Supervised Meta-Reinforcement Learning with Trajectory Optimization for Manipulation Tasks

Supervised Meta-Reinforcement Learning with Trajectory Optimization for Manipulation Tasks

Lei Wang, Yunzhou Zhang, Delong Zhu, Sonya Coleman, Member, IEEE, Dermot Kerr

***Abstract*—Learning from small amounts of samples with reinforcement learning (RL) is challenging in many tasks, especially in real-world applications, such as robotics. Meta-Reinforcement Learning (meta-RL) has been proposed as an approach to address this problem by generalizing to new tasks through experience from previous similar tasks. However, these approaches generally perform meta-optimization by focusing direct policy search methods on validation samples from adapted policies, thus requiring large amounts of on-policy samples during meta-training. To this end, we propose a novel algorithm called *Supervised Meta-Reinforcement Learning with Trajectory Optimization (SMRL-TO)* by integrating Model-Agnostic Meta-Learning (MAML) and iLQR-based trajectory optimization. Our approach is designed to provide online supervision for validation samples through iLQR-based trajectory optimization and embed simple imitation learning into the meta-optimization rather than policy gradient steps. This is actually a bi-level optimization that needs to calculate several gradient updates in each meta-iteration, consisting of off-policy reinforcement learning in the inner loop and online imitation learning in the outer loop. SMRL-TO can achieve significant improvements in sample efficiency without human-provided demonstrations, due to the effective supervision from iLQR-based trajectory optimization. In this paper, we describe how to use iLQR-based trajectory optimization to obtain labeled data and then how leverage them to assist the training of meta-learner. Through a series of robotic manipulation tasks, we further show that compared with the previous methods, the proposed approach can substantially improve sample efficiency and achieve better asymptotic performance.**

***Index Terms*—Reinforcement learning, meta learning, iLQR, trajectory optimization, robotic manipulation.**

I. Introduction

REINFORCEMENT learning (RL) provides a powerful framework for automatic acquisition of behaviour skills by interacting with unstructured environments. In recent years, with the assistance of powerful and flexible neural network representations, RL has made great success in many fields ranging from playing games [1]–[5] to autonomous flying and driving [6]–[12]. Although these RL methods can achieve state-of-the-art results in many domains by learning a neural network policy, they often require a large amount of experiential data from system interaction. Hence, it is often very challenging for these RL approaches to be used in real-world applications where experience must be collected on real physical systems, such as robots, which can be costly.

Recently, meta-reinforcement learning (meta-RL) has been proposed as an approach to learn from previous similar experience rather than from scratch. Built on the fact that most tasks to be solved usually share common structure, the mechanism for meta-RL is primarily to learn internal representations and/or constraint rules, both of which are widely applicable to many tasks. Once learned, the meta-learner will quickly adapt to a new task through only a few interactions with the environment. While most current meta-RL algorithms have made great progress in a variety of tasks, they generally train a meta-learner by focusing direct policy search methods on validation samples from adapted policies, thus requiring large amounts of on-policy samples during the meta-training procedure [13]–[15]. To this end, [16] proposed a context-based off-policy meta-RL algorithm (PEARL) that learns a probabilistic encoder of tasks for sample efficiency. However, these direct on- and off-policy search methods in the meta-RL context easily fall into local optimum and suffer from sample complexity. To alleviate the problem, [17] proposed an off-policy meta-RL algorithm, Guided Meta-Policy Search (GMPS) that uses imitation learning in meta-optimization through previously learned expert policies or human-provided demonstrations across tasks. Despite great advances, this approach is limited to domains where human demonstrations have been provided. When demonstrations are not available, the policy of each meta-training task is trained with large amounts of data through standard single-task RL algorithms to obtain expert trajectories which are used by GMPS. In our work, we determined that when adapting to each task through off-policy data during meta-training, GMPS suffers from the problem posed by the mixture of both instability and imbalance of adaption [18]. In addition, since training global experts for all tasks and learning meta-level policies π are completely separated in GMPS, the global experts find it difficult to correctly provide supervision due to the distribution mismatch.

We determine in this work that supervision itself does not need to be acquired by directly learning the global optimal policies of meta-training tasks, although using imitation learning can effectively help train a meta-level policy. Instead, we just need to provide supervision only on the trajectories collected in each iteration of the meta-training phase. This way of providing targeted online supervision

Lei Wang is with Faculty of Robot Science and Engineering, Northeastern University, Shenyang, China.

Yunzhou Zhang is with College of Information Science and Engineering, Northeastern University, Shenyang, China (Corresponding author: Yunzhou Zhang, Email: zhangyunzhou@mail.neu.edu.cn).

Delong Zhu is with Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China.

Sonya Coleman and Dermot Kerr are with School of Computing, Engineering and Intelligent Systems, Ulster University, N. Ireland, UK.

avoids distribution mismatch. When available, the meta-RL algorithm can obtain direct supervision and enable efficient meta-training by leveraging imitation learning, while avoiding falling into local minimum. In this work, we demonstrate that a practical idea is to use the trajectory optimization with iterative LQR (iLQR) [19], where the dynamic system is locally linear and the cost is quadratic around the current trajectory. Given an initial trajectory with only a few samples, iLQR can calculate an optimal trajectory from the initial to the target state by iteratively solving for the locally optimal policy. Compared with the neural network for solving the whole task, the locally optimal policy for tracking a single trajectory is easy to learn.

In this paper, we develop a novel off-policy meta-RL algorithm called ***S**upervised **M**eta-**R**einforcement **L**earning with **T**rajectory **O**ptimization (**SMRL-TO**)*. As illustrated in Figure 1, our algorithm integrates Model-Agnostic Meta-Learning (MAML) [15] with iLQR-based trajectory optimization, which involves a bi-level optimization including off-policy RL in the inner loop and online imitation learning in the outer loop. In each meta-iteration, this bi-level optimization needs to calculate several gradient updates on the collected samples. To resist the composite effect from instability and imbalance, our approach exploits a clipping surrogate objective function in the inner loop. Our approach can improve the sample efficiency without human-provided demonstrations and will not consume large amounts of additional samples like GMPS, particularly in domains where trajectories are not easy to collect. In addition, we only use trajectory optimization in meta-training. In meta-testing, we still use vanilla policy gradient like MAML. Therefore, our approach has the same runtime performance as MAML and is also easy to use, while reducing the sample complexity. As far as we know, our work is the first to combine trajectory optimization and RL in the meta-learning context. We perform the experimental evaluation in robotic manipulation environments. Compared with the previous meta-RL methods, SMRL-TO achieves significant gains in sample efficiency and has better asymptotic performance. We further conduct an ablation study to understand the impact of the components of our approach.

