Model-Based Meta Automatic Curriculum Learning

Anonymous Authors[1]

Abstract

When an agent trains for one target task, its experience is expected to be useful for training on another target task. This paper formulates the meta curriculum learning problem that builds a sequence of intermediate training tasks, called a curriculum, which will assist the learner to train toward any given target task in general . We propose a model-based meta automatic curriculum learning algorithm (MM−ACL) that learns to predict the performance on one task when trained on another, given contextual information such as the history of training tasks, loss functions, rollout state-action trajectories from the policy, etc. This predictor facilitates the generation of a curriculum that optimizes the performance of the learner on different target tasks. Our empirical results demonstrate that MM−ACL outperforms a random curriculum, a manually created curriculum, and a commonly used non-stationary bandit algorithm in a GRIDWORLD domain.

1. Introduction

Curriculum Learning (CL) is the problem of generating a sequence of *source tasks* for a reinforcement learning agent (learner) to train on, in order to improve some objective related to the learner's performance on a *target task*. To address this problem, automatic curriculum learning (ACL) has demonstrated its success by improving sample efficiency (Schaul et al., 2015), dealing with sparse reward (Zheng et al., 2018; Durugkar et al., 2021), facilitating generalization (Akkaya et al., 2019), and finding solutions to hard-exploration challenges (Ecoffet et al., 2019) . The curriculum is realized as a sequence of intermediate tasks such as the next environment (Narvekar et al., 2017) or distribution of environments (Portelas et al., 2020a) to train on, or a sequence of learning contexts such as the discount factor, bootstrapping parameter (Xu et al., 2018), and intrinsic rewards (Zheng et al., 2018). In an attempt to find a good curriculum, most existing work optimizes surrogate objectives such as learning progress, diversity or intermediate difficulty (Portelas et al., 2020b). However, this approach may lead to sub-optimal solutions or failures when the surrogate objective does not align well with the intrinsic learning objective. Rather than solving for surrogate objectives, there is a line of ACL work that directly optimizes the learning time (Narvekar et al., 2017; Narvekar & Stone, 2018).

In this paper, we optimize a similar objective directly on a given target task, i.e., the performance on the target task, using the proposed MM−ACL algorithm. First, this algorithm learns from training different target tasks. Specifically, it learns to predict the performance on one task while training on another, based on the offline data collected from multiple lifetimes, where at each lifetime the learner trains toward a different target task. Based on this model, MM−ACL chooses the next intermediate task to train, such that the expected performance on the chosen target task is maximized. This process enables directly optimizing the performance on the target task, instead of focusing on the surrogate goals.

2. Problem description

Meta curriculum learning problem Consider a domain that includes a collection of possible source tasks $W \cup V$, where each task is characterized by controllable parameters $\mathbf{w} \in W$, representing its task control or learning contexts, and uncontrolled parameters $\mathbf{v} \in V$ that remain fixed when training a target task, but may differ between different target tasks. Since $\mathbf{v}$ is fixed in a curriculum, we refer to the task instantiated by the parameter set $\mathbf{w}$ as "task $\mathbf{w}$". The policy is parameterized by θ. To find the best policy for a target task $\mathbf{w}_{target}$, a straightforward approach is to use gradient-based optimization by training exclusively on the target task. However, in this approach the agent might not be able to learn a good policy due to reward sparsity or high variance of returns. Instead, we train the learner on a sequence of tasks defined as a curriculum $\mathcal{W} = (\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_N)$ generated by a teacher policy π_C. In particular, at each step t, given a policy θ_t, we rollout this policy on a task $\mathbf{w}_t \sim \pi_C$ multiple times, and use the data collected for a gradient update:

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}(\theta_t, \mathbf{w}_t, \mathbf{v}) \quad (1)$$

where $\mathcal{L}(\theta_t, \mathbf{w}_t, \mathbf{v})$ is the loss of policy θ_t on task $\mathbf{w}_t$.

