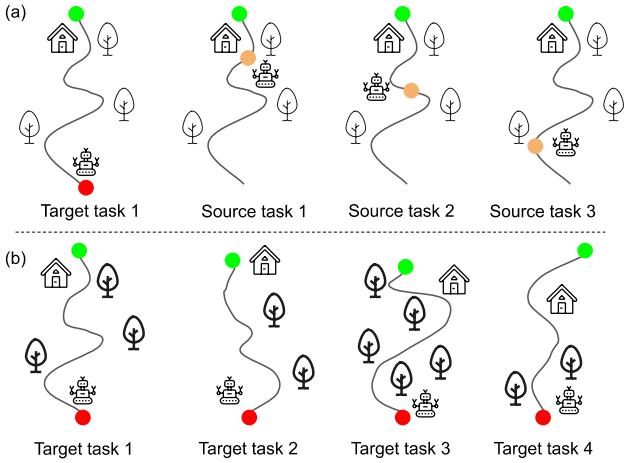

Figure 1: A navigation domain with a robot navigating from red to green markers, (a) a sample target task with a set of simpler source tasks (starting from the orange markers) that form the curriculum; (b) examples of four different target tasks in the same navigation domain.

1 Introduction

Deep Reinforcement Learning (DRL) is becoming a popular paradigm for solving complicated sequential decision tasks. However, despite its many advances (Mnih et al. 2013; Hausknecht and Stone 2015; Haarnoja et al. 2018), DRL techniques still struggle in tasks with sparse reward signals, which prohibits its further use in real-world tasks. Curriculum Learning has been proposed to alleviate these difficulties in DRL by generating a sequence of *source tasks* for an RL agent, a *student*, to train on in order to facilitate the student's training on a target task (Narvekar et al. 2020). Fig. 1 (a) shows a simple example of a student learning to navigate from a starting location (red) to a goal location (green) with a potentially long navigation trajectory. Such hard-exploration task will benefit from source tasks 1–3 that gradually move the starting locations (orange) further from the goal location.

While initial studies on CL in reinforcement learning used manually-designed curricula, based on humans' knowledge of the domain (Sanger 1994), the DRL community has developed a family of mechanisms called Automatic Curriculum Learning (ACL), which automatically adapts the distribution of source tasks according to the capabilities of learning agents (Portelas et al. 2020b). Such an adaptation of the source task distribution, however, is typically performed with heuristics or surrogate objectives. For example, `GoalGAN` trains a generator to generate source tasks with intermediate difficulty levels, which can avoid both too hard tasks that are infeasible to learn and too easy tasks that do not provide any novelty (Florensa et al. 2018). These heuristics are often suboptimal and can even be arbitrarily bad. For example, the definition of intermediate difficulty level often relies on a manually-picked threshold of the evaluation metrics, which can lead to transitioning too early to hard tasks that are actually not learnable. In addition, estimating the heuristics can be costly due to random exploration. `GoalGAN` tries random tasks, measures the difficulty levels, and updates the generator that generates the tasks of intermediate difficulty.

Alternatively, the CL problem can naturally be viewed as a *meta-learning* problem with an *inner loop* training the agent on each source task, and an *outer loop* updating the

source task or the teacher module such that an *outer objective* of maximizing the agent's performance on the target task will be improved. Such an explicitly defined outer objective forms the key to meta-learning (Hospedales et al. 2020). However, this meta-learning view of CL or meta-CL still remains mostly unexplored.

To solve such a bi-level problem, meta-learning usually trains the meta-representation (teacher) in an *offline* setting (Hospedales et al. 2020). Denoting the complete process of a student learning a target task *from scratch* under the supervision of a teacher as a *lifetime*, the teacher is trained in an offline manner over multiple lifetimes, where each lifetime corresponds to a new student learning a different target task. This process is usually called the *meta-training* phase. Then, the teacher is tested in new lifetimes with novel, but related target tasks, which is called the *meta-testing* phase. Looking back to the previous example in Fig. 1 (b), the meta-training phase will have the student taught by the teacher to learn a class of similar navigation tasks but with different landscapes (target tasks 1-3). Such teaching experiences can improve and prepare the teacher to teach a new target task 4 from this navigation domain. In this paper, we first formulate such a meta-CL problem that requires meta-training a teacher in an *offline* setting, which differs from an *online* setting where the teacher and the student co-evolve within one single lifetime.

To solve the meta-CL problem, we introduce a model-base meta automatic curriculum learning (MM-ACL) algorithm that is applicable with any policy gradient RL algorithm. MM-ACL selects the training tasks that will, to the greatest extent, improve the performance of the student on a given target task. To estimate the improvement, MM-ACL maintains a *learning dynamics model* which predicts the improvement on a given validation task after training on a given source task for some number of gradient updates. During the meta-training phase, MM-ACL performs "online validation" on a validation task both before and after the training of a selected source task. Then the improvement of the validation task given a source task can be measured and used as a data point for learning the learning dynamics model. Instead of learning a task selection model directly, learning such a learning dynamics model helps to improve sample efficiency and generalize more easily to different target tasks by training such a learning dynamics model.

We test MM-ACL on two simple MiniGrid domains and one more complex AI Habitat domain. The method is compared with a basic multi-task training baseline, an RL-based meta-curriculum learning algorithm, a handcrafted curriculum, and two state-of-the-art ACL algorithms. MM-ACL outperforms the baselines in terms of sample efficiency, and requires fewer meta-training lifetimes compared to the RL-based meta-curriculum learning algorithm.

2 Related Work

2.1 ACL for RL

MM-ACL solves a meta-CL problem that reuses previous teaching experiences for future target tasks, which differs from existing ACL algorithms that focus on online adaptation of the teacher for a specific target task or no target task (Dennis et al. 2020; Wang et al. 2020). The implications of this difference are two-fold: First, MM-ACL is an offline algorithm that trains a teacher module during a meta-training phase, then deploys the teacher without further updating on a specific set of target tasks. Existing ACL algorithms, on the other hand, are online algorithms which co-evolve teachers during the training of their students (Portelas et al. 2020a; Florensa et al. 2018). Second, existing ACL algorithms do not maintain an explicit outer-loop objective. The tasks are selected by heuristics or surrogate objectives which might not align with the objective of improving the learner's performance on the set of target tasks.