To summarize, the core contributions of our work are triple:

- We propose a novel off-policy meta-RL architecture called SMRL-TO, which achieves significant improvements in sample efficiency without human-provided demonstrations due to the effective supervision from trajectory optimization.
- We describe how to use iLQR-based trajectory optimization to obtain the local expert policies, resulting in supervised data, as shown in Section IV-A.
- We show how to leverage supervised data from Section IV-A to train the meta-learner. Moreover, we exploit a clipping surrogate objective to suppress the instability and guarantee the balance of adapting to tasks during meta-training, as shown in Section IV-B.

II. Related Work

A. Meta-Reinforcement Learning

Our work builds on the meta-RL framework that is inspired by the fact that human beings can realize fast learning based on past experience. Recently, a variety of different meta-RL approaches has been proposed to tackle sample complexity. In one approach, the algorithm is encoded with the weights of the recurrent neural network, which are trained with states across episodes within a task [14], [20]–[22]. Another approach is to meta-optimize the parameters of the neural network and then fine tune these parameters at test time on the new task [15], [23], [24]. Other approaches generally meta-learn constraint rules, such as some hyperparameters that are usually set manually [25], the loss function [26] and the exploration strategies [13]. These approaches focus direct policy methods or evolutionary strategies on on-policy samples to acquire an optimal meta-learner. In contrast, we introduce a supervision mechanism and embed simple imitation learning into meta-optimization instead of policy gradient steps. This permits our algorithm to obtain more information from a few samples and achieve significant improvements in sample efficiency during meta-training.

B. Guided Meta-Policy Search

Meta Imitation Learning [27] combines MAML [15] with imitation learning using expert video demonstrations to alleviate sample complexity during meta-training. Similarly, EMRLD [28] jointly utilizes RL and imitation learning over the expert data to generate a meta-policy and focuses on the sparse reward environments. Closest to our approach is Guided Meta-Policy Search (GMPS) [17] that combines imitation learning with RL by leveraging offline expert demonstrations in the meta-RL context. Specifically, GMPS separates meta-training into two phases. In the first phase, GMPS learns the global optimal expert policies for all meta-training tasks by standard single-task RL algorithms or imitation learning with human-provided demonstrations. In the second phase, GMPS uses the learned expert policy for each meta-training task to relabel the data from the current adapted policy and then performs meta-optimization with imitation learning. Despite excellent sample efficiency, these approach are limited to some domains where human-provided demonstrations or optimal expert policies of each task have been obtained in advance. In contrast, our approach integrates a component of trajectory optimization into the meta-RL framework, which can provide online supervision for validation samples without human-provided demonstrations.

C. Guided Policy Search

Previous methods, such as varieties of Guided Policy Search algorithms [29]–[33], have also explored the combination of trajectory optimization and imitation learning for efficiency in the context of RL. Similar to our approach, these methods create a local approximation to the environmental dynamics, and then learn a set of simple local "expert policies" by using trajectory optimization. However, the purpose of these methods is to obtain the global optimal policy for single tasks by using as few samples as possible. In contrast, our approach aims to train a meta-learner (or meta-level policy) that can quickly adapt to new tasks rather than apply a single optimal policy, so as to reduce the pressure on needing perfect "expert

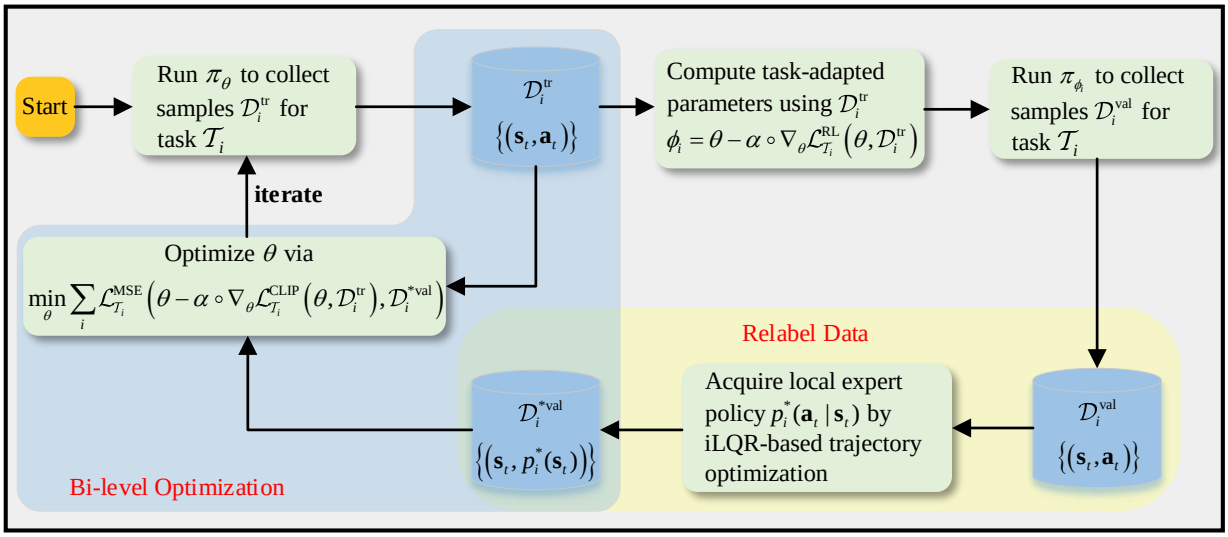

Figure 1. Meta-training procedure for the SMRL-TO algorithm. We aim to learn a meta-level policy π_θ that quickly adapts to new tasks by leveraging only a few interactions in the meta-testing. In each meta-iteration, the algorithm first collects samples $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{val}}$ for each task $\mathcal{T}_i$. Then using iLQR-based trajectory optimization acquires each local expert policy p_i^* for $\mathcal{D}_i^{\text{val}}$ and relabels data, resulting in supervised data $\mathcal{D}_i^{*\text{val}}$ (shown in the yellow box). Finally, the algorithm performs the bi-level optimization across batch tasks (shown in the blue box). Iterate the above process.

policies". In this respect, our approach can be regarded as a relaxed extension of the Guided Policy Search algorithm [29] in the meta-learning setting.