Given the policy θ_N after N gradient updates following

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

curriculum $\mathcal{W}$, the learning progress can be characterized by the difference between the return of the initial policy θ_0 and the return of policy θ_N on the target task $\mathbf{w}_{target}$. Defining the return of a policy θ on task $\mathbf{w}$ as $\mathcal{G}(\theta, \mathbf{w}, \mathbf{v})$, a curriculum learning problem is to find the best curriculum that maximize the learning progress:

$$\arg\max_{\pi_C} \mathbb{E}_{\mathbf{w}_t \sim \pi_C}[\mathcal{G}(\theta_N, \mathbf{w}_{target}, \mathbf{v}) - \mathcal{G}(\theta_0, \mathbf{w}_{target}, \mathbf{v})]$$

A meta curriculum learning problem, instead, seeks to find a curriculum learning policy π_C that maximizes the learning progress over multiple lifetimes. We define a lifetime as a two-tuple $(\mathbf{w}_{target}, \mathbf{v})$ that differs from other lifetimes by its target task and uncontrolled parameters. Taking lifetimes into consideration, the objective of a meta curriculum learning problem becomes finding a curriculum policy π_C that maximizes the expectation of learning progress over a distribution of lifetime D:

$$\operatorname*{argmax}_{\pi_C} \mathbb{E}_{(\mathbf{w}_{target}, \mathbf{v}) \sim D, \mathbf{w}_t \sim \pi_C} [\mathcal{G}(\theta_N, \mathbf{w}_{target}, \mathbf{v}) - \mathcal{G}(\theta_0, \mathbf{w}_{target}, \mathbf{v})]$$

Optimizing local learning progress Due to the intractability of directly finding a sequence of N source tasks, we select the best task locally at each time step. To avoid insufficient training in each source task and coming back to the same source task frequently, we compute k gradient updates for each source task at each time step, $\forall j = 1, ..., k$:

$$\theta_{t+j} = \theta_{t+j-1} - \alpha \nabla_{\theta_{t+j-1}} \mathcal{L}^{train}(\theta_{t+j-1}, \mathbf{w}_t))$$

And the local objective is to choose the immediate task $\mathbf{w}_t$ to train next, in order to maximize the learning progress $\mathcal{P}_t^k(\mathbf{w}_t, \mathbf{w}_{target}, \mathbf{v})$ of this policy after k gradient updates, which is described as follows:

$$\mathcal{P}_t^k(\mathbf{w}_t, \mathbf{w}_{target}, \mathbf{v}) = \mathcal{G}^{val}(\theta_{t+k}, \mathbf{w}_{target}, \mathbf{v}) - \mathcal{G}^{val}(\theta_t, \mathbf{w}_{target}, \mathbf{v})$$

Here, we use superscript val to denote the gain/losses for validating the learning progress only, and $train$ to denote the gain/losses used for training and updating the weights.

3. MM-ACL

One common method to decide the task control parameters for the next iteration is to update the parameters by a meta gradient, or $\mathbf{w}_{t+k} = \mathbf{w}_t - \beta \nabla_{\mathbf{w}_t} \mathcal{P}_t^k(\mathbf{w}_t, \mathbf{w}_{target}, \mathbf{v})$ (Xu et al., 2018). However, in curriculum learning, the task control parameters are usually start-goal locations or deeply embedded in the transition dynamics. Therefore, the loss functions are usually non-differentiable with respect to these parameters unless the dynamics model of the environment is given. An alternative approach is to approximate the local learning progress $\mathcal{P}_t^k(\mathbf{w}_t, \mathbf{w}_{target}, \mathbf{v})$ with a black box function, and then choose the best task $\mathbf{w_t}$ based on the approximated learning progress. Instead of approximating the local learning progress on the target task, we learn a general function that captures the learning progress from $\mathbf{w}$ to any other task $\mathbf{w}'$. Formally, we define a *learning dynamics model* that models local learning progress as function $f^k(c, \mathbf{w}, \mathbf{w}', \mathbf{v})$, where c is the context, $\mathbf{w}$ is the task the agent is trained on, $\mathbf{w}'$ is the task the agent is evaluated on, and $\mathbf{v}$ is the set of uncontrolled parameters. There are several ways to represent the context, such as the following.