2.2 Meta-RL

One important line of meta-RL is meta-policy gradients (MPG), which utilizes differentiability of update function with respect to some hyper-parameters in the policy-gradient-based RL algorithms. MPG performs online cross validation (Sutton 1992) to collect hold-out trajectories to estimate outer-loop loss after every few gradient updates, then computes a meta-gradient of the hyper-parameters based on the outer-loop loss. Those hyper-parameters can be thought of as different tasks from a curriculum learning perspective, but are only limited to differentiable parameters, for example, discount factor (Xu, van Hasselt, and Silver 2018), intrinsic reward (Zheng, Oh, and Singh 2018), and auxiliary tasks (Veeriah et al. 2019). However, the parameters that facilitate learning most are sometimes directly related to the tasks. For example, transition dynamics and state space are not differentiable without knowing an analytic function of the transition dynamics. MM-ACL, however, can select the tasks that are controlled by any form of parameters including the non-differentiable ones. It deserves noting that, even though most MPG algorithms are online to make use of the efficiency of the meta-gradient, which differs from the offline setting employed by MM-ACL, MPG can also be performed offline by introducing episodic memory (Zheng et al. 2020).

Some existing meta-RL approaches do not use meta-gradients, and instead control specific aspects of RL that are generally non-differentiable, for example, the exploration-exploitation tradeoff (Ishii, Yoshida, and Yoshimoto 2002). They can also be thought of as a form of curriculum learning, but only limited to one specific aspect of the task, and therefore, do not fully answer the requirements of a CL approach.

2.3 Meta-ACL

There is also prior work that formulates the meta-CL problem in a similar fashion to the formalization presented in this paper. In CMDP's, introduced by Narvekar and Stone (2019), the teacher is represented as a curriculum policy and is optimized by an RL algorithm in the outer-loop optimization. Such a curriculum policy has also been demonstrated to be generalizable to different target tasks (Narvekar and Stone 2020). However, using RL as the outer-loop optimizer is extremely costly. Their approach uses hundreds of lifetimes to train an optimal curriculum policy with a tubular algorithm, where the low-level RL algorithm for learning was deployed in a simple grid-world domain. Modern deep RL algorithms may suffer from higher variance and longer training steps. Another

meta-curriculum learning framework AGAIN (Portelas et al. 2020c) improves the teacher from previous curricula based on *curriculum distillation*. However, AGAIN has two limitations: i) it is only defined for one particular base ACL algorithm, ALP-GMM (Portelas et al. 2020a), and cannot be adapted to different ACL algorithms; ii) It aims to optimize a heuristic of *absolute learning progress* (used by ALP-GMM) instead of tackling a meta-CL problem directly. Intuitively, based on the previous curriculum, AGAIN exploits the useful source tasks that fulfill the heuristic and prunes the bad tasks from random exploration to estimate the heuristic. This algorithm aims mostly to reduce the random exploration needed to estimate the heuristic when teaching a new student, which means that the algorithm is still limited to one specific heuristic.

Meta-CL has also been explored for supervised learning (Fan et al. 2018; Jiang et al. 2018). However, curriculum learning for RL encounters special challenges, such as high variance and exploration, that requires specially designed algorithms to tackle.

Lastly, in contrast to our work, which uses meta-learning to improve curriculum learning, some prior work focuses on curriculum learning for meta-learning (Stergiadis, Agrawal, and Squire 2021; Wu et al. 2021). More specifically, this line of work aims at automatically adapting the training task distribution for generic meta-learning (not necessary CL) problems. These types of algorithms are similar to prior ACL algorithms without any meta-learned component.

3 Preliminaries

3.1 Meta-CL domain

Meta-curriculum learning aims to leverage a set of target tasks during the meta-training phase to learn *how to teach*, i.e. gain knowledge that can improve the learning of the agent on new target tasks that were not seen during meta-training. We use $\mathcal{M} := \{m\}$ to denote a meta-CL domain defined by a set of all possible tasks m, from which a target task m_{target} can be chosen. For example, in the navigation domain shown in Fig. 1, a target task m is uniquely defined by a combination of a navigation environment and start-goal locations. All the possible combinations constitute a meta-CL domain $\mathcal{M}$. However, it is often the case that the teacher can only control some aspects of the task when solving a given target task. For example, when the navigation robot in Fig. 1 is situated in the environment specified by the target task, the teacher can change the start-goal locations, but may not be able to move the student to a completely different environment. For this reason, we factorize a meta-CL domain into two orthogonal spaces: a controllable space $\mathcal{W}$ and an uncontrollable space $\mathcal{V}$. A task m can, therefore, be represented as a tuple (w, v) with $w \in \mathcal{W}$ and $v \in \mathcal{V}$. Considering the same example of the navigation domain, w and v are the variables that specify the navigation environment and the start-goal locations respectively. For a given target task $m_{target} = (w_{target}, v_{target})$ in a lifetime, a teacher can control the variable w to generate a sequence of source tasks as a curriculum that can facilitate the learning of m_{target}.

3.2 Student

We consider a student in a meta-CL problem as an RL algorithm that can improve its policy based on the trajectories collected from a given training task. We use ψ to denote the current status of a student, which contains all the internal states of the RL algorithm, such as all the weights of the actor and critic neural networks and the states of the optimizer if the algorithm is an actor-critic deep RL algorithm based on deep neural networks. Such a student can be updated from status ψ_t to ψ_{t+1} during a training iteration t based on a given training task m_t, which we formulate as follows:

$$(\psi_{t+1}, \tau_t) = \textit{Train}(\psi_t, m_t) \tag{1}$$

Here, *Train* represents a training process that updates the student ψ, and τ_t is the rollout trajectory at iteration t. For example, the training process of REINFORCE (Sutton and Barto 2018) with a linearly-decayed learning rate is a simple policy gradient update combined with a learning rate update:

$$\tau_t \sim (\pi_{\theta_t}, m_t), \;\; \alpha_{t+1} = \alpha_t - 0.0001,$$
$$\theta_{t+1} = \theta_t - \alpha_t \nabla_{\theta_t} \mathcal{G}(\tau_t)$$

with $\psi_t = (\theta_t, \alpha_t)$ being the student status, θ_t the parameters of the policy π_{θ_t}, τ_t the trajectories rolled out from task m_t using policy π_{θ_t}, α_t the learning rate, and $\mathcal{G}(\tau_t)$ the returns of τ_t. Then, during a lifetime lasting N training iterations with a curriculum $(m_1, m_2, ..., m_N)$, the student, starting from an initial status ψ_0, is updated recursively following Eq. 1, which generates a sequence of student statuses $(\psi_0, \psi_1, ..., \psi_N)$.

