CoMPS: Continual Meta Policy Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

We develop a new continual meta-learning method to address challenges in sequential multi-task learning. In this setting, the agent's goal is to achieve high reward over any sequence of tasks quickly. Prior meta-reinforcement learning algorithms have demonstrated promising results in accelerating the acquisition of new tasks. However, they require access to all tasks during training. Beyond simply transferring past experience to new tasks, our goal is to devise continual reinforcement learning algorithms that learn to learn, using their experience on previous tasks to learn new tasks more quickly. We introduce a new method, continual meta-policy search (CoMPS), that removes this limitation by meta-training in an incremental fashion, over each task in a sequence, without revisiting prior tasks. CoMPS continuously repeats two subroutines: learning a new task using RL and using the experience from RL to perform completely offline meta-learning to prepare for subsequent task learning. We find that CoMPS outperforms prior continual learning and off-policy meta-reinforcement methods on several sequences of challenging continuous control tasks.

1 Introduction

Meta-reinforcement learning algorithms aim to address the sample complexity challenge of conventional reinforcement learning (RL) methods by *learning to learn* – utilizing the experience of solving prior tasks in order to solve new tasks more quickly. Such methods can be exceptionally powerful, learning to solve tasks that are structurally similar to the meta-training tasks with just a few dozen trials [14, 10, 56, 63]. However, prior work on meta-reinforcement learning is generally concerned with asymptotic meta-learning performance, or how well the meta-trained policy can adapt to a single new task at the end of a long meta-training period. The meta-training process itself requires iteratively attempting each meta-training task in a "round-robin" fashion. While this is reasonable in supervised settings, in reinforcement learning revisiting and repeatedly interacting with previously seen tasks in the real world may be difficult or impossible. For example, when learning to tidy in different homes – effective generalization requires visiting many homes and interacting with many items, but needing to revisit every prior home and item on each iteration of meta-training would be impractical. Instead, we would want the robot to use each new experience in each new home to *incrementally* augment its skillset so that it can acquire new cleaning skills in new homes more quickly, as shown in Figure 1 using examples from MetaWorld [60]. In this paper, we study the continual meta-reinforcement learning setting, where tasks are experienced one at a time, without the option to collect additional data on previous tasks. The objective is to decrease the time it takes to learn each successive task, as well as achieve high asymptotic performance.

This paper proposes a novel meta-learning algorithm for tackling the continual multi-task learning problem in a reinforcement learning setting. The key desiderata for such a method are the following.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

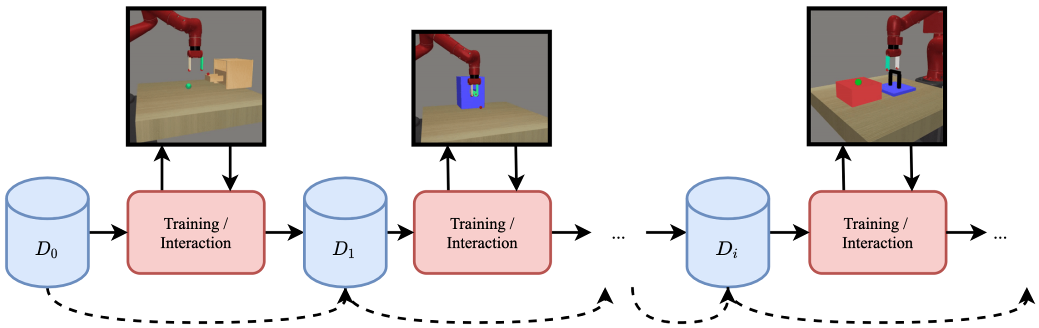


Figure 1: In the continual meta-RL setting, the agent interacts with a single task at a time and, once finished with a task, never interacts with it again. An agent who can efficiently learn should reuses experience from previous tasks to more quickly adapt to the subsequent tasks more quickly.

First, an effective continual meta-learning algorithm should adapt quickly to new tasks that resemble previously seen tasks, and at the same time still adapt (even if slowly) to completely novel tasks. This adaptation is crucial in the early stages of meta-training when the number of tasks seen so far is small, and every new task appears new and different. Second, an effective continual meta-learning algorithm should be able to use all previously seen tasks to improve its ability to adapt to future tasks, integrating information even from tasks seen much earlier in training, and are therefore far off-policy. To address the first requirement, we base our approach on model-agnostic meta-learning (MAML) [14]. MAML adapts to new tasks via gradient descent, while the meta-training process optimizes the initialization for this gradient descent process to enable the fastest possible adaptation. Because the adaptation process in MAML corresponds to a well-defined learning algorithm, even new out-of-distribution tasks are learned (albeit slowly), while tasks that resemble those seen previously will be learned much more quickly [13]. To address the second requirement, and make it possible to incorporate data from older tasks without revisiting them, we devise a method where the adaptation process corresponds to on-policy policy gradient. This meta-training uses behavioral cloning on successful episodes experienced by the agent from older tasks. Although the "inner loop" policy gradient adaptation process is on-policy, and the agent adapts to each new task with on-policy experience, the meta-training process, which is similar to distillation of previously collected experience, is off-policy. Essentially, the agent meta-trains the model such that a few steps of policy gradient result in a policy that mimics the most successful episodes on each previously seen task. This can effectively enable our approach to incorporate experience from much older policies and tasks into the meta-training process. Although our algorithm is intended to operate in settings where new tasks are revealed sequentially, one at a time, as in the home cleaning robot example before, it still relies on storing all experience – therefore, we do not address the forgetting challenges explored in prior work on continual learning [16], and instead focus on how sequential meta-learning can accelerate incremental task acquisition.