D. Trajectory Optimization

Trajectory optimization algorithms deal with the problem of optimizing trajectory-centric policies and minimizing the cost of individual trajectories [33], [34]. In one approach, the algorithms are model-free and fit the policies to the best or weighted samples, needing for "trial and error" random exploration [35]–[37]. These algorithms can handle arbitrary complex dynamics at the cost of sample efficiency. Another approach is to model the global dynamical systems [38]–[40]. For example, PILCO [40] learns a global dynamic model with Gaussian processes to optimize policies, which relies on large nonlinear optimizations in each iteration. Although these model-based algorithms typically have the advantage of improving sample efficiency, they are not always able to accurately model complex and unknown dynamical systems to obtain effective policies [41]. In contrast, our method does not require learning a global model, instead iteratively solving for the locally optimal policy through iLQR [19] under local and time-varying linear dynamics model. Our method can deal with complex and discontinuous dynamics with fewer samples, combining the advantages of model-free and model-based methods.

III. Preliminaries

In this section, we describe the meta-RL problem formulation and trajectory optimization that our work builds on.

A. Meta-RL Problem Statement

Meta-RL is a meta-learning framework for reinforcement learning. The goal of meta-RL is to learn a reinforcement learner (or meta-learner), which can quickly learn an optimal policy for a new task after interacting with the environment a few times by making full use of previous knowledge and experience. To this end, meta-RL algorithms require samples across different tasks drawn from a distribution $\rho(\mathcal{T})$ to train the reinforcement learner. Considering the distribution over tasks, each task $\mathcal{T}_i$ corresponds to a Markov decision process (MDP), which shares common action and state space, but differs in the reward function and/or the environmental dynamics.

In this work, we only consider the well-known gradient-based meta-RL method called Model-Agnostic Meta-Learning (MAML) [15], which our approach builds on. Formally, we define a finite-horizon MDP at discrete time for each RL task $\mathcal{T}_i$ by a tuple $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}^i, \mathcal{R}^i, \gamma, H)$, wherein $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathcal{P}^i$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition (or dynamics) distribution for task $\mathcal{T}_i$, $\mathcal{R}^i$: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for task $\mathcal{T}_i$, $\gamma \in [0, 1]$ is the discount factor, and H is the episode length. We consider the reinforcement learner to be a policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$, which maps state $\mathbf{s}_t$ to $\mathbf{a}_t$. The loss function of a reinforcement learning problem for task $\mathcal{T}_i$ is the following:

$$\mathcal{L}_{\mathcal{T}_i}^{\text{RL}}(\theta, \mathcal{D}_i) = -\mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\theta, \mathcal{P}^i}\left[\sum_{t=1}^{H} \gamma^{t-1} r^i(\mathbf{s}_t, \mathbf{a}_t)\right], \quad (1)$$

where θ is the optimized parameters and $\mathcal{D}_i$ denotes the samples from task $\mathcal{T}_i$. During meta-training, the policy is trained with samples $\mathcal{D}_i^{\text{tr}}$ by the policy gradient for each task $\mathcal{T}_i$ and the updated parameters are denoted as ϕ_i:

$$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{RL}}(\theta, \mathcal{D}_i^{\text{tr}}), \quad (2)$$

where α is the fast adaptation learning rate. Then we collect new samples $\mathcal{D}_i^{\text{val}}$ from $\mathcal{T}_i$ via the adapted policy parameters

ϕ_i and update the model parameters θ via stochastic gradient descent as follows:

$$\begin{aligned}\theta &\leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}_{\mathcal{T}_i}^{\mathrm{RL}}\left(\phi_i, \mathcal{D}_i^{\mathrm{val}}\right) \\ &= \theta - \beta \nabla_\theta \sum_i \mathcal{L}_{\mathcal{T}_i}^{\mathrm{RL}}\left(\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\mathrm{RL}}\left(\theta, \mathcal{D}_i^{\mathrm{tr}}\right), \mathcal{D}_i^{*\mathrm{val}}\right),\end{aligned} \tag{3}$$

where β is the meta learning rate.

Subsequently, during meta-testing, a new task $\mathcal{T}_{\mathrm{test}}$ is drawn from $\rho(\mathcal{T})$, and the meta-learner with updated parameters θ can adapt its behaviors to the new task via a few interactions $\mathcal{D}^{\mathrm{test}}$ by gradient descent:

$$\phi_{\mathrm{test}} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_{\mathrm{test}}}^{\mathrm{RL}}\left(\theta, \mathcal{D}^{\mathrm{test}}\right). \tag{4}$$

In the meta-training phase of meta-RL, direct policy search requires a large number of on-policy samples to train a meta-level policy. In this work, we instead direct the training by leveraging local "expert policies". In the next section, we will describe how to acquire these "expert policies" by trajectory optimization.

B. Trajectory Optimization with iLQR

Trajectory optimization is a technique for optimizing the parameters of the local policy $p(\mathbf{a}_t|\mathbf{s}_t)$, a probability distribution with respect to a trajectory $\tau = \{\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_H, \mathbf{a}_H\}$ that is a sequence of states and actions over a period of time. Given some cost function $\ell(\mathbf{s}_t, \mathbf{a}_t)$ and system transition distribution $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$, we define the total cost of the trajectory as $\ell(\tau) = \sum_{t=1}^{T} \ell(\mathbf{s}_t, \mathbf{a}_t)$ and optimize the policy to minimize the expectation $E_{p(\tau)}[\ell(\tau)] = \int \ell(\tau) p(\tau) d\tau$ under the policy trajectory distribution

$$p(\tau) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) p(\mathbf{a}_t \mid \mathbf{s}_t),$$

where $p(\mathbf{s}_1)$ is the initial state distribution.