3.3 Meta-teacher

A meta-teacher will dynamically decide on which source tasks to train based on the information of (1) *which student is being trained* and (2) *what is the target task*. As defined in Sec. 3.1 and 3.2, the student and the target task can be represented as ψ and (w_{target}, v_{target}) respectively. However, for complex domains solved by deep RL algorithms, it might be prohibitive to take ψ as input since it contains all the neural network weights. Instead, in the ACL literature, a more commonly used representation of a student status is *contextual information* that comprises the training history of the student. More specifically, a context $c = (w_j, \tau_j)_{j=1}^{J} \in \mathcal{C}$ is a sequence of J training task-trajectory pairs experienced so far, where w_j is the set of controllable parameters that define a source task (w_j, v_{target}), τ_j is the rollout trajectory, and $\mathcal{C}$ is a set of all possible contextual information.

In this paper, we use the second type of student representation to keep the compatibility with modern RL training methods. Formally, we define a meta-teacher $f : \mathcal{C} \times \mathcal{M} \times \mathcal{W} \to \mathbb{R}$ as a mapping from a context in $\mathcal{C}$ and a target task in $\mathcal{M}$ to a probability distribution over controllable parameters $\mathcal{W}$. With the teacher being defined, Eq. 1 can be modified to indicate teacher-supervised training as follows:

$$(\psi_{t+1}, \tau_t) = \textit{Train}(\psi_t, m_t) = \textit{Train}(\psi, w_t, v_{target}),$$
$$\text{with } w_t \sim f(c_t, m_{target}, w)$$

3.4 The Meta-CL Problem

To evaluate a meta-teacher function f's performance when teaching a given target task m_{target} over a student's lifetime,

we measure the return of the student $\hat{\psi}_N(f)$ on the target task trained after an entire lifetime of N training iterations, where the functional form $\hat{\psi}_N(f)$ is used to indicate the dependence of the student's status on the teacher f. We use $\mathcal{G}(\hat{\psi}_N(f), m_{target})$ to denote such an evaluated return. Then, a meta-CL problem can be defined as finding an optimal teacher f^* such that the expectation of students' evaluated returns over a distribution of target tasks $p(m)$ is maximized:

$$f^* = \arg\max_f [\mathbb{E}_{m_{target} \sim p(m)} [\mathcal{G}(\hat{\psi}_N(f), m_{target})]] \quad (2)$$

An algorithm that solves the meta-CL problem is allowed to freely sample target tasks from $p(m)$ and teach the students on the target tasks during a meta-training phase. These teaching experiences are used to update the teacher so that the objective defined in Eq. 2 will be improved. During the meta-testing phase, the teacher is tested by having it train new student lifetimes on unseen target tasks sampled from the same distribution $p(m)$.

4 Approach

4.1 Local Learning Progress

The meta-CL problem requires optimizing the teacher f based on an objective that can be evaluated only after an entire lifetime training of N iterations (Eq. 2). Such a long-term objective, usually referred to as maximizing *global learning progress* (Portelas et al. 2020b), is typically considered intractable in hard-exploration domains that may require a few thousand iterations to solve, even with a good curriculum.

Instead, we consider a short-term objective that maximizes *local learning progress*, which is defined as the improvement of the target task performance after training on a source task for one or a few iterations. The local learning progress gained by training on a source task w_t (selected for the t-th iteration) for k additional iterations, is denoted as $\mathcal{P}_t^k(w_t, w_{target}, v_{target})$ and formulated as follows:

$$\mathcal{P}_t^k(w_t, w_{target}, v_{target}) = \mathcal{G}(\psi_{t+k}, w_{target}, v_{target}) - \mathcal{G}(\psi_t, w_{target}, v_{target}),$$

where ψ_{t+k} is computed recursively by:

$$\psi_{t+i+1} = \textit{Train}(\psi_{t+i}, w_t, v_{target}), \ \ \forall i = 0, ..., k-1$$

4.2 Learning Dynamics Model

The short-term objective of maximizing the local learning progress is analogous to some MPG algorithms that optimize the meta-representations based on online validations with a short interval of K-step iterations (Xu, van Hasselt, and Silver 2018). Such algorithms only consider meta-representations that are differentiable w.r.t. the *local learning progress*, which requires that a derivative $\nabla_{\eta_t} \textit{Train}(\phi_t, \eta_t)$ of the *Train* process w.r.t. some meta-representation η can be computed. In our case, given the meta-representation w_t, the meta-gradient update can be used for choosing the source task at the $t + k^{th}$ iteration as follows:

$$w_{t+k} = w_t - \beta \nabla_{w_t} \mathcal{P}_t^k(w_t, w_{target}, v_{target})$$

However, MPGs are not applicable in CL, as the *Train* process is usually non-differentiable with respect to the the parameters w_t that control the source tasks. An alternative approach is to approximate the local learning progress $\mathcal{P}_t^k(w_t, w_{target}, v_{target})$ with a black box function, and then choose the best controllable parameters w_t based on the approximated learning progress. Instead of approximating $\mathcal{P}_t^k$ on the target task only, we learn a general function that captures the learning progress from w to any other w'. Formally, we define a *learning dynamics model* that approximates $\mathcal{P}_t^k(w, w', v_{target})$ with a function $g^k(c, w, w', v_{target})$, where c is the contextual information introduced in Sec. 3.3, (w, v_{target}) is the source task that the agent is trained on for k iterations, and (w', v_{target}) is the validation task the agent is evaluated on. Once a learning dynamics model $g^k(c, w, w', v_{target})$ is learned, a teacher picks source tasks by sampling the controllable parameters w with the probabilities proportional to the local learning progress on the target task.

4.3 MM-ACL

The learning dynamics model $g^k(c, w, w', v_{target})$ depends on v_{target} (as well as c, w, and w'). However, it is usually infeasible to represent the complicated uncontrollable parameter space needed to encode the environment. For example, in real-world navigation tasks, the uncontrollable parameters v specify the environment that includes all the locations, shapes, and visual textures of the objects, which are very hard to encode. As a result, in `MM-ACL`, we ignore the dependence of the learning dynamics model on the uncontrollable parameters v_{target}, and attempt to learn a dynamic model of the form $g_\phi^k(c, w, w')$ with a regression model parameterized by ϕ. In practice, some information regarding the uncontrollable parameters is still preserved, as it can be implicitly encoded by the contextual information that contains past observations and interactions with different source tasks. The neural network architecture of the learning dynamics model is shown in Fig. 2 (a). Each task-return pair in the contextual information[1] is embedded by one fully-connected layer and is fed into a LSTM (Gers, Schmidhuber, and Cummins 2000) as a sequence of task-return embeddings. Then, the output from the LSTM is concatenated with the embedding of w and w' followed by a MLP to compute the prediction of local learning progress $\mathcal{P}_t^k(w, w', v_{target})$ on the validation task w'. More details of the learning dynamics model are shown in Appendix A.1.