Our primary contribution is a meta-reinforcement learning algorithm that supports the sequential multi-task learning setting, where the agent cannot revisit previous tasks to collect data. To evaluate our approach, we modify a collection of commonly used meta-RL benchmarks into continual multi-task problems, with tasks presented one at a time. Our method outperforms other methods, achieving a higher average reward with fewer samples on average over each of the tasks in the sequence. In addition, we evaluate each method's ability to generalize over a collection of held-out tasks during training. We find that CoMPS achieves a higher meta-test time performance on held-out tasks. Lastly, we show that as the agent experiences more tasks, learning time on new tasks decreases, indicating that meta-learning performance increases asymptotically with the number of tasks.

2 Related Work

Meta-learning, or *learning-to-learn* [44, 4, 54], is concerned with the problem of learning a prior, given a set of tasks, that enables more efficient learning in the future. We focus on meta-learning for reinforcement learning [44, 10, 56, 14, 31]. There are many ways to represent the meta-learned model, including black box models [10, 56, 31, 49, 37, 63, 43, 12, 50, 9], applying gradient descent from initial parameters [14, 19, 40, 8, 30, 62, 32], and training a critic to provide policy gradients [51, 24, 58, 5]. Recent work has made progress toward using supervised meta-learning in an online setting [38, 15, 27, 26, 18, 61, 3, 55, 59, 57]. Adaptation without task boundaries or inside of an RL episode has also become a new area of investigation for meta-learning [33, 25, 21, 1, 23, 21]. Our work focuses on a distinct problem setting, where for meta-training the RL tasks are experienced sequentially, and the goal is to learn each RL task more quickly by leveraging the experience from the prior tasks, without the need to revisit prior tasks.

Continual or online learning studies the streaming data setting, where experience is used for training as soon as it is received [54, 22, 7, 35]. Both of these terms are often used to describe the process of learning tasks in sequence while avoiding the problem of *forgetting*, which refers to negative transfer to prior tasks [16, 41, 36, 42, 6, 53, 2]. Following Rolnick et al. [39], we do not aim to address the problem of forgetting, and instead retain data from prior tasks in a replay buffer to use for training a model that can adapt quickly to new tasks. Other recent methods support continual learning without ground truth task boundaries [45, 48, 34], but have not yet been demonstrated to perform well in reinforcement learning settings. Even with replay buffers, the data from previous tasks is still challenging to reuse since it was collected by a different policy. In Section 4 we describe our approach that explicitly optimizes for efficiently learning new tasks, which we find effective for accelerating the lifelong reinforcement learning process using stored off-policy data.

Off-policy RL methods are known to achieve good sample efficiency by reusing prior data. Therefore, several recently proposed sample-efficient meta-RL algorithms have been formulated as off-policy methods [37, 12, 33, 43]. In principle, these methods could be extended to the continual meta-learning setting. However, in practice, their ability to utilize data from past tasks collected under an older policy is limited, and we find, via our experiments, that they tend to perform poorly in the continual meta-reinforcement learning setting, possibly due to their limited ability to extrapolate to the new out of distribution tasks.

To meta-train without revisiting prior tasks, our method uses a type of self-imitation or distillation procedure. Although this resembles imitation learning, it does not use external demonstrations: meta-self-imitation learning uses high reward experience the agent itself collected in prior tasks. There are a number of non-meta-learning based methods that used behavioral cloning, distillation, and self-imitation for re-integrating previous experience [41, 36, 29, 52, 17]. Our work uses meta-self-imitation learning as the outer objective. As such not only trains a model to imitate a previous policy but also trains this policy to adapt given little data quickly. This training is similar to GMPS [30]; however, CoMPS does not use on-policy data and must generate its own high reward data that it can use for meta-self-imitation-learning. Instead, comps itself must generate its own high reward data that it can use for meta-self-imitation learning. Section 4 describes how we construct an off-policy meta-RL algorithm that collects its own data for meta-imitation.

3 Preliminaries

Reinforcement learning framework. RL problems are generally formalized as a *Markov decision process* (MDP), defined by the tuple MDP $= (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \rho, \gamma, T)$, with the state space $s \in \mathcal{S}$, the action space $a \in \mathcal{A}$, a transition probability function $\mathcal{P}(s'|s,a)$, a reward function $R(s,a)$, an initial state distribution $\rho(s_0)$, a discount factor $\gamma \in (0,1]$, and a time horizon T. The agent's actions are defined by a policy $\pi(a|s,\theta)$ parametrized by θ. The objective of the agent is to learn an optimal policy: $\theta^* := \text{argmax}_\theta J(\theta)$, where $J(\theta) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t), a_t\sim\pi(\cdot|s_t;\theta), s_0\sim\rho}\left[\gamma^t R(s_t,a_t)\right]$ is the expected discounted return.