The local policy can be optimized efficiently with a small number of samples using the iterative LQR (iLQR) algorithm [19], [29]. The iLQR method iteratively optimizes trajectories by solving for local expert policies under the assumption of linear dynamics and quadratic costs. The system dynamics $f(\mathbf{s}_t, \mathbf{a}_t)$ are approximated as the time-varying linear-Gaussian form $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(f_{\mathbf{s}t}\mathbf{s}_t + f_{\mathbf{a}t}\mathbf{a}_t + f_{ct}, \mathbf{F}_t)$, where $\mathbf{F}_t$ denotes the covariance of the Gaussian noise. Given a trajectory, denoted $\{\hat{\mathbf{s}}_1, \hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{s}}_H, \hat{\mathbf{a}}_H\}$, the iLQR algorithm approximates the quadratic value function and the Q-function as follows:

$$\begin{aligned}V(\mathbf{s}_t) &= \frac{1}{2}\mathbf{s}_t^T V_{\mathbf{s},\mathbf{s}t}\mathbf{s}_t + \mathbf{s}_t^T V_{\mathbf{s}t} + \text{const} \\ Q(\mathbf{s}_t, \mathbf{a}_t) &= \frac{1}{2}\mathbf{s}_t^T Q_{\mathbf{s},\mathbf{s}t}\mathbf{s}_t + \frac{1}{2}\mathbf{a}_t^T Q_{\mathbf{a},\mathbf{a}t}\mathbf{a}_t + Q_{\mathbf{s}t}\mathbf{s}_t + Q_{\mathbf{a}t}\mathbf{a}_t \\ &\quad + \mathbf{s}_t^T Q_{\mathbf{s},\mathbf{a}t}\mathbf{a}_t + \text{const}.\end{aligned}$$

Subscripts denote derivatives, e.g. $Q_{\mathbf{s}t}$ is the gradient of Q at time t with respect to $\mathbf{s}_t$, $Q_{\mathbf{s},\mathbf{s}t}$ is the Hessian, and so forth. Under the approximation, we can recursively compute the derivatives of the Q-function as:

$$\begin{aligned}Q_{\mathbf{s}t} &= \ell_{\mathbf{s}t} + f_{\mathbf{s}t}^T V_{\mathbf{s}t+1} & Q_{\mathbf{s},\mathbf{s}t} &= \ell_{\mathbf{s},\mathbf{s}t} + f_{\mathbf{s}t}^T V_{\mathbf{s},\mathbf{s}t+1} f_{\mathbf{s}t} \\ Q_{\mathbf{a}t} &= \ell_{\mathbf{a}t} + f_{\mathbf{a}t}^T V_{\mathbf{s}t+1} & Q_{\mathbf{a},\mathbf{a}t} &= \ell_{\mathbf{a},\mathbf{a}t} + f_{\mathbf{a}t}^T V_{\mathbf{s},\mathbf{s}t+1} f_{\mathbf{a}t} \\ Q_{\mathbf{a},\mathbf{a}t} &= \ell_{\mathbf{a},\mathbf{a}t} + f_{\mathbf{a}t}^T V_{\mathbf{s},\mathbf{s}t+1} f_{\mathbf{a}t} & Q_{\mathbf{a},\mathbf{s}t} &= \ell_{\mathbf{a},\mathbf{s}t} + f_{\mathbf{a}t}^T V_{\mathbf{s},\mathbf{s}t+1} f_{\mathbf{s}t}\end{aligned}$$

as well as the derivatives of the value functions:

$$\begin{aligned}V_{\mathbf{s}t} &= Q_{\mathbf{s}t} - Q_{\mathbf{a},\mathbf{s}t}^T Q_{\mathbf{a},\mathbf{a}t}^{-1} Q_{\mathbf{a}t} \\ V_{\mathbf{s},\mathbf{s}t} &= Q_{\mathbf{s},\mathbf{s}t} - Q_{\mathbf{a},\mathbf{s}t}^T Q_{\mathbf{a},\mathbf{a}t}^{-1} Q_{\mathbf{a},\mathbf{s}t}\end{aligned}$$

The locally optimal policy is given by:

$$g(\mathbf{s}_t) = \hat{\mathbf{a}}_t + \mathbf{k}_t + \mathbf{K}_t(\mathbf{s}_t - \hat{\mathbf{s}}_t), \tag{5}$$

where $\mathbf{k}_t = -Q_{\mathbf{a},\mathbf{a}t}^{-1} Q_{\mathbf{a}t}$ and $\mathbf{K}_t = -Q_{\mathbf{a},\mathbf{a}t}^{-1} Q_{\mathbf{a},\mathbf{a}t}$ denote the linear bias term and feedback term respectively. The detailed formula derivation process of iLQR algorithm can be found in [42].

In this work, we employ the iLQR algorithm to learn a time-varying linear-Gaussian policy as following

$$p(\mathbf{a}_t \mid \mathbf{s}_t) = \mathcal{N}\left(g(\mathbf{s}_t), \mathbf{C}_t\right), \tag{6}$$

where $\mathbf{C}_t = Q_{\mathbf{a},\mathbf{a}t}^{-1}$ denotes the covariance.

IV. Supervised Meta-Reinforcement Learning

In this section, we will combine MAML [15] with iLQR-based trajectory optimization to design a novel meta-RL algorithm called SMRL-TO for improving sample efficiency during meta-training. SMRL-TO solves the meta-optimization problem in two steps: SMRL-TO first explicitly learns local "expert policies" around on-policy validation samples by iLQR-based trajectory optimization and produces supervised data in Section IV-A; these data are then used to direct the training of the meta-level policy with imitation learning in Section IV-B.