1) **No context**: if no context is provided, the *learning dynamics model* is simply represented as $f^k(\mathbf{w}, \mathbf{w}', \mathbf{v})$, which is equivalent to modeling the curriculum learning as a non-stationary bandit problem . Under this representation, a function $R(\mathbf{w}) := f^k(\mathbf{w}, \mathbf{w}_{target}, \mathbf{v})$ can be thought of as the reward function for choosing a bandit arm $\mathbf{w}$ for only one lifetime with fixed $(\mathbf{w}_{target}, \mathbf{v})$ (Matiisen et al., 2019). Such a modeling enables learning f^k adaptively in a single lifetime, however in this case f^k be applied to decide tasks under different context or in different lifetimes.

2) **Training tasks and losses**: the context is represented as $c_t = (\mathbf{w}_0, \mathcal{L}_0^{train}, \mathbf{w}_1, \mathcal{L}_1^{train}, ..., \mathbf{w}_{t-1}, \mathcal{L}_{t-1}^{train})$, a sequence of all the history of training tasks and associated losses. Such a context is lightweight and does not add additional computation overhead (e.g., more rollouts or gradient updates).

3) **Validation trajectories**: the context is represented as $c_t = \{\tau_t^j\}_{j=1}^{j=J}$, a sequence of state-action pairs generated from evaluating the current policy θ_t on the task $\mathbf{w}_{target}$. Trajectories carry more information, but require much more training data to learn to efficiently encode the state-action pairs (Fang et al., 2020).

For the rest of this paper, we focus on the context of **training tasks and losses**, with which the *learning dynamics model* can be trained with a relatively small amount of data, while still be applied for different lifetimes. In addition, in practice, uncontrolled parameters are usually difficult to be encoded and are usually hidden to the teacher. Therefore, we consider only the *learning dynamics model* of form $f_\phi^k(c, \mathbf{w}, \mathbf{w}')$ that does not take $\mathbf{v}$ as an input. Once a *learning dynamics model* is learned, the curriculum learning policy can be created by greedily selecting the task that maximizes the local learning progress, or $\pi_C = \arg\max_w [f_\phi^k(c, \mathbf{w}, \mathbf{w}_{target})]$.

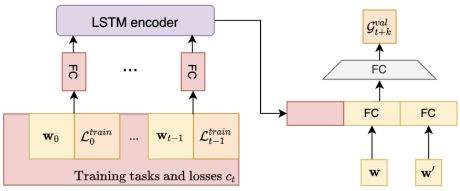

Figure 1. Diagram of *learning dynamics model*.

A *learning dynamics model* $f_\phi^k(c, \mathbf{w}, \mathbf{w}')$ parameterized by ϕ is a regression model that takes an input of three-tuple