Meta learning. Given a function $f(X;\theta)$ with parameters θ and a loss function ℓ, such as the mean squared error, and a set of samples $\mathcal{D}_i := (X_i, Y_i)$, we can write the task-loss as $\mathcal{L}(\mathcal{D}_i, \theta_i) = \mathbb{E}_{x_i, y_i \in \mathcal{D}_i}[\ell(f(\mathbf{x}_i;\theta_i), \mathbf{y}_i)]$. In the supervised setting, each task is specified with paired data of input $\mathbf{x}_i$ and output $\mathbf{y}_i$ samples. Meta-learning aims to leverage training across a set of meta-training tasks to enable fast adaptation on a different set of meta-test tasks not seen during training. MAML [14] accomplishes this by meta-training a set of initial parameters θ over the training tasks to efficiently adapt to a new task. We first summarize MAML for the supervised learning setting. A fixed distribution of tasks $p(\mathcal{T})$ is assumed. During meta-training a set of M tasks are drawn from this distribution $\{\mathcal{T}_i\}_{i=0}^M$. When the agent is deployed it experiences new tasks $\mathcal{T}_j \sim p(\mathcal{T})$ that provied a new set of data $\mathcal{D}_j := \{\mathbf{x}_j, \mathbf{y}_j\}$. Meta-learning trains the parameters for a model θ on $\{\mathcal{T}_i\}_{i=0}^M$ such that when the agent is deployed and recieved data from new tasks $\mathcal{D}_j$ the objective $f(X;\theta)$ is low after few gradient updates on the data $\mathcal{D}_j$. MAML minimizes the training loss:

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{train}), \mathcal{D}_i^{test}) \tag{1}$$

where for each tasks that data $\mathcal{D}_i$ is split into training $\mathcal{D}_i^{train}$ and testing $\mathcal{D}_i^{test}$ data. This objective essentially optimizes the initial parameters θ for few-shot generalization.

Meta reinforcement learning. Meta-learning in an RL setting requires extending this framework to MDPs. Each RL task $\mathcal{T}_i$ is a different MDP, with its own task objective J_i, defined as before. The state $\mathcal{S}$ and action space $\mathcal{A}$ are the same for these MDPs, however their transitions, rewards, and initial states can differ. To meta-train over these MDPs, the supervised losses in Eq. 1 are replaced with the expected discounted return. However, this meta-training process itself is sample inefficient (even though meta-test time adaptation is fast), requiring on-policy trajectories to estimate the inner policy gradient and many more trajectories for the outer objective. To reduce the cost of needing additional trajectories for the outer objective, Mendonca et al. [30] propose meta-training with the expected discounted return as the inner task loss and supervised imitation as the outer loss:

$$\min_{\theta} \sum_{\mathcal{T}_i} \sum_{\mathcal{T}_i^v \sim \mathcal{D}_{0:i}^*} \mathbb{E}_{\mathcal{T}_i}[\mathcal{L}_{BC}(\theta + \alpha \nabla_\theta J_i(\theta), \mathcal{D}_i^v)], \;\; \mathcal{L}_{BC}(\theta_i, \mathcal{D}_i) = - \sum_{(s_t, a_t) \in \mathcal{D}_i} \log \pi(a_t | s_t, \theta_i). \tag{2}$$

The outer behavioral cloning loss $\mathcal{L}_{BC}$ does not require collecting more data from the environment, but on-policy data from the environment is needed for computing the inner update on the policy parameters $\phi_i = \theta + \alpha \nabla_\theta J_i(\theta)$. This meta-RL method is more sample efficient when we have near-optimal data for the outer behavioral cloning loss $\mathcal{L}_{BC}$, but it cannot be trivially extended to a continual setting, the inner objective requires data to be repeatedly collected from each meta-training task. The following section will outline the continual multi-task learning problem and describe how we can extend GMPS to such a setting, removing the need to revisit prior tasks and carefully include a process for the agent to generate its own near-optimal data.

4 Continual Meta Policy Search

In the continual multi-task reinforcement learning setting, which we study in this paper, an agent proceeds through many tasks $\mathcal{T}_i$, one at a time. The agent's goal is to quickly solve each new task using a learning rule L, achieving high reward as efficiently as possible. In order to accomplish this, the agent can use all of its experience solving tasks $\mathcal{T}_{0:i}$ to learn how to adapt quickly to each new task $\mathcal{T}_{i+1}$, but cannot revisit past tasks to collect additional data once it moves on to the next task. This differs from the standard reinforcement learning setup, in which there is a single task $\mathcal{T}_0$, and the standard meta-RL setting, where all tasks can be revisited as many times as needed during meta-training. After each round of training on a new task, the agent produces a new set of policy parameters θ_i to serve as initialization for the next task. The learning rule L needs to serve two purposes: (1) solve the current task; (2) prepare the model parameters for efficiently solving future tasks. This process is depicted in Figure 2.

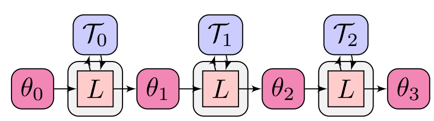


Figure 2: The continual multi-task reinforcement learning problem for a sequence of three tasks. The agent applies learning algorithm L on task $\mathcal{T}_i$ and forwards policy parameters θ_{i+1} after each task.

CoMPS overview. CoMPS addresses the continual multi-task RL problem one task at a time through a sequence of tasks $\mathcal{T}_i$. The L process for CoMPS consists of two main parts, a reinforcement learning RL process to learn the new task involving potentially hundreds of training steps and a meta-RL M process, which uses the experience from previous tasks to meta-train the initial parameters for RL. We illustrate the flow of these processes for CoMPS in Figure 3. The combination of RL and M creates a solution to the continuous multi-task RL problem that can perform non-trivial learning via meta-learning across tasks to accelerate learning. The RL step consists of using an on-policy RL algorithm to optimize a policy on $\mathcal{T}_i$ (described next), which benefits from the previous rounds of meta-training and is therefore very fast. This RL training process produces a dataset of trajectories $\mathcal{D}_i = \{\tau_0, \ldots, \tau_j\}$ where $\tau = \{(s_0, a_0, r_0), \ldots, (s_T, a_T, r_T)\}$. From this dataset, we set aside the experience that achieved the highest reward on the task as $\mathcal{D}_i^* \leftarrow \max_{\tau \in \mathcal{D}_i} \sum_{(a_t, s_t) \in \tau} R_i(s_t, a_t)$ called the skilled experience. The details of the RL step, the algorithm used, and the implementation are given in Section 4.1.