A. Learning Local Expert Policy

In this work, we apply iLQR-based trajectory optimization to validation samples $\mathcal{D}_i^{\mathrm{val}}$ collected from the adapted policy π_{ϕ_i} in task $\mathcal{T}_i$. Given an initial trajectory from $\mathcal{D}_i^{\mathrm{val}}$, iLQR can optimize the parameters of the local policy $p_i(\mathbf{a}_t|\mathbf{s}_t)$ with respect to this trajectory. While the linear-Gaussian policy is simple, it admits a very efficient optimization process and works well for individual trajectory. Since the trajectories from each task are optimized independently, we can omit the subscript i in order to be more concise in this section. For continuous control tasks with complex environmental dynamics, it is difficult to design the system dynamics $\mathcal{N}(f_{\mathbf{s}t}\mathbf{s}_t + f_{\mathbf{a}t}\mathbf{a}_t + f_{ct}, \mathbf{F}_t)$ through domain knowledge. Fortunately, we can use linear regression to estimate these system dynamics over the current samples $\{(\hat{\mathbf{s}}_t, \hat{\mathbf{a}}_t, \hat{\mathbf{s}}_{t+1})\}$ in $\mathcal{D}^{\mathrm{val}}$. Since the number of samples scales with the state dimension, this method suffers from severe sample complexity at each iteration to obtain a good dynamics model.To greatly reduce the number of samples, we employ a Gaussian mixture model (GMM) to construct a normal-inverse-Wishart (NIW)

prior of the dynamics distribution [43]. We define the NIW prior using parameters Φ, μ_0, m and n_0. After obtaining these prior parameters from several previous iterations, we can calculate the maximum a posteriori estimate (MAP) on $[\mathbf{s}_t; \mathbf{a}_t; \mathbf{s}_{t+1}]$ of the system dynamics, given by

$$\mu = \frac{m\mu_0 + n_0\hat{\mu}}{m + n_0}$$
$$\Sigma = \frac{\Phi + N\hat{\Sigma} + \frac{Nm}{N+m}\left(\hat{\mu} - \mu_0\right)\left(\hat{\mu} - \mu_0\right)^{\mathrm{T}}}{N + n_0},$$

where $\hat{\mu}$ and $\hat{\Sigma}$ are respectively the empirical mean and empirical covariance of N trajectory samples $\{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\}$. To obtain these prior parameters, we fit a Gaussian distribution over current samples $\{(\hat{\mathbf{s}}_t, \hat{\mathbf{a}}_t, \hat{\mathbf{s}}_{t+1})\}$ and obtain the mean μ_0 and covariance Φ. Here m and n_0 should be the number of samples; in this work, we set them both to 1.

When applied to the iLQR-based trajectory optimization method, the system dynamics $\mathcal{N}(\mu, \Sigma)$ are then approximated as linear-Gaussian dynamics, $\mathcal{N}(f_{\mathbf{s}t}\mathbf{s}_t + f_{\mathbf{a}t}\mathbf{a}_t + f_{ct}, \mathbf{F}_t)$, given by

$$f_{\mathbf{sa}t} = \Sigma^{-1}_{[\mathbf{sa}t,\mathbf{sa}t]}\Sigma_{[\mathbf{sa}t,\mathbf{s}t+1]} \qquad f_{ct} = \mu_{[\mathbf{s}t+1]} - f_{\mathbf{sa}t}\mu_{[\mathbf{sa}t]}\big]$$
$$\mathbf{F}_t = \Sigma_{[\mathbf{s}t+1,\mathbf{s}t+1]} - f_{\mathbf{sa}t}\Sigma_{[\mathbf{sa}t,\mathbf{sa}t]}f^T_{\mathbf{sa}t}.$$

Once we have obtained the system dynamics, we can use the iLQR-based trajectory optimization to obtain the local "expert policy" $p(\mathbf{a}_t \mid \mathbf{s}_t)$ of the current validation samples $\mathcal{D}^{\mathrm{val}}$. Unfortunately, the local policy and global policy may have different state distributions. Hence, we use the following constraint formulation:

$$\min_{p(\tau)} E_{p(\tau)}[\ell(\tau)] \quad \text{s.t. } D_{\mathrm{KL}}\left(p(\tau) \| \pi_\phi(\tau)\right) \leq \mu \tag{7}$$

where the KL-divergence constraint is to encourage the local policy $p\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$ to stay close to the global policy $\pi_\phi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$.

In previous work [30], a similar constraint optimization was used and μ in the constraint, was determined by other components in each iteration, so as to achieve compact constraints and accurate results. In contrast, our approach, a relaxed version, uses a fixed μ for constraint optimization. However, the resulting errors from our method will not accumulate or get magnified in practice. We empirically analyze this in detail. First, the trajectory optimizations between different tasks are independent of each other, and the generated local "expert policies" only provide supervision information for a few trajectory samples collected in each iteration, making these small deviations scattered irregularly. However, the nature for our proposed meta-RL is primarily to learn the common internal representations between different tasks, which mitigates the impact of the deviation. Second, since our approach is a nested optimization where the standard RL in the inner loop is not affected by trajectory optimization, it can alleviate the impact caused by the trajectory optimization in the previous iteration.

The local policy and the global policy have different representation forms (linear Gaussian and neural network respectively), which increases the computational complexity.

Algorithm 1 Trajectory Optimization for Meta-training Tasks

Require: Batch meta-training tasks $\{\mathcal{T}_i\}$ drawn from distribution $\rho(\mathcal{T})$
Require: The adapted policy π_{ϕ_i} for each task $\mathcal{T}_i$
Require: Validation samples $\mathcal{D}_i^{\mathrm{val}} = \{(\mathbf{s}_t, \mathbf{a}_t)\}$ from each π_{ϕ_i}
Require: Initialize labeled data $\mathcal{D}^{*\mathrm{val}} = \{\varnothing\}$
1: **for all** $\mathcal{T}_i$ **do**
2: Fit dynamics $p^i\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$ to $\mathcal{D}_i^{\mathrm{val}}$ with GMM prior
3: Fit linearized global policy $\bar{\pi}_{\phi_i}$ using samples in $\mathcal{D}_i^{\mathrm{val}}$
4: Optimize constraint objective in Equation 7 to get local expert policy $p_i^*\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$
5: Label validation samples $\mathcal{D}_i^{\mathrm{val}}$, resulting in data $\mathcal{D}_i^{*\mathrm{val}} = \{(\mathbf{s}_t, p_i^*(\mathbf{s}_t))\}$
6: Aggregate $\mathcal{D}^{*\mathrm{val}} \leftarrow \mathcal{D}^{*\mathrm{val}} \cup \mathcal{D}_i^{*\mathrm{val}}$
7: **end for**

To simplify the calculation in practice, we use the linear-Gaussian approximation $\bar{\pi}_\phi$ of the global policy π_ϕ around $\{\hat{\mathbf{s}}_t, \hat{\mathbf{a}}_t\}$. This approximation can be obtained in the same way as fitting system dynamics.