In the Appendix A.2, algorithm 1 shows the whole pipeline for training g_ϕ^k. The training data is stored in a replay buffer D collected from learners trained from scratch in multiple i.i.d. lifetimes. At each iteration t of each lifetime, a training task w^{train} is selected based on the `SelectTask` function (Algorithm 2) that has an exploration probability p to select the task based on the current *learning dynamics model* and a probability $1 - p$ to select a randomly sampled exploratory task (lines 6-7). Once a training task w^{train} is selected, the

[1]The full contextual information is tasks and entire trajectories, but we found that simply taking tasks and returns evaluated on the trajectories is sufficient for most problems.

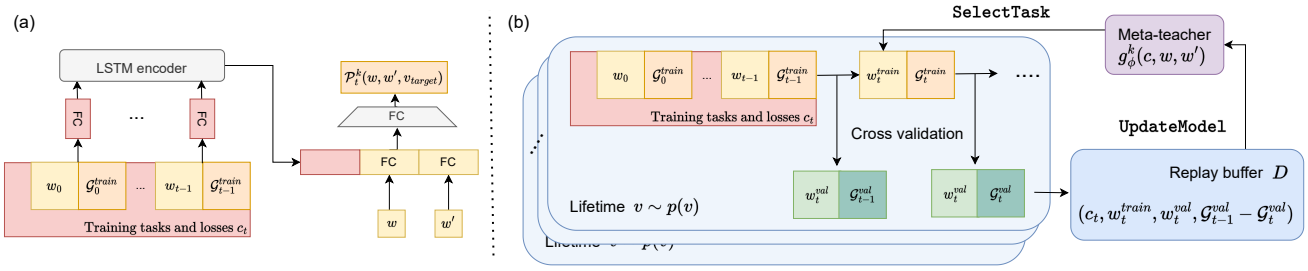

Figure 2: (a) the neural network architecture of the learning dynamics model; (b) The meta-training pipeline of MM−ACL.

student is updated for k iterations and the history of tasks and returns is appended to the contextual information (lines 8-12). Then, the policies before and after the update are evaluated on the same randomly sampled validation task w^{val}, resulting in the returns $\mathcal{G}_t^{val}$ and $\mathcal{G}_{t+k}^{val}$, respectively (lines 13-16). The contextual information, training task, validation task, and learning progress constitute a training data point $(c_t, w^{train}, w^{val}, \mathcal{G}_{t+k}^{val} - \mathcal{G}_t^{val})$, which is added into the replay buffer D (line 17). At every fixed interval of iterations, the parameter ϕ is updated based on the training data in the replay buffer D (lines 19-21). Once the *learning dynamics model* is learned, the model can be deployed online following the SelectTask function with exploration probability $p = 0$. During the deployment, the contextual information c_t is built in the same way as line 12 in Algorithm 1. Notice that during deployment, lines 13 to 21 in Algorithm 1 that perform online validations are not executed; this omission guarantees that the algorithm does not add extra rollouts and gradient updates. A diagram of the meta-training pipeline is shown in Fig. 2 (b).

5 Experimental Results

We test MM−ACL in a grid-world domain and a visual-based navigation domain. The domains are designed such that some form of curriculum is necessary, as training on the target task directly will always fail. We introduce the domains in detail in their corresponding sections. To evaluate the performance of ACL algorithms in these domains, we use a metric of *iterations to threshold*, which is formally defined as follows:

Definition 1 (Iterations to threshold) *Given a completed lifetime with a target task m_{target} and a sequence of student statuses $(\psi_i)_{i=1}^N$, the iterations to threshold (ItT) is defined as the minimum index i such that the student's past-K evaluated returns on the target task are all greater than or equal to a certain threshold δ with K being a history length. Denoting $\xi = \{i \mid i \in [K, N],\ \mathcal{G}(\psi_{i-j}, m_{target}) \geq \delta,\ \forall j \in [0, K]\}$,*

$$ItT = \begin{cases} \min_i\{i \in \xi\} & \xi \neq \emptyset \\ N & \text{otherwise.} \end{cases} \quad (3)$$

An algorithm with a low ItT is said to have a high sample efficiency, and vice versa.

The performance of MM−ACL is compared with four different baselines: (1) ALP−GMM; (2) a random curriculum; (3) a handcrafted curriculum; and, (4) a DQN-based method for solving CMDPs (as done by Narvekar and Stone (2020)), denoted as DQN−CMDP. Note that ALP−GMM is designed for domains of continuous control. To apply it in discrete domains, we modified the bandit algorithm from Matiisen et al. (2019) but used the same heuristic as ALP−GMM. For simplicity we refer to this customized bandit algorithm as ALP−GMM as well. In addition, due to the large amount of data required to train DQN−CMDP, we only compare with it in the grid-world domain. In all the experiments, the students are PPO agents (Schulman et al. 2017) with their policy neural networks and hyper-parameters tuned for each of the domains, as detailed in Appendix B.

5.1 FourRoom domain

We test MM−ACL in the grid-world **FourRoom** domain based on the widely used MiniGrid environment by Chevalier-Boisvert, Willems, and Pal (2018). As shown in Figure 3, the target tasks in this domain require an agent to navigate through four rooms (labeled as room 0, 1, 2, and 3 from left to right) and reach a goal location (green tiles) in the right-most room (room 3) by executing one of the four discrete actions: move forward, turn left, turn right, and open the door. Three types of skills are required to pass the rooms: (1) navigate around the blocks (grey tiles) which the agent cannot overlap with; (2) avoid tiles covered with lava (orange tiles) that cause immediate failure of an episode once the agent visit them; (3) open the door that blocks the way to the goal location. The only positive reward of $1 - 0.0009 * T$ is assigned upon reaching the goal location, where T is the total number of time steps. In this domain, a threshold $\delta = 0.92$ and a history length $K = 4$ are used for measuring the ItT.