Algorithm 1 MM-ACL

1: **Input:** Horizon k, task loss $\mathcal{L}(\theta, \mathbf{w})$, target task $\mathbf{w}_{target}$, lifetime distribution D.
2: **Output:** ϕ
3: Initialize $\mathcal{D} = \{\}$, ϕ
4: **for** $l = 1$ **to** $NumLifetime$ **do**
5: Initialize θ_0, $c_0 = \emptyset$, $(\mathbf{w}, \mathbf{v}) \sim D$.
6: **while** $t < NumIteration$ **do**
7: $\mathbf{w}^{train} \leftarrow$ SelectTask$(c_t, \mathbf{w}_{target}, \phi)$
8: **for** $j = 0$ **to** k **do**
9: $\mathcal{L}^{train}_{t+j} \leftarrow \mathcal{L}(\theta_{t+j}, \mathbf{w}^{train}, \mathbf{v})$
10: $\theta_{t+j+1} = \theta_{t+j} - \alpha \nabla_{\theta_{t+j}} \mathcal{L}^{train}_{t+j}$
11: $c_{t+j+1} \leftarrow c_{t+j} \cup (\mathbf{w}^{train}, \mathcal{L}^{train}_{t+j})$
12: **end for**
13: **for** $j = 1$ **to** $NumValidationTask$ **do**
14: $\mathbf{w}^{val} \leftarrow$ GetRandomTask()
15: $\mathcal{L}^{val}_{t+k} \leftarrow \mathcal{L}(\mathbf{w}^{val}, \theta_{t+k}, \mathbf{v})$
16: $\mathcal{L}^{val}_{t} \leftarrow \mathcal{L}(\mathbf{w}^{val}, \theta_{t}, \mathbf{v})$
17: $\mathcal{D} \leftarrow \mathcal{D} \cup (c_t, \mathbf{w}^{train}, \mathbf{w}^{val}, \mathcal{L}^{val}_{t+k} - \mathcal{L}^{val}_{t})$
18: **end for**
19: **if** $t + 1 \bmod UpdateInterval == 0$ **then**
20: $\phi \leftarrow$ UpdateModel$(\mathcal{D}, \phi)$
21: **end if**
22: $t \leftarrow t + k$
23: **end while**
24: **end for**

Algorithm 2 SelectTask

1: **Input:** Context c, target task $\mathbf{w}_{target}$, *learning dynamics model* ϕ
2: **Output:** Selected task $\mathbf{w}$
3: **Initialize:** $p \leftarrow Uniform(0, 1)$
4: **if** $p < ExplorationProbability$ **then**
5: $\mathbf{w} \leftarrow$ ALP-GMM(c)
6: **else**
7: $\mathbf{w} \leftarrow \arg\max_w [f^k_\phi(c_t, \mathbf{w}, \mathbf{w}_{target})]$
8: **end if**

$(c, \mathbf{w}, \mathbf{w}')$ and returns a prediction of learning progress on task $\mathbf{w}'$ after k iterations of reinforcement learning update. The architecture of the *learning dynamics model* is shown in Fig. 1. The history training task-loss pairs are embedded by one fully-connected layer and are fed into a LSTM (Gers et al., 2000) as a sequence of task-loss embedding. Then, the output embedding of the LSTM is concatenated with the embedding of $\mathbf{w}$ and $\mathbf{w}'$ followed by a MLP to compute the validation return prediction $\mathcal{G}^{val}_{t+k}$.

Algorithm 1 shows the whole pipeline of training f^k_ϕ. The training data is stored in a replay buffer $\mathcal{D}$ collected from learners trained from scratch in multiple i.i.d lifetimes. At an iteration t of a lifetime, a training task $\mathbf{w}^{train}$ is selected based on a SelectTask function (Algorithm 2) that has a exploration probability p to select the task based on the current *learning dynamics model* and a probability $1 - p$ to select a exploratory task. For limited discrete task space, the exploratory tasks are uniformly sampled from parameter space $\mathcal{W}$. For more complex continuous task space, the exploratory tasks are selected by ALP-GMM (Portelas et al., 2020a), an ACL algorithm that has been reported to efficiently solve more tasks compared to randomly selected curriculum in a continuous task space during the training of one single lifetime (Portelas et al., 2020a). After a training task $\mathbf{w}^{train}$ is selected, the parameters of the agent are updated for k iterations. During the updates, the history of tasks and losses are appended to the context variable (see line 12 in Algorithm 1). Then, the losses $\mathcal{L}^{val}_t$ and $\mathcal{L}^{val}_{t+k}$ are evaluated on the same randomly sampled validation task $\mathbf{w}^{val}$ before and after the updating respectively. The learning progress, context, training and validation tasks form a new instance of training data $(c_t, \mathbf{w}^{train}, \mathbf{w}^{val}, \mathcal{L}^{val}_{t+k} - \mathcal{L}^{val}_t)$ which is added into a replay buffer $\mathcal{D}$. At every fixed interval of iterations, the *learning dynamics model* parameter ϕ is updated based on the training data in the replay buffer $\mathcal{D}$. Once the *learning dynamics model* is learned, the model can be deployed online following the SelectTask function with exploration probability $p = 0$. During deployment, the conext variable c_t is built in the same way as line 12 in Algorithm 1. Notice that during deployment, lines 15 to 22 in Algorithm 1 are not executed, which guarantees that the algorithm does not add extra rollouts and gradient updates.