In each meta-iteration, we can obtain the local "expert policy" $p_i^*\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$ on samples $\mathcal{D}_i^{\mathrm{val}}$ for each task $\mathcal{T}_i$ by iLQR-based trajectory optimization as previously described. Then we leverage these local "expert policies" to label the validation samples for all batch tasks, resulting in supervised data $\mathcal{D}^{*\mathrm{val}}$. We summarize this procedure in Algorithm 1.

During meta-training, MAML needs to compute the Hessian-vector product of neural network parameters, and is limited to small models. In contrast, our SMRL-TO uses iLQR-based trajectory optimization to obtain local expert policies. This process is only related to the states and actions of trajectories, and does not involve the parameters of neural network. So SMRL-TO is not limited to small models.

B. Supervised Meta-Reinforcement Learning Algorithm

To reduce the sample complexity in the meta-training phase, we use simple imitation learning instead of a single-step direct policy search. This needs to calculate several gradient descent steps in each meta-iteration using the collected samples, including training samples $\mathcal{D}^{\mathrm{tr}}$ and validation samples $\mathcal{D}^{\mathrm{val}}$. When adapting to each task $\mathcal{T}_i$ drawn from the task distribution $\rho(\mathcal{T})$, and the updated model parameter ϕ_i is trained using the training samples $\mathcal{D}_i^{\mathrm{tr}}$ from the current policy with parameters θ_{init} at each step of gradient descent. Therefore, we only collect samples $\mathcal{D}_i^{\mathrm{tr}}$ once for each task $\mathcal{T}_i$ using the policy $\pi_{\theta_{\mathrm{init}}}$ at the start of a meta-iteration. However, after the first step of gradient descent, the target policy π_θ we need to learn has already changed from the initial policy $\pi_{\theta_{\mathrm{init}}}$ used to collect samples. To solve this, we use the importance weighted policy gradient, which is one of the most popular tricks for off-policy learning. At each gradient step, the adapted parameter ϕ_i for each task $\mathcal{T}_i$ is calculated by

$$\phi_i = \theta + \alpha\nabla_\theta \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\theta_{\mathrm{init}}}}\left[r_t(\theta)\hat{A}_t\right], \tag{8}$$

where $r_t(\theta) = \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{\text{init}}}(\mathbf{a}_t|\mathbf{s}_t)}$ denotes the marginal importance weights and $\hat{A}_t$ denotes an estimate of the advantage function at time step t.

In this scheme, if there is no constraint on importance weights $r_t(\theta)$, the adapted parameters ϕ_i in Equation 8 will be updated greatly, since θ gradually moves away from θ_{init} in the gradient updates. This can cause instability and slow convergence during training. Although simply forcing the fast adaptation learning rate α to reduce can alleviate this problem, it can not prevent policy parameters ϕ_i from the occasional large updates, and thus can not maintain a desired learning speed. Additionally, the multi-task form in the meta-RL framework will aggravate this problem as the loss required for updating policy parameters is obtained by averaging the losses of batch tasks in each iteration. If the loss of one task is occasionally larger than that of other tasks because of importance weights, the policy update will be biased towards the task, resulting in imbalance of adapting to tasks. As a result, the learned policy parameters in the meta-training phase may be more sensitive to the loss function of new tasks during meta-testing and can easily achieve rapid adaptation, while the other new tasks are just the opposite [18].

Actually, this is a constrained optimization problem. In the MAML method, we can ignore this constraint since $r_t(\theta)$ is always equal to 1. In this work, we apply the clipping trick [44] to the objective function to alleviate this problem. Although simple, the clipping trick turns the constrained optimization into unconstrained optimization, which makes it easier to train the meta-learner through gradient descent. The "surrogate" objective we proposed is as follows:

$$\mathcal{L}_{\mathcal{T}_i}^{\text{CLIP}}\left(\theta, \mathcal{D}_i^{\text{tr}}\right) = -\mathbb{E}_{\tau\sim\pi_{\theta_{\text{init}}}}\left[\min\left(r_t(\theta)\hat{A}_t, r_t^{\text{clip}}(\theta)\hat{A}_t\right)\right], \tag{9}$$

where $r_t^{\text{clip}}(\theta) = \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)$, and ϵ is a hyper-parameter. When ϵ is within a reasonable range, the clipping trick is not sensitive to ϵ, which is also what we expect. In this work, we set this parameter ϵ as the default value 0.2, which was first proposed in the PPO algorithm [44]. Then we take the gradient step for each task $\mathcal{T}_i$ on the policy parameters $\theta : \bar{\phi}_i = \theta - \alpha \circ \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{CLIP}}\left(\theta, \mathcal{D}_i^{\text{tr}}\right)$. In our approach, the fast adaptation learning rate α is meta-learned like Meta-SGD [45]. The objective $\mathcal{L}^{\text{CLIP}}$ makes the updated parameters $\bar{\phi}_i$ similar to θ_{init} by clipping importance weight $r_t(\theta)$. It is worth noting that we use the clipping objective only when the optimizing meta-objective, and still use the vanilla loss function when collecting trajectories. Compared with the on-policy method PPO, the approach we proposed applies the clipping trick to the off-policy setting in the inner loop of meta-RL and performs well in practice.

After computing the adapted parameters $\bar{\phi}_i$ for each task $\mathcal{T}_i$, we will perform meta-optimization with simple imitation learning by leveraging the mean-squared-error (MSE) loss function:

$$\mathcal{L}_{\mathcal{T}_i}^{\text{MSE}}\left(\bar{\phi}_i, \mathcal{D}_i^{*\text{val}}\right) = \sum_{\mathbf{s}^{(j)}, \mathbf{a}^{(j)} \sim \mathcal{D}_i^{*\text{val}}} \left\| \pi_{\bar{\phi}_i}(\mathbf{s}^{(j)}) - \mathbf{a}^{(j)} \right\|_2^2, \tag{10}$$

Algorithm 2 SMRL-TO Meta-training.