The meta-training in M uses the experience collected during RL in two separate data sets. The first dataset corresponds to the skilled experience $\mathcal{D}^*_{0:i}$, which is used in the outer meta-imitation learning objective. The second dataset consists of all the experience seen so far, $\mathcal{D}_{0:i}$, which is used for estimating the inner policy gradient based on Equation 2. Thus, instead of naïvely finetuning parameters on each new task, as in the case of standard continual RL methods, the M step in CoMPS produces meta-trained parameters that are optimized such that all prior tasks can be learned as quickly as possible starting from these parameters. When there are enough such tasks, these meta-trained parameters can provide forward transfer and generalize to enable fast adaptation to new tasks, enabling them to be acquired more efficiently than with naïve finetuning. However, to accomplish this, the M step must train from off-policy data from prior tasks. In Section 4.2, we describe how we implement this procedure.

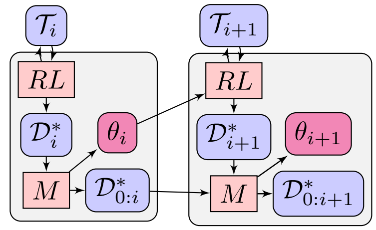


Figure 3: CoMPS is split in to two process. A reinforcement learning step RL and a meta-learning step M. The meta-learning step M uses the data gathered thus far, denoted $\mathcal{D}^*_{0:i}$, to meta-train θ_i. RL is initialized from these parameters θ_i for the next task.

4.1 Task Adaptation via Policy Gradient

To solve each task in the RL step, we use the popular policy gradient algorithm PPO [46]. PPO uses stochastic policy gradients to determine how to update the policy parameters θ compared to a recent version of the parameters that generated the current data θ', using a distribution ration $r_t(\theta) = \frac{\pi(a|s,\theta)}{\pi(a|s,\theta')}$. A first-order constraint, in the form of a gradient clipping term, is used to limit on-policy distribution shift of r_t while optimizing the policy parameters using $\mathcal{L}_{ppo}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_{\pi_{\theta'}}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\pi_{\theta'}})]$. The advantage $\hat{A}_{\pi_{\theta'}} = r_{t+1} + \gamma V_{\pi_{\theta'}}(s_{t+1}) - V_{\pi_{\theta'}}(s_t)$ is a score function that measures the improvement an action has over the expected policy performance $V_{\pi_{\theta'}}(s_t)$. CoMPS increases the sample efficiency of PPO in the RL step via initialization from the network parameters θ_i from the M step, which performs off-policy meta-RL, that we describe next. During the RL step, we collect 20 rollouts and perform 16 training updates with a batch size of 256. Additional details, including the learning parameters and network design, can be found in Appendix C.

4.2 Outer Loop Meta-Learning

The M process of CoMPS meta-trains a set of parameters θ_i using meta-self-imitation from the skilled experience. The outer self-imitation learning objective in Eq. 2 uses the skilled experience $\mathcal{D}^*_{0:i}$ to train the agent to be capable of (re)learning these skilled behaviors from one or a few policy gradient steps using previously logged off-policy experience, sampled randomly from $\mathcal{D}_{0:i}$. In contrast to methods that are concerned with forgetting, the parameters produced by this meta-RL training can quickly learn new behaviors that are similar to the high-value policies from previous tasks and, if enough prior tasks have been seen, likely generalize to quickly learn new tasks as well. The use of gradient-based meta-learning is particularly important here: as observed in prior work [13], gradient-based meta-learning methods are more effective at generalizing to new tasks under mild distributional shift as compared to contextual methods, making them well-suited for continual meta-learning with non-stationary task sequences, where new tasks can deviate from the distribution of tasks seen previously. In this case, gradient-based meta-learning methods degrade gracefully standard gradient-based optimization – in this case, policy gradient. However, in the continual setting, where prior tasks cannot be revisited, a significant challenge in this procedure is that the meta-RL optimization needs to estimate the policy gradient $\nabla_\theta J(\theta)$ in its inner loop for each previously seen task, without collecting additional data from the task (which it is not allowed to revisit). To address this challenge, we will utilize an importance-sampled update that we describe

Algorithm 1 CoMPS Meta-Learning

```
require: θ, skilled D*_{0:i} and off-policy D_{0:i}
for n ← 0...N do
  for j ← 0...i do
    D^{tr}_j ←sample m rollouts from D_j
    φ_j ← θ + α∇J_j(θ) (via imp. weights)
    Sample data D^{val}_j ∼ D*_j
    Update θ ← θ − β∇L_BC(φ_j, D^{val}_j)
  end for
end for
```