As shown in Fig. 3, different tasks in this domain may have different room configurations (e.g., openings on the walls) and different start-goal locations. These two types of variations correspond to the uncontrollable space $\mathcal{V}$ and controllable space $\mathcal{W}$ respectively. We assume that the start-goal locations can only be varied in a limited way: w can only specify the rooms from which the start and goal locations are randomly sampled. More specifically, the controllable space $\mathcal{W} = \{0, 1, 2, 3\}$ is a four-category discrete set with w equal to 0, 1, 2, 3 denoting the tasks of navigating from room 0 to 3, room 1 to 3, room 2 to 3, and room 1 to 2, respectively. The uncontrollable space $\mathcal{V}$ is all the possible

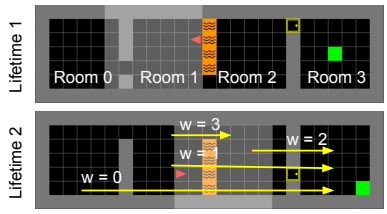

Figure 3: Examples of **FourRoom** environments from two different lifetimes. The yellow arrows indicate the start and goal rooms for different controllable parameters w.

	FourRoom	**Habitat**
MM-ACL	$\mathbf{129 \pm 25}$	$\mathbf{82 \pm 21}$
DQN-CMDP	202 ± 39	-
ALP-GMM	158 ± 28	105 ± 32
Handcrafted	175 ± 84	102 ± 18
MM-ACL-single	175 ± 73	-
Random	213 ± 46	126 ± 54

Table 1: Average ItTs in **FourRoom** and **Habitat**.

room configurations. Different lifetimes in this domain have the target tasks with the same $w_{target} = 0$, but different v_{target} uniformly sampled from $\mathcal{V}$.

We applied MM-ACL to the **FourRoom** domain. The *learning dynamics model* of MM-ACL is trained for 150 lifetimes during the meta-training phase. Then, the model is tested in five different lifetimes with previously unseen target tasks. The hyper-parameters of MM-ACL can be found in Appendix A.3. During meta-testing, each lifetime is repeated for five independent runs with different random seeds. While training a student, the return of the student is evaluated on the target task for 16 episodes after every training iteration. Fig. 4 (left) shows plots of return on the target task at different training iterations collected from two meta-testing lifetime. As shown in the figure, MM-ACL achieves the best sample efficiency with the lowest ItTs in both lifetimes, followed by ALP-GMM whose ItT is 29 more. On the other hand, both the random and handcrafted curricula fail to reach the threshold for one or a few runs with a 400-iteration budget. The average ItTs over all five lifetimes are reported in Table 1.

5.2 Visual-based Navigation Domain: Habitat

We also test MM-ACL in a more complex visual-based navigation **Habitat** domain realized by AI Habitat (Savva et al. 2019), illustrated in Fig. 5. AI Habitat can simulate navigation in a variety of household environments, constructed from real-world houses (Chang et al. 2017), while providing photo-realistic camera images as observations. We focus on an object-oriented navigation task that requires the agent to navigate to a given type of object in the environment. At each time step, based on an observation of RGB-D image and target object index, the agent can take one of three actions: move forward by 0.25 meter; turn left by 30°, or turn right by 30°. The episode is considered successful and a reward of +1 is obtained if the agent's distance to the closet target object is less than 0.5 meter. In this domain, a threshold $\delta = 1$ and a history length $K = 4$ are used to measure the ItT.

During each lifetime in the **Habitat** domain, the agent has to search for one fixed type of object in the same environment. The uncontrollable space $\mathcal{V}$ is, therefore, all the possible object-environment pairs. The controllable parameter $w \in \mathcal{W}$ is a variable that specifies the *geodesic distance* to the closest target object, i.e., given a parameter value w, the teacher samples a starting location for the agent such that the shortest path between the agent and the closest object is exactly w. Lastly, the target tasks from different lifetimes have a random v_{target} sampled uniformly from $\mathcal{V}$, and w_{target} is the largest possible *geodesic distance* to the target object in the environment specified by v_{target}.

Similarly to **FourRoom**, we apply MM-ACL to this domain with the *learning dynamics model* trained over 80 lifetimes. Then, the model is deployed in five lifetimes, each repeated for five independent runs. When training a student, the return of the student is evaluated on the target task for eight episodes after every two training iterations. Fig. 4 (right) shows two plots of the return on the target task at different training iterations collected from two meta-testing lifetimes. As seen in the figure, MM-ACL achieves the best sample efficiency in both lifetimes, while both ALP-GMM and the handcrafted curriculum perform sub-optimally in one of the lifetimes.

6 Analysis

In this section, we conduct an analysis and ablation study. We focus our attention on the grid-world domains due to the lower computational cost compared to **Habitat**.

What curriculum does MM-ACL generate? To study why MM-ACL achieves lower sample efficiency, in Fig. 6, we plot the probability distributions used by MM-ACL and ALP-GMM to sample the tasks at different training iterations. As seen in the figure, MM-ACL focuses equally on $w = 3$ and $w = 2$ at the beginning, which correspond to the two basic skills of avoiding lava and opening the door. Then it switches to $w = 1$ that combines the two skills in the same tasks. Finally, MM-ACL turns to the target task $w = 0$ after about 50 iterations, when the target task is learnable. On the other hand, ALP-GMM heavily exploits $w = 2$ at the beginning since it gives the highest absolute learning progress, which actually slows down the learning of the target task.

How many lifetimes does MM-ACL need? To answer this question, we train MM-ACL with limited data collected from 10, 50, 100 and 150 lifetimes, and test it in the **FourRoom** domain, similarly to Sec. 5.1. The results are shown in Fig. 7 (left), in which the sample efficiency of ALP-GMM is marked as the red line. MM-ACL is outperformed by ALP-GMM with only ten lifetimes, but is comparable to ALP-GMM after 50 lifetimes and clearly outperforms it after 100 lifetimes.