Require: Distribution over tasks $\rho(\mathcal{T})$;
Require: Meta learning rate β.
1: Initialize policy network π_θ and fast adaptation learning rate α.
2: **while** not done **do**
3: initialize θ_{init} with weights θ
4: Sample batch tasks $\mathcal{T}_i \sim \rho(\mathcal{T})$
5: **for all** $\mathcal{T}_i$ **do**
6: Collect samples $\mathcal{D}_i^{\text{tr}}$ using π_θ in task $\mathcal{T}_i$
7: Compute adapted parameters using $\mathcal{D}_i^{\text{tr}}$: $\phi_i = \theta - \alpha \circ \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{RL}}\left(\theta, \mathcal{D}_i^{\text{tr}}\right)$
8: Collect samples $\mathcal{D}_i^{\text{val}} = \{(\mathbf{s}_t, \mathbf{a}_t)\}$ using π_{ϕ_i} in task $\mathcal{T}_i$
9: **end for**
10: Produce supervised data $\mathcal{D}^{*\text{val}}$ using Algorithm 1
11: **for** iteration $k = 1, ..., K$ **do**
12: **for all** $\mathcal{T}_i$ **do**
13: Compute importance weights $r_t(\theta) = \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{\text{init}}}(\mathbf{a}_t|\mathbf{s}_t)}$ in task $\mathcal{T}_i$
14: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{CLIP}}\left(\theta, \mathcal{D}_i^{\text{tr}}\right)$ using $\mathcal{D}_i^{\text{tr}}$ and $r_t(\theta)$ according to Equation 9
15: Compute adapted parameters by gradient descent: $\bar{\phi}_i = \theta - \alpha \circ \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{CLIP}}\left(\theta, \mathcal{D}_i^{\text{tr}}\right)$
16: Sample supervised data $\mathcal{D}_i^{*\text{val}} \sim \mathcal{D}^{*\text{val}}$ for the meta-update
17: **end for**
18: $(\theta, \alpha) \leftarrow (\theta, \alpha) - \beta \nabla_\theta \sum_i \mathcal{L}_{\mathcal{T}_i}^{\text{MSE}}(\bar{\phi}_i, \mathcal{D}_i^{*\text{val}})$ in Equation 10
19: **end for**
20: **end while**

where $\mathcal{D}_i^{*\text{val}}$ is the labeled validation samples for each task $\mathcal{T}_i$ from Section IV-A.

The structure of SMRL-TO is summarized in Algorithm 2. In each meta-iteration, we first initialize θ_{init} with meta-level policy weights θ on line 3 for further calculation of importance weights. Similar to the MAML method [15], we respectively collect samples $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{val}}$ through π_θ and π_{ϕ_i} for each drawn task $\mathcal{T}_i$ on lines $4-9$. Unlike prior meta-RL methods, we leverage iLQR-based trajectory optimization to produce supervised data $\mathcal{D}^{*\text{val}}$ on line 10 for meta-updates. Subsequently, we optimize the meta-objective for K gradient steps with the same batch samples on lines $11-19$, including computing adapted parameters $\bar{\phi}_i$ with the clipping objective $\mathcal{L}_{\mathcal{T}_i}^{\text{CLIP}}$ on line 15 and updating θ and α to optimize the mean-squared-error objective $\mathcal{L}_{\mathcal{T}_i}^{\text{MSE}}$ over all batch tasks on line 18.

V. Experiments

The aim of our experimental evaluation is to answer the following questions: (1) without human-provided demonstrations, can SMRL-TO use imitation learning to learn a meta-learner only using the online supervision from trajectory optimization? (2) is SMRL-TO better in sample efficiency than previous meta-RL algorithms during meta-training? (3) can SMRL-TO quickly adapt to new tasks during meta-testing?

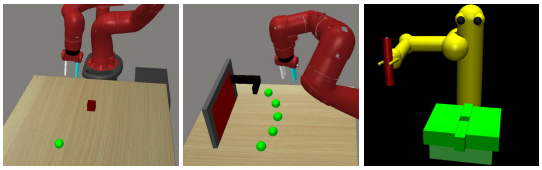

Figure 2. 3D robotic manipulation environments for experimental evaluation. From left to right: Pusher, Door Opening, Peg Insertion. In the Pusher task, a Sawyer arm pushes a red block to the target position indicated by the green marker. The goal of Door Opening task is to open the door to a target angle sampled from 30 to 90 degrees as shown by the green markers. In Peg Insertion, the robotic arm insert a peg into a hole, where the goal location of hole is sampled from a 20 cm $\times$ 20 cm region.

A. Experimental Setup

To answer these questions, we evaluate our algorithm using simulated robotic manipulation tasks as shown in Figure 2. Each task needs to perform robotic motor control based on the MuJoCo physics engine [46]. For further design of reward function, we use the following general metric on vectors $\mathbf{z}_1$ and $\mathbf{z}_2$:

$$\ell\left(\mathbf{z}_1, \mathbf{z}_2\right) = w_\alpha \left\|\mathbf{z}_1\right\|^2 + w_\beta \log\left(\left\|\mathbf{z}_2\right\|^2 + \epsilon\right), \tag{11}$$

where w_α and w_β are the parameters corresponding to weight squared loss and log-square loss, respectively, and we set $\epsilon = 10^{-5}$.

1) Sawyer Manipulation Tasks: The simulation environments used for the tasks come from the Meta-World [47], which is a benchmark for multi-task and meta reinforcement learning. The tasks involve moving a 7-DoF Sawyer arm with a gripper by motor control rather than 3D position control. For two specific Sawyer tasks, we use the general reward function

$$R = -w_1 \ell\left(\mathbf{z}_1, \mathbf{z}_2\right) - w_2 \ell\left(\mathbf{z}_3, \mathbf{z}_4\right) - w_{\mathbf{a}} \left\|\mathbf{a}\right\|^2 - w_{\mathbf{s}} \left\|\mathbf{s}\right\|^2 \tag{12}$$

$$\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{d}_1 \tag{13}$$

$$\mathbf{z}_3 = \frac{t}{H}\mathbf{d}_2 \qquad \mathbf{z}_4 = (\frac{t}{H})^2\mathbf{d}_2 \tag{14}$$

where H denotes the horizon length, ℓ is the general metric described in Equation 11, $w_{\mathbf{a}}$ and $w_{\mathbf{s}}$ weight the squared losses of actions and states, respectively. The details of $\mathbf{d}_1$ and $\mathbf{d}_2$ are described later. The two specific tasks are as follows:

- Pusher: In these tasks, the Sawyer arm needs to push a red block of wood with a gripper to the target marked by the green ball, which is sampled from a 20 cm $\times$ 10 cm region. The full state observations consist of joint angles, velocities, block position and end-effector position, but not target positions. The Sawyer arm needs to acquire the ability to adapt to different target positions through training. In the reward function of this environment, $\mathbf{d}_1$ is the distance between the block and the gripper, and $\mathbf{d}_2$ is the distance between the block and the target.
- Door Opening: These tasks are challenging, requiring 3D control of the arm to grab the door handle and then open the door to a target angle sampled from 30 to 90 degrees. The full state observations consist of joint angles, velocities, handle position and end-effector position, but not target angle/position. In the reward function for this environment, $\mathbf{d}_1$ is the distance between the door handle and the gripper and $\mathbf{d}_2$ is the distance between the door handle and the target.