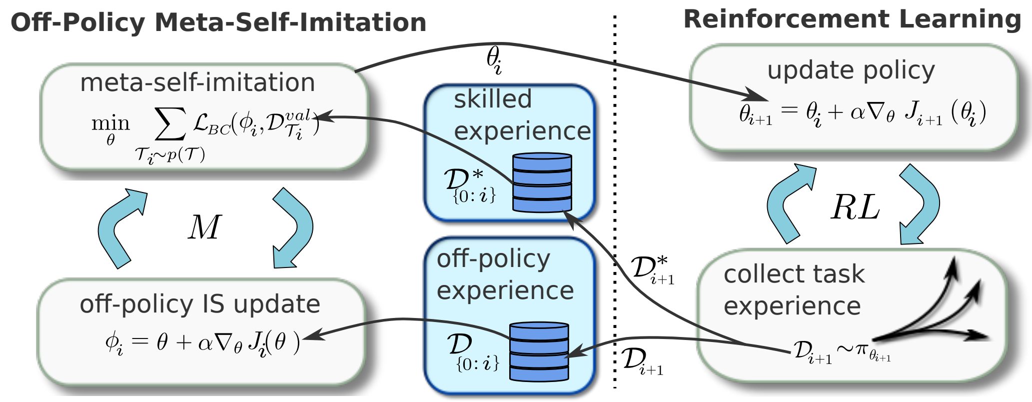


Figure 4: Outline of CoMPS. The left side corresponds to the M block for Figure 3 and the right the RL block. On the right (RL), for each round i the policy π_{θ_i} is initalized using the previously trained meta-policy parameters. At the end of policy training for round i the experience for task i is collected into the off-policy buffer $\mathcal{D}_{0:i}$ and skilled experience is stored in another buffer $\mathcal{D}^*_{0:i}$. This experience is given to M that uses the off-policy experience for the inner expected reward updates. The outer step behavior clones from the skilled experience the RL agent generated itself previously.

in Section 4.3, using samples from the full dataset of off-policy experience for that task, $\mathcal{D}_{0:i}$, to estimate an inner loop policy gradient. In effect, this procedure trains the model to learn policies that are close to the near-optimal trajectories in $\mathcal{D}^*_{0:i}$ by taking (off-policy) policy gradient steps on the sub-optimal trajectories in $\mathcal{D}_{0:i}$. The complete meta-training process is summarized in Algorithm 1, and corresponds to a reinforcement learning inner update and a meta-imitation learning outer update, though imitation uses the agent's own experience without requiring any demonstrations. In our implementation, the skilled data consists of the 20 highest-scoring rollouts per task. Further details on the networks and hyperparameters used for Algorithm 1 are available in Appendix D.

4.3 Off-Policy Inner Gradient Estimation for Meta-Learning

In this section, we describe the particular form of the inner-loop policy gradient estimator used in Algorithm 1. Although many prior works have studied importance-sampled policy gradient updates, and GMPS [30] uses an importance-sampling update based on PPO [46], we found this simple importance sampled approach to be insufficient to handle the highly off-policy data in $\mathcal{D}_{0:i}$. This is because the data for the earlier tasks may have been collected by substantially different policies than the data from the latest tasks. To enable our method to handle such highly off-policy data, we utilize *both* an improtance-sampled policy gradient estimator and an importance-sampled value estimate for the baseline in the policy gradients. The former is estimated via clipped importance weights, while the latter uses an estimator based on V-trace [11] to compute value estimates, which are then used as the baseline. For state s_m given a trajectory $(s_t, a_t, r_t)_{t=m}^{m+n}$, we define the n-step V-trace value targets for $V(s_m) = \sum_{t=0}^{n} \gamma^t r_t$, as:

$$v_m = V(s_m) + \sum_{t=m}^{m+n=1} \gamma^{t-m} \left(\prod_{i=m}^{t-1} c_i \right) \rho_t (r_t + \gamma V(s_{t+1}) - V(s_t)). \tag{3}$$

The values $\rho_t = \min(\bar{\rho}, r_t(\theta))$ and $c_i = \min(\bar{c}, r_i(\theta))$ are truncated importance weights, where $\bar{\rho}$ and $\bar{c}$ are hyperparameters, and $r_t(\theta) = \pi(a_t|s_t, \theta)/\pi(a_t|s_t, \theta')$, where θ' denotes the parameter vector of the policies that sampled the trajectory in the dataset. The value function parameters ω are trained to minimize the $l2$ loss $|v_m - V_\omega(s_m)|$. The V-trace value estimate is then used to estimate the advantage values for policy gradient, which are given by $\hat{A}_m = r_m + v_{m+1} - V_\omega(s_m)$, and the gradient is then given by $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \rho_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}_i$, analogously to PPO and other importance-sampled policy gradient algorithms. This importance-sampled gradient estimator is used for the inner loop update in Algorithm 1 line 5. We show in our ablation experiments that this approach is needed to enable successful meta-training using the exhaustive off-policy experience collected by CoMPS.

4.4 CoMPS Summary

An outline of the entire CoMPS algorithm, including how data is accumulated as more tasks are solved, is shown in Figure 4. The RL process keeps track of the set of trajectories that achieve the highest sum of rewards $\mathcal{D}_i^*$. The RL step returns the separate skilled experience $\mathcal{D}_i^*$, and all experience collected during RL training in $\mathcal{D}_i$. The M step uses this experience to perform meta-RL via training a model to learn how to reproduce the best policies achieved from previous tasks. Significantly, this meta-RL training process can accelerate the RL process even in fully offline settings, allowing the agent to train a meta-RL model without collecting additional experience.