Does MM-ACL generalize to different student statuses? As discussed in Sec. 3.3, a meta-teacher has to infer the student status from the contextual information if it does not have access to the full representation of the student status (e.g. the weights of the neural network). To study the ability of MM-ACL to infer and adapt to the current status of the

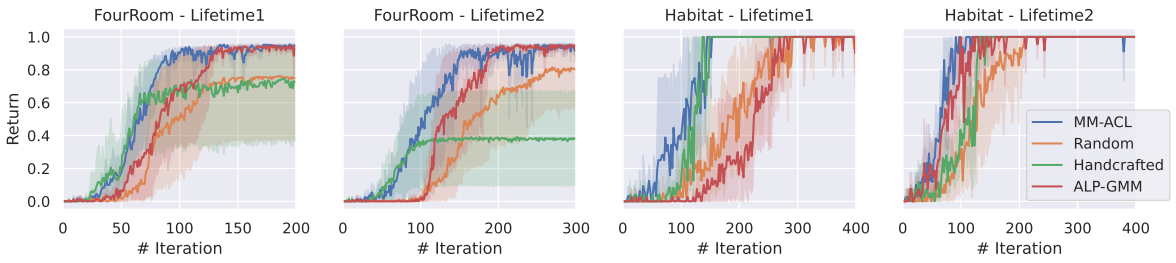

Figure 4: Average returns in the **FourRoom** and **Habitat** domains; the shaded areas show the 95% confidence interval.

Figure 5: (left) a top-down-view image of an agent's trajectory of finding a chair in an environment. The blue dot representing the start location, the light blue curve tracing the path taken by the agent until reaching within 0.5m of a chair, and the pink dots marking the locations of chairs; (right) the RGB and depth images observed by the agent.

student, we record a fixed history curriculum generated by MM−ACL in one of the runs, and deploy it on the same target task, but with five different random seeds that may lead to different student statuses. As shown in Fig. 7 (right), the curve of evaluated returns of MM−ACL is above the curve of the fixed curriculum throughout the training and shows much lower variance. On the other hand, the fixed curriculum shows higher variances which indicates a failure to adapt to different student statuses.

Does MM−ACL benefit from experiencing different target tasks during training? To verify that MM−ACL can improve its generalization to unseen target tasks from experiencing more target tasks during training, we conduct an ablation study that trains MM−ACL with the same number of 150

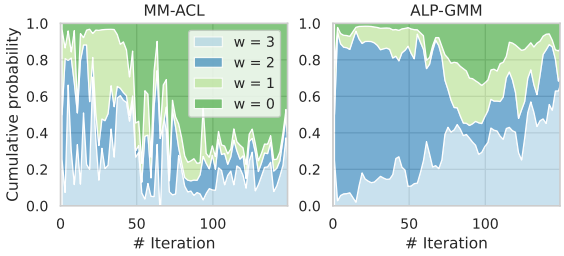

Figure 6: The probability distributions used by MM−ACL (left) and ALP−GMM (right) to sample source tasks at different training iterations in the same meta-testing lifetime.

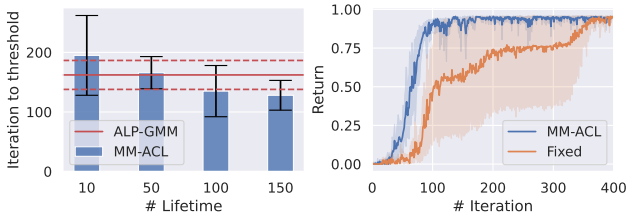

Figure 7: (left) the sample efficiencies of MM−ACL (blue) trained for 10, 50, 100, and 150 lifetimes, with the red solid line marking the sample efficiency of ALP−GMM, and the dashed lines marking its 95% confidence interval; (right) evaluation of the target task return of the students taught by MM−ACL and a fixed curriculum generated by MM−ACL.

lifetimes, with all the lifetimes having the same *single* target task. We denote the resulting model as MM−ACL−single, which is then evaluated in the same way as MM−ACL in Sec. 5.1. The results in Table 1 (rows 1 and 6) show that MM−ACL−single achieves a lower sample efficiency of 175 ItT compared to 129 by MM−ACL. Notice that this sample efficiency is similar to that of a handcrafted curriculum (row 5 in Table 1) designed for a few target tasks, which further supports the necessity of generalizing teachers to different target tasks.

7 Conclusion

This paper first formulates a new meta-CL problem, in which a teacher can be benefit from previous lifetimes' teaching experience to be able to teach new students and new target tasks. As an initial approach to solving this problem, a new meta-CL algorithm MM−ACL is proposed that learns a *learning dynamics model* that predicts the improvement on one task when trained on another, which can be utilized to select the source task that maximizes the improvement on the target task. Our empirical results show an improvement of sample efficiency compared to the state-of-the-art ACL algorithms and existing meta-CL algorithms. However, MM−ACL suffers from myopia that might lead to failures in cases in which training on multiple source tasks is required before observing any improvement in the student's performance on the target task. In addition, we only tested one type of contextual information based on the history tasks and returns; more informative contextual information based on the whole trajectories could also be considered in future work.

References

Chang, A.; Dai, A.; Funkhouser, T.; Halber, M.; Niessner, M.; Savva, M.; Song, S.; Zeng, A.; and Zhang, Y. 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)*.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33: 13049–13061.

Fan, Y.; Tian, F.; Qin, T.; Li, X.-Y.; and Liu, T.-Y. 2018. Learning to teach. *arXiv preprint arXiv:1805.03643*.

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

Gers, F. A.; Schmidhuber, J.; and Cummins, F. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10): 2451–2471.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hausknecht, M.; and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Ishii, S.; Yoshida, W.; and Yoshimoto, J. 2002. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6): 665–687.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, 2304–2313. PMLR.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lucas Willems, D. B. M. C.-B., Ye Yuan. 2013. RL Starter Files. https://github.com/lcswillems/rl-starter-files.

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M. E.; and Stone, P. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181): 1–50.

Narvekar, S.; and Stone, P. 2019. Learning Curriculum Policies for Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Narvekar, S.; and Stone, P. 2020. Generalizing Curricula for Reinforcement Learning. In *4th Lifelong Learning Workshop at the International Conference on Machine Learning (ICML 2020)*.

Portelas, R.; Colas, C.; Hofmann, K.; and Oudeyer, P.-Y. 2020a. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, 835–853. PMLR.

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

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

Sanger, T. D. 1994. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, 10(3): 323–333.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Stergiadis, E.; Agrawal, P.; and Squire, O. 2021. Curriculum Meta-Learning for Few-shot Classification. *arXiv preprint arXiv:2112.02913*.

Sutton, R. S. 1992. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, 171–176. San Jose, CA.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Veeriah, V.; Hessel, M.; Xu, Z.; Rajendran, J.; Lewis, R. L.; Oh, J.; van Hasselt, H. P.; Silver, D.; and Singh, S. 2019. Discovery of useful questions as auxiliary tasks. *Advances in Neural Information Processing Systems*, 32.