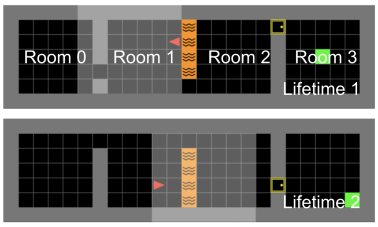

Figure 2. Examples of four-room environments from two different lifetimes. Each lifetime has a unique room configuration.

4. Experimental Results

We test MM-ACL in the FourRoom domain based on the widely used MiniGrid environment by Chevalier-Boisvert et al. (2018). As shown in Figure 2, the target task in this domain requires an agent to navigate through three rooms (labeled as room 0, 1, and 2 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 with wave patterns) 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 controllable parameter space in this domain is a four-category discrete set $W = \{0, 1, 2, 3\}$ with $\mathbf{w}$ 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.

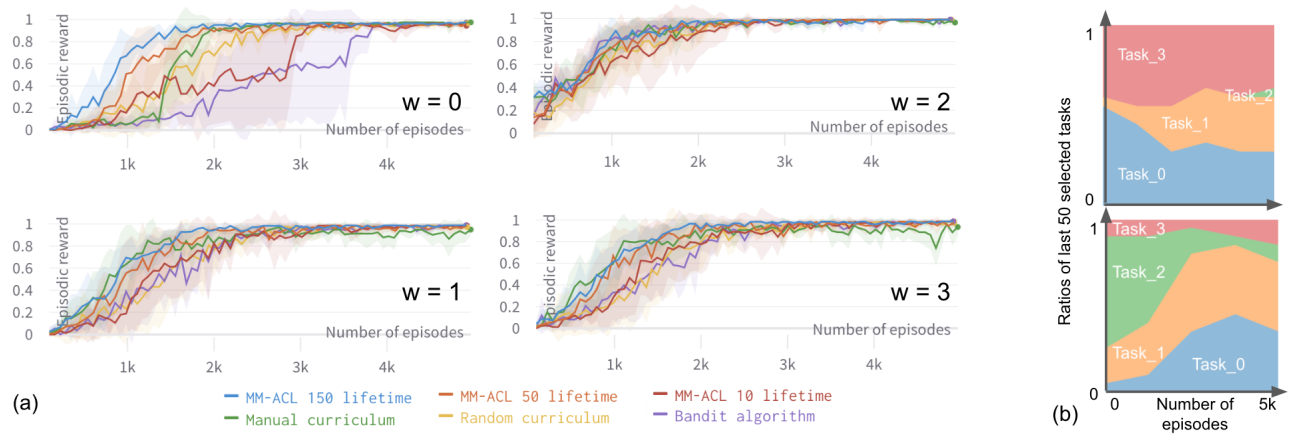

Figure 3. (a). The episodic reward of the agents evaluated on tasks of $\mathbf{w} = 0, 1, 2, 3$ during the training with curriculum from different methods; (b). Ratios of last 50 selected tasks by MM-ACL (top) and bandit algorithm (bottom) at the training episodes from 0 to 5000.