5 Experiments

Our experiments aim to analyze the performance of CoMPS on both stationary and non-stationary task sequences in the continual meta-learning setting, where each task is observed once and never revisited again during the learning process. To this end, we construct a number of sequential meta-learning problems out of previously proposed (non-sequential) meta-reinforcement learning benchmarks. We separate our evaluation into experiments with stationary task distributions, where each task in the sequence is sampled identically, and non-stationary task distributions, where the tasks either become harder over time or else are selected to be maximally dissimilar for prior tasks (see discussion below). We describe the task domains and the methods in our comparisons below, with further details provided in Appendix A.

Tasks. An illustration of the tasks in our evaluation is provided in Figure 5, along with a visualization of the tasks, and includes the following task families:

Ant Goal: In this environment, a quadrupedal robot must reach different goal locations, arranged in a semicircle in front of the robot. The non-stationary distribution selects the locations that are furthest from previously chosen location each time, while the stationary one selects them at random.

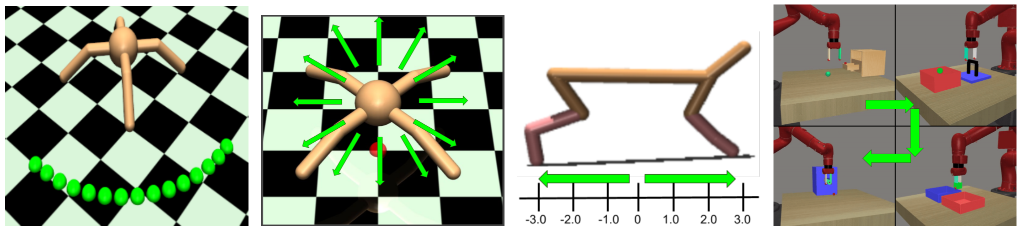
Ant Direction: Here, the same quadrupedal robot must run in a particular direction. The non-stationary and stationary distributions are constructed as above.

Half Cheetah: Here, the goal is to control the half-cheetah to run at different velocities, either forward or backward. The direction is chosen randomly, but in the non-stationary distribution, the desired velocity magnitude increases over time.

MetaWorld: We utilize the suite of robotic manipulation tasks from Yu et al. [60], which we arrange into a sequence. The non-stationary sequence orders the tasks in increasing difficulty, as measured by how long regular PPO can solve the tasks individually.

Prior methods. We compare CoMPS both to prior meta-learning methods and to prior methods for continual learning. Since learning without revisiting prior tasks requires an off-policy algorithm, we include PEARL [37] as a meta-learning baseline, which utilizes the off-policy SAC [20] algorithm and can in principle learn without revisiting prior tasks. While several prior methods use policy gradients with meta-RL [40, 1], these methods require on-policy data, making them unsuited for this continual meta-learning problem setting. For continual learning, we include a PPO transfer learning baseline (denoted PPO+TL), which trains sequentially on the tasks, as well as the more sophisticated Progress & Compress (P&C) method [47], which further guards against forgetting of prior tasks. Although we don't evaluate backward transfer, we still include P&C as a representative example of prior continual learning methods.

Meta-learning over stationary task distributions. In our first set of experiments, we compare the methods on stationary task sequences. The results are presented in Figure 6. The plots show the average number of episodes needed to reach a success threshold on each task, such that methods that solve each task faster take few episodes that task. In this evaluation protocol, once the fraction of successful rollouts exceeds a threshold, the algorithm moves on to the next task, and the goal is to solve all the tasks as fast as possible. For additional details on how success is computed and the thresholds used see Appendix A. In these experiments, we can see that CoMPS solves the tasks the fastest, and in fact solves tasks *faster* as more tasks are experienced, indicating the benefits of meta-learning. The improvements on the harder **Ant** environments are most pronounced. This indicates the benefits of continual meta-learning, where each task enables the method to solve new tasks even more quickly. Reaching the success threshold on the more challenging **MetaWorld** tasks is generally



(a) Ant Goal (b) Ant Direction (c) Half cheetah (d) MetaWorld

Figure 5: Environments and tasks used in our evaluation. In **Ant Goal** the non-stationary task distribution selects the next goal location that is furthest from all previous locations, e.g. $\mathcal{T}_{0:i} = ((5,0),(-5,0),(0,5),(0,-5),\ldots)$. In **Ant Direction** the tasks start at $0°$ along the x-axis and rotate ccw $70°$ for each new task. The **Half Cheetah** tasks start with low target velocity and alternate between larger $+-$ velocities .e.g. $\mathcal{T}_{0:i} = (0.5, -0.5, 1.0, -1.0, \ldots)$.

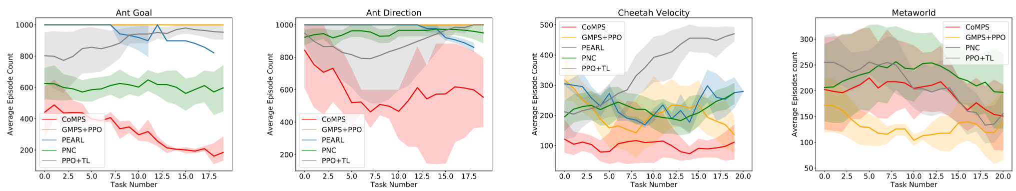


Figure 6: These figures show the average number of episodes needed to solve each new task after completing i tasks (fewer is better). Results are computed over 6 sequences of 20 tasks, averaged over 6 random seeds.

difficult, and all methods struggle with this. In Appendix E we include additional results that show CoMPS receives higher reward on average for these experiments, while in the non-stationary task analysis, we will also show that CoMPS significantly improves on **MetaWorld** in terms of average reward. In the next paragraph, we provide a more fine-grained analysis of the average rewards for each method, using the non-stationary task distributions.