Wang, R.; Lehman, J.; Rawal, A.; Zhi, J.; Li, Y.; Clune, J.; and Stanley, K. 2020. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, 9940–9951. PMLR.

Wu, T.; Li, X.; Li, Y.; Haffari, G.; Qi, G.; Zhu, Y.; and Xu, G. 2021. Curriculum-Meta Learning for Order-Robust Continual Relation Extraction. In *AAAI*, 10363–10369. AAAI Press.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31.

Zheng, Z.; Oh, J.; Hessel, M.; Xu, Z.; Kroiss, M.; Van Hasselt, H.; Silver, D.; and Singh, S. 2020. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, 11436–11446. PMLR.

Zheng, Z.; Oh, J.; and Singh, S. 2018. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31.

Appendix: Model-Based Meta Automatic Curriculum Learning

A MM-ACL

A.1 Learning Dynamics Model

A data point that trains the learning dyanmics model is a 5-tuple $(c_t, w^{train}, w^{val}, \mathcal{G}_t^{val}, \mathcal{G}_{t+k}^{val})$, where c_t is the context at the t-th training iteration, $\mathcal{G}_t^{val}$ and $\mathcal{G}_{t+k}^{val}$ are the returns evaluated on the task w^{val} before and after training on the task w^{train} for k iterations. Then, the learning dynamics model $g_\phi(c_t, w^{train}, w^{val})$, parameterized by ϕ, is trained to predict the learning progress $\mathcal{G}_{t+k}^{val} - \mathcal{G}_t^{val}$ given an input of $(c_t, w^{train}, w^{val})$. More specifically, the parameters ϕ is optimized to minimize the mean square error of the prediction of learning progress evaluated over all the data points in a replay buffer D, or:

$$\min_\phi \mathbb{E}_{(c_t, w^{train}, w^{val}, \mathcal{G}_t^{val}, \mathcal{G}_{t+k}^{val}) \sim D}[(\mathcal{G}_{t+k}^{val} - \mathcal{G}_t^{val} - g_\phi(c_t, w^{train}, w^{val}))^2]$$

In addition, since the evaluated return $\mathcal{G}_t^{val}$ is available in each data point, we use the the same parameter ϕ with an additional head of two fully-connected layers to perform an auxiliary task that predicts $\mathcal{G}_t^{val}$. The final loss function is the linear combination of the two losses, which can be written as follows:

$$\min_\phi \mathbb{E}_{(c_t, w^{train}, w^{val}, \mathcal{G}_t^{val}, \mathcal{G}_{t+k}^{val}) \sim D}[(\mathcal{G}_{t+k}^{val} - \mathcal{G}_t^{val} - g_\phi(c_t, w^{train}, w^{val}))^2 + \alpha * (\mathcal{G}_t^{val} - h_\phi(c_t, w^{train}, w^{val}))^2],$$

where α is a constant coefficient, and h_ϕ represents the function that predicts the evaluated return $\mathcal{G}_t^{val}$. We found that such an auxiliary task can reduce the over-fitting when the model is tested in new lifetimes.

In the experiments for both the **FourRoom** and the **Habitat** domains, we use the same neural network architecture to represent the learning dynamics model, which we detail as follows. The contextual information $c_t = \langle (w_j, \mathcal{G}_j^{train}) \rangle_{j=1}^{J}$ is a sequence of task-return pairs. The model first feeds each task-return pair into a 16-dimensional linear embedding layer. Then, the output of the sequence of embeddings is fed into two 64-hidden-size layers of LSTMs. The resulting 64-dimensional contextual embedding is concatenated with two 16-dimensional embeddings that embeds w^{train} and w^{val} respectively. Lastly, the 96-dimensional feature embedding is fed into two fully-connected layers with 64 and 32 hidden units, which outputs the learning progress prediction. In addition, another two 64×32 layers are created to predict the evaluated return. We used the constant coefficient $\alpha = 0.1$, which is chosen from grid search that minimizes the loss on hold-out validation data points. The model is optimized by Adam optimizer (Kingma and Ba 2014) with an learning rate of 0.0001 and a batch size of 512. We used a function `UpdateModel`$(D, \phi,$ *NumEpoch*$)$ that will learn the model on replay buffer D for *NumEpoch* epochs with the aforementioned hyper-parameters.

A.2 Meta-Training Pipeline

We provide the pseudo-code for the meta-training pipeline of the `MM-ACL` in Algorithm 1.

A.3 Hyper-parameters

The hyper-parameters that specify the learning dynamics model are described in Appendix A.1. In the table below, we list the hyper-parameters that specify the meta-training process.

	FourRoom	**Habitat**
NumLifetime	150	120
NumIteration	300	500
NumValidationTask	4	4
Horizon k	1	2
UpdateInterval	45000	20000
NumEpoch	100	100
ExplorationProbability	1	[0.9, 0.5, 0.1]

Note that in the **FourRoom** domain, *UpdateInterval* is set to 450000 that only updates the learning dynamics model once with the data collected purely from random curriculum. This works for the **FourRoom** domain with relatively simple controllable parameter space and can speed up the training by parallelizing the training of multiple lifetimes. In the **Habitat** domain, the *ExplorationProbability* is decreased by 0.4 at every 40 lifetimes when the model is updated for 100 epochs.

Algorithm 1: `MM-ACL`

Input: Horizon k, RL algorithm *Train*, meta-CL domain $\mathcal{M} = \mathcal{W} \times \mathcal{V}$, target task distribution $p(m)$.
Output: ϕ
Initialize $D \leftarrow \{\}$, ϕ
for $l = 1$ **to** *NumLifetime* **do**
  Initialize ψ_0, $c_0 \leftarrow \emptyset$, $(w_{target}, v_{target}) \sim p(m)$, $t = 0$.
  while $t <$ *NumIteration* **do**
    $w^{train} \leftarrow \texttt{SelectTask}(c_t, w_{target}, \phi)$
    for $j = 0$ **to** k **do**
      $(\psi_{t+j+1}, \tau_{t+j}) \leftarrow \textit{Train}(\psi_{t+j}, w^{train}, v_{target})$
      $\mathcal{G}^{train}_{t+j} \leftarrow \mathcal{G}(\tau_{t+j})$
      $c_{t+j+1} \leftarrow c_{t+j} \cup (w^{train}, \mathcal{G}^{train}_{t+j})$
    end for
    for $j = 1$ **to** *NumValidationTask* **do**
      $w^{val} \leftarrow \texttt{GetRandomTask}(\mathcal{W})$
      $\mathcal{G}^{val}_{t+k} \leftarrow \mathcal{G}(\psi_{t+k}, w^{val}, v_{target})$
      $\mathcal{G}^{val}_{t} \leftarrow \mathcal{G}(\psi_{t}, w^{val}, v_{target})$
      $D \leftarrow D \cup (c_t, w^{train}, w^{val}, \mathcal{G}^{val}_{t+k} - \mathcal{G}^{val}_{t})$
    end for
    if $(t + l * \textit{NumIteration})$ mod *UpdateInterval* $== 0$ **then**
      $\phi \leftarrow \texttt{UpdateModel}(D, \phi, \textit{NumEpoch})$
    end if
    $t \leftarrow t + k$
  end while
end for