The uncontrollable parameter space V is a set of all possible room configurations. Each configuration can have different locations of the openings and doors (see two example room configurations in Fig. 2). Directly learning $\mathbf{w}_{target}$ ($\mathbf{w} = 0$) is difficult because of the long navigation trajectory from room 0 to room 3 and the hard exploration due to the lava tiles that cause immediate failure. We assume that the agent will experience multiple lifetimes, such that in each lifetime, the agent will be faced with a room configuration uniformly sampled from V with its weight reinitialized.

In the experiment, the agent is trained using PPO (Schulman et al., 2017) with its hyper-parameters kept similar to the example reinforcement learning algorithm implementation (Chevalier-Boisvert et al., 2018). This set of hyper-parameters is not guaranteed to be optimal for FOURROOM, but is good enough to learn the target task if a good curriculum is given. The model is trained for 150 lifetimes. After the training, the *learning dynamics model* is deployed for 5 hold-out lifetimes that train learners from scratch with tasks selected by the *learning dynamics model*. During the deployment, the agents' performances are evaluated on all the four tasks for 8 episodes at every 4 iterations. To investigate the dependence of performance on the amount of training data, we deployed the *learning dynamics model* trained with only 10 and 50 lifetimes. The results are compared with a random curriculum, a non-stationary-bandit ACL algorithm similar to (Matiisen et al., 2019), and a manually designed curriculum that trains on tasks $\mathbf{w} = \mathbf{3}$, $\mathbf{w} = \mathbf{1}$, and $\mathbf{w}_{target}$ for 500, 800, and 4000 episodes respectively. The training curves are shown in Fig. 3(a), and the ratios of the four tasks being selected by MM-ACL and bandit algorithm for every 50 selected tasks are shown in Fig. 3(b).

As shown in Fig. 3(a), given sufficient training data of 150 lifetimes, MM-ACL (red) can train the agent with the best sample efficiency that learns the target task with about 1.8k episodes compared to 2.8k by random curriculum and 4k by the bandit algorithm. The sample efficiencies on tasks $\mathbf{w} = 1$ and $\mathbf{w} = 3$ are also marginally improved, which take about 400 less episodes to converge. The MM-ACLs trained by 150, 50, and 10 lifetimes are shown by blue, orange and red curves in Fig. 3(a). The sample efficiency monotonously decreases with the decreasing number of training lifetimes with MM-ACL trained by 10 lifetimes performing even worse than the random curriculum.

Surprisingly, the curriculum generated by the bandit algorithm does not benefit the training, because the bandit algorithm exploits the easiest task of $\mathbf{w} = 2$, which actually disrupts the training of harder tasks. By carefully analyzing the tasks in this domain, we find that the skill of avoiding lava trained by task $\mathbf{w} = 3$ is the key to learning more difficult task, since directly training on $\mathbf{w} = 0$ and $\mathbf{w} = 1$ can easily step into the lava and cause immediate failures that make the training very difficult. Instead, the curriculum generated by MM-ACL focuses evenly on task $\mathbf{w} = 3$ and $\mathbf{w} = 0$ that develop the skill of avoiding the lava and trying the target task with the developed skill. The MM-ACL ignores the task $\mathbf{w} = 2$ completely as it does not contribute to the learning of target task.

5. Conclusion

This paper first formulates a new meta curriculum learning problem, in which a teacher is required to generalize its curriculum policy, learned from multiple lifetimes, so it can train the agents toward different target tasks. A new meta curriculum learning algorithm MM-ACL is proposed that leverages the idea of cross validations and selects the tasks that directly optimize the target task performance. Our empirical results show an improvement of sample efficiency of the training process, tested in multiple lifetimes.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Durugkar, I., Tec, M., Niekum, S., and Stone, P. Adversarial intrinsic motivation for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Fang, K., Zhu, Y., Savarese, S., and Fei-Fei, L. Adaptive procedural task generation for hard-exploration problems. *arXiv preprint arXiv:2007.00350*, 2020.

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

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

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

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

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

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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

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

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