Meta-learning over non-stationary task distributions. In our second set of experiments, we evaluate all of the methods on non-stationary task sequences. The results of these comparisons are presented in Figure 7, which show complete learning curves for each method over the task sequence (left), as well as a plot of the average performance on each task (right) – the plot on the right is obtained by averaging within each task, and provides a clearer visualization of aggregate performance. The plots show that CoMPS attains the best performance in each task family, and the gains are particularly large on the higher-dimensional **Ant Direction** and **Ant Goal** tasks. Note that the decrease in performance on the **Half-Cheetah** task is due to the increasing difficult of tasks later in the sequence, but CoMPS still attains higher rewards than other methods. On the two **Ant** tasks and **MetaWorld**, the average performance of CoMPS increases as more tasks are seen, indicating that the meta-learning procedure accelerates acquisition of new tasks. Note the clear improvement for CoMPS in this domain, in contrast to the comparatively inconclusive results in terms of time steps to success in the previous paragraph – since the **MetaWorld** tasks are significantly harder than **Ant** or **Cheetah**, success on each task may be out of reach for all methods [60], though the analysis in Figure 7 still shows a clear difference in terms of average rewards. PEARL generally performs poorly on the harder **Ant** tasks: although SAC is an off-policy algorithm, it is well known that such methods do not perform well when they are not allowed to gather any additional online data (as, for example, in the case of offline RL) [28]. This may account for the poor performance of PEARL and for this reason we leave it out of our **MetaWorld** results here and include them Appendix E. PPO+TL and P&C provide strong baselines, but do not benefit from meta-learning as CoMPS does, and therefore their performance does not improve as much as more tasks are observed.

Ablation study. We perform an ablation analysis to compare CoMPS to GMPS+PPO that does not use V-trace-based off-policy importance sampling. The results in Figure 6 show that GMPS+PPO can not make good use of the off-policy experience and, as a consequence, performs worse than CoMPS, especially on **Ant Goal** and **Ant Direction**. In Figure 7, where non-stationary task distributions are used, CoMPS also outperforms GMPS+PPO on average.

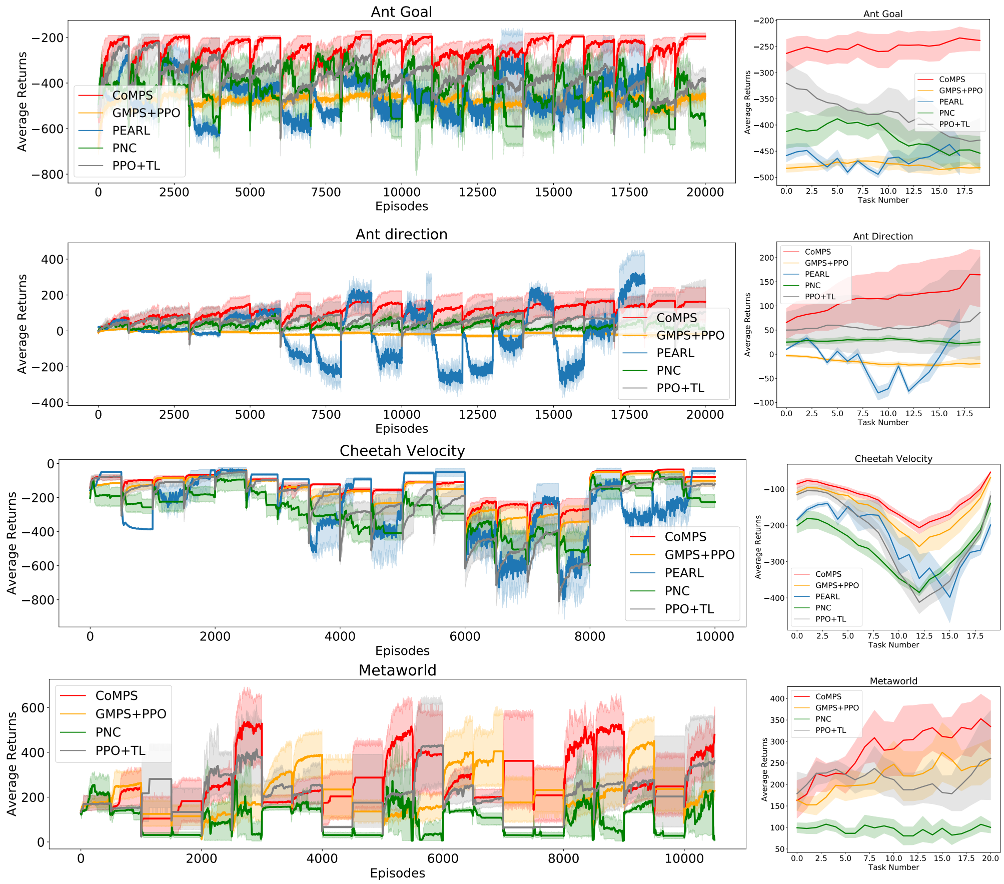


Figure 7: On the left are "lifelong" plots of rewards received for every episode of training over 20 tasks. Results are averaged over 6 seeds, and each task gets 500 episodes where each episode collects 5000 samples. The right plots show the average average return across the 500 episodes on each of the 20 individual tasks. CoMPS achieves higher average returns and improves its performance as more tasks are solved.