Algorithm 2: `SelectTask`

Input: Contextual information c, w_{target}, *learning dynamics model* ϕ
Output: Selected task w
Initialize: $p \leftarrow \text{Uniform}(0, 1)$
if $p <$ *ExplorationProbability* **then**
  $w \leftarrow \texttt{GetRandomTask}(\mathcal{W})$
else
  Sample 40 tasks uniformly and randomly, call it $\mathcal{W}^\dagger$.
  $\forall i \in \mathcal{W}^\dagger, \;\; P[i] \leftarrow \text{Softmax}\big(g^k_\phi(c_t, w, w_{target})\big)[i]$.
  Sample $w \sim P$.
end if

B Baselines

B.1 Student (PPO)

We used a customized PPO implementation (Schulman et al. 2017) for both **FourRoom** domain and **Habitat** domain. We detail the neural network architectures and hyper-parameters as follows.

For **FourRoom** domain, the observation is composed of a three-channel seven-by-seven image with the channels encode color, object type and object status respectively. The image is fed into a CNN neural network of sequential layers: **Input** $\rightarrow$ **Conv2D[16, 2, 1]** $\rightarrow$ **MaxPool[2]** $\rightarrow$ **Conv2D[32, 2, 1]** $\rightarrow$ **Conv2D[64, 1, 1]** $\rightarrow$ **LSTM[128]** $\rightarrow$ **FC[128]**, where **FC** is the fully connected layers with the number denoting the number of hidden units, **LSTM** is one layer of LSTM with the number denoting hidden size, and **Conv2D** is the 2D convolution neural network with the numbers denoting the number of channels, kernel size, and stride respectively. All the layers are followed by the ReLU activation. This design of the architecture is modified from an RL implementation (Lucas Willems 2013) that is known to solve the MiniGrid environment.

For **Habitat** domain, the observation is composed of a four-channel 84-by-84 RGB-D image. The image is fed into a CNN neural network of sequential layers: **Input** $\rightarrow$ **Conv2D[32, 8, 4]** $\rightarrow$ **Conv2D[64, 4, 2]** $\rightarrow$ **Conv2D[64, 2, 2]** $\rightarrow$ **FC[128]** $\rightarrow$ **FC[128]**, where **FC** is the fully connected layers with the number denoting the number of hidden units, and **Conv2D** is the 2D convolution neural network with the numbers denoting the number of channels, kernel size, and stride respectively. All the layers are followed by the ReLU activation.

Other PPO related hyper-parameters are listed as follows:

PPO	**FourRoom**	**Habitat**
Discount factor	0.9	0.9
Ratio clip	0.2	0.2
Entropy coefficient	0.01	0.01
GAE (Schulman et al. 2015) coefficient	0.97	0.97
Batch size	256	256
Learning rate	0.0001	0.0001
Number of epochs per update	4	4
Number of steps between updates	2000	8000

B.2 `ALP-GMM`

The core idea of `ALP-GMM` is to fit a Gaussian Mixture Model (GMM) every n episodes on recently sampled tasks *concatenated with their respective absolute learning progress (ALP) value*. The ALP is defined as the return difference of the current sampled task and the same task sampled in the history. Since it is impossible to sample exactly the same task twice for continuous task space, `ALP-GMM` computes a per-task ALP from the entire history of sampled tasks using a knn-based approach. The Gaussian from which to sample a new task is then chosen proportionally to its respective learning progress. The original `ALP-GMM` is applied to the **Habitat** domain with the hyper-parameters listed as follows:

`ALP-GMM` (continuous)	**Habitat**
Fit rate	100
Buffer size	200
Random task ratio	0.2
Number of bootstrap episodes	50
Number of mixture models	[2, 3, 4, 5]

For the **FourRoom** domain with discrete task space, instead of fitting a GMM, `ALP-GMM` maintains one replay buffer of the returns for each of the tasks. The ALP is measured by the return difference between the current task and the last sampled task in the same reply buffer. Then, the tasks are sampled proportional to the their average ALPs in the replay buffers. The related hyper-parameters are as follows:

`ALP-GMM` (discrete)	**FourRoom**
Buffer size	50
Random task ratio	0.2

B.3 `DQN-CMDP`

`DQN-CMDP` models the curriculum learning problem as Markov Decision Process (MDP) and solves the MDP with a standard DQN algorithm. More specifically, the action space of such an MDP is the uncontrollable task space $\mathcal{W}$, and the state space is the space of contextual information $\mathcal{C}$. An episode of this MDP is an entire lifetime that terminates when the evaluated return reaches the threshold (the same criteria used to compute the *iterations to threshold*). The teacher receives a +10 reward only when the evaluated return reaches the threshold.

We only applied `DQN-CMDP` to the **FourRoom** domain that uses discrete task space. We used the same encoder as described in Appendix A.1 to encode the contextual information. Then, the extracted feature embedding is fed into a 64×32 fully-connected layers to output a value prediction. Such a DQN policy is trained for the same number of 150 lifetimes (episodes) for comparison.

B.4 Handcrafted Curriculum

The handcrafted curricula are empirically created based on their performances on the target tasks that are different from the five hold-out target tasks used for the meta-testing. For the **FourRoom** domain, the handcrafted curriculum trains a student on controllable parameter $w = 2$, $w = 3$, $w = 1$, and $w = 0$ for 50k, 50k, 50k, and 100k time steps respectively. For the **Habitat** domain, the handcrafted curriculum trains a student on controllable parameter $w = 4.5$, $w = 5.75$, $w = 7$, $w = 8.25$, $w = 9.5$, $w = 10.75$, and $w = 12$ with each trained for 200k steps, and finally the target task trained until the end of the lifetime.