6 Discussion

In this work, we proposed CoMPS, a new method for continual meta-reinforcement learning. Unlike standard meta-RL methods, CoMPS learns tasks one at a time, without the need to revisit prior tasks. Our experimental evaluation shows that CoMPS can acquire long task sequences more efficiently than prior methods, and can master each task more quickly. Crucially, the more tasks CoMPS has experienced, the faster it can acquire new tasks. At the core of CoMPS is a hybrid meta-RL approach that uses an off-policy importance-sampled inner loop policy gradient updated combined with a simple supervised outer loop objective based on imitating the best data from prior tasks produced by CoMPS itself. This provides for a simple and stable approach that can be readily applied to a wide range of tasks. CoMPS does have several limitations. Like all importance-sampled policy gradient methods, the variance of the importance weights can become large, necessitating clipping and other tricks. We found that including a V-trace off-policy value estimator for the baseline helps to mitigate this, providing better performance even for highly off-policy prior task data, but better gradient estimators could likely lead to better performance in the future. Additionally, CoMPS still requires all prior data to be stored and does not provide for any mechanism to handle forgetting. While this is reasonable in some settings, an interesting direction for future work could be to develop a fully online method that does not require this. Since CoMPS does not require revisiting prior tasks, it can be a practical choice for real-world meta-reinforcement learning, and a particularly exciting direction for future work is to apply CoMPS to realistic lifelong learning scenarios for real-world applications, in domains such as robotics.

References

[1] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[2] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

[3] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*, 2020.

[4] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche . . . , 1990.

[5] Yevgen Chebotar, Artem Molchanov, Sarah Bechtle, Ludovic Righetti, Franziska Meier, and Gaurav Sukhatme. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

[6] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[7] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145, 2016.

[8] Ignasi Clavera, Anusha Nagabandi, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 3, 2018.

[9] Ron Dorfman and Aviv Tamar. Offline meta reinforcement learning. *arXiv preprint arXiv:2008.02598*, 2020.

[10] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[11] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[12] Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-q-learning. 2019.

[13] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. 2018.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

[15] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pp. 1920–1930, 2019.

[16] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[17] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. 2018.

[18] Erin Grant, Ghassen Jerfel, Katherine Heller, and Thomas L. Griffiths. Modulating transfer between tasks in gradient-based meta-learning, 2019.

[19] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 80:1861–1870, 10–15 Jul 2018.

[21] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *arXiv preprint arXiv:1912.08866*, 2019.

[22] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[23] Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *arXiv preprint arXiv:1906.05201*, 2019.

[24] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pp. 5400–5409, 2018.

[25] Khurram Javed and Martha White. Meta-learning representations for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 1820–1830. Curran Associates, Inc., 2019.

[26] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 9122–9133. Curran Associates, Inc., 2019.

[27] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 5917–5928. Curran Associates, Inc., 2019.

[28] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. 32, 2019.

[29] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[30] Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. *arXiv preprint arXiv:1904.00956*, 2019.

[31] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[32] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. *arXiv preprint arXiv:2008.06043*, 2020.

[33] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019.

[34] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[35] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019. ISSN 0893-6080.

[36] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *International Conference on Learning Representations*, 2016.

[37] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

[38] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[39] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 350–360. Curran Associates, Inc., 2019.

[40] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

[41] 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.

[42] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv*, 2016.

[43] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

[44] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[45] Jürgen Schmidhuber. POWERPLAY: training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *CoRR*, abs/1112.5309, 2011.

[46] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[47] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4528–4537, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[48] Rupesh Kumar Srivastava, Bas R. Steunebrink, and Jürgen Schmidhuber. First experiments with powerplay. *CoRR*, abs/1210.8385, 2012.

[49] Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[50] Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231, 2019.

[51] Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

[52] 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.

[53] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv*, pp. 1–6, 2016.

[54] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[55] Matthew Wallingford, Aditya Kusupati, Keivan Alizadeh-Vahid, Aaron Walsman, Aniruddha Kembhavi, and Ali Farhadi. Are we overfitting to experimental setups in recognition?, 2020.

[56] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016.

[57] Huaxiu Yao, Yingbo Zhou, Mehrdad Mahdavi, Zhenhui Li, Richard Socher, and Caiming Xiong. Online structured meta-learning. *arXiv preprint arXiv:2010.11545*, 2020.

[58] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. 2018.

[59] Tianhe Yu, Xinyang Geng, Chelsea Finn, and Sergey Levine. Variable-shot adaptation for online meta-learning. *arXiv preprint arXiv:2012.07769*, 2020.

[60] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

[61] Zhenxun Zhuang, Yunlong Wang, Kezi Yu, and Songtao Lu. Online meta-learning on non-convex setting. *arXiv preprint arXiv:1910.10196*, 2019.

[62] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pp. 7693–7702, 2019.

[63] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [No] Our code does require a license to mujoco. It is possible to get a free lisence for students use that can be used to run our code.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] You can see in our experiments section that we do back up our claim that CoMPS can acheive faster learning speeds in sequential learning problems across multiple environments.
   (b) Did you describe the limitations of your work? [Yes] At the end of Section 6 we discuss the general limitations of our method that are based on the limitations of the methods components.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix F.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Yes, we have.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We are including code with this submission. This will include instruction on how to reproduce the experiments in the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see the accompanying supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 6 and Figure 7 we included error bars and performed our experiments over 6 random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see the accompanying supplemental material in Appendix A.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use a set of environments from [37] and some code from [30].
   (b) Did you mention the license of the assets? [Yes] It is well known that Mujoco, which is needed to use the robotics simulator for our experiments, needs a software liscence.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are not using peoples data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] See above.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]