Table I
PARAMETERS OF THE REWARD FUNCTION

Environment		
Name	Parameter	Value
Pusher	w_α	5.0
	w_β	0.05
	w_1	0.5
	w_2	5.0
	$w_{\mathbf{a}}$	5×10^{-3}
	$w_{\mathbf{s}}$	5.0
Door Opening	w_α	5.0
	w_β	0.05
	w_1	0.5
	w_2	2.5
	$w_{\mathbf{a}}$	2.5×10^{-4}
	$w_{\mathbf{s}}$	0.5
Peg Insertion	w_α	$5.0 \mid 0.$
	w_β	0.05
	w_1	1.0
	w_2	1.0
	$w_{\mathbf{a}}$	5×10^{-4}

Table II
HYPERPARAMETERS FOR ALL EXPERIMENTS

HyperParameters	Value
Policy network	2 hidden layers with 64 neurons
Discount rate (γ)	0.99
GAE λ	0.98
Meta learning rate (β)	0.001
Update steps per meta-iteration (K)	500
Trajectories sampled per task	20
Clipping parameter (ϵ)	0.2

2) Peg Insertion: The final environment comes from previous work in [30]. The tasks are more complex and involve manipulating a 7-DoF arm to insert a tight-fitting peg into a hole. The location of the hole is sampled uniformly from a 20 cm $\times$ 20 cm region. The full state observations include joint angles, velocities, and the position of the end-effector relative to the hole, but not hole position. For this environment, we employ the reward function

$$R = -w_1 \ell^1\left(\mathbf{z}_1, \mathbf{z}_2\right) - w_2 \ell^2\left(\ .\ , \mathbf{z}_3\right) - w_{\mathbf{a}} \left\|\mathbf{a}\right\|^2 \tag{15}$$

$$\ell = \begin{cases} \ell^1 & \text{if } w_\alpha = 5.0 \quad w_\beta = 0.05 \\ \ell^2 & \text{if } w_\alpha = 0. \quad w_\beta = 0.05 \end{cases} \qquad \mathbf{z}_1 = \mathbf{z}_2 = \mathbf{d} \tag{16}$$

$$\mathbf{z}_3 = 10 \times \mathbf{d} \times \mathbb{1}_{t=H-1} \tag{17}$$

where $\mathbf{d}$ is the distance between the hole and the peg and $\mathbb{1}$ is an indicator function. The parameters of the reward function for each environment are shown in Table I. The hyperparameters for all experiments using SMRL-TO are shown in Table II.

B. Comparisons

We compare our algorithm SMRL-TO with previous methods, including MAML [15], PEARL [16] and GMPS [17].

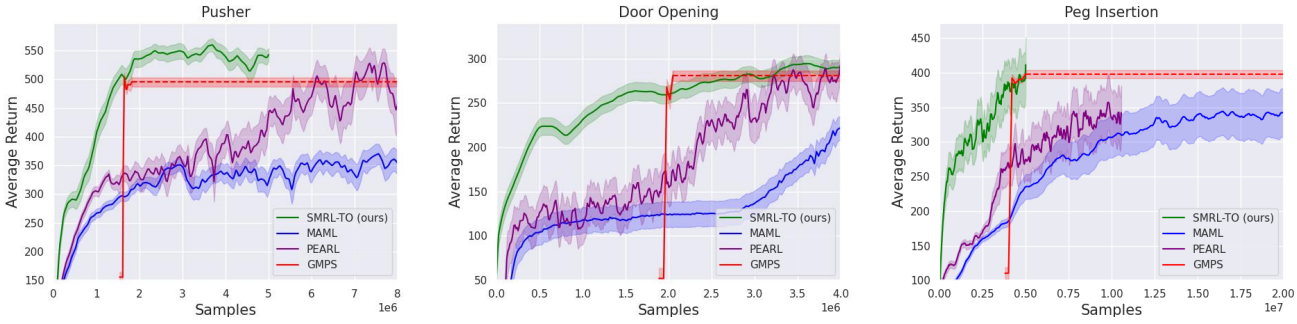

Figure 3. Meta-training sample efficiency comparisons on Pusher (left), Door Opening (middle) and Peg Insertion (right). Compared with MAML and PEARL, SMRL-TO is able to achieve higher average return in the first few iterations and continues to improve over iteration, resulting in better performance. While similar asymptotic performance with GMPS at the end of meta-training, SMRL-TO can adapt to new tasks robustly during meta-testing (analysis in Section V-B). The curve of GMPS does not indicate the part of training expert policies, but the number of samples used during the period is included. The red dotted line indicates the final result.

MAML is an on-policy meta-RL algorithm that first computes adapted parameters via a vanilla policy gradient and then performs meta-optimization with TRPO [48]. Compared with MAML, PEARL is an off-policy actor-critic meta-RL algorithm, which learns a probabilistic encoder of tasks. GMPS is an off-policy meta-RL algorithm that requires previously learned expert policies or human-provided demonstrations of all meta-training tasks and uses imitation learning to train the meta-level policy. To reduce the affects of stochastics, we re-run three repetitions for each algorithm with random initializations at the meta-training and meta-testing.

In the meta-training phase, we evaluate the algorithms in different ways according to their features. When evaluating MAML, PEARL and SMRL-TO, in each meta-iteration of the meta-training phase, we drawn batch tasks (30 in Pusher, 20 in Door Opening, 20 in Peg Insertion) from distribution $\rho(\mathcal{T})$ as training tasks to optimize the meta-level policy and then drawn another new batch tasks (40 in Pusher, 40 in Door Opening, 20 in Peg Insertion) from $\rho(\mathcal{T})$ as testing tasks to evaluate the trained meta-level policy. Unlike the above three algorithms, GMPS is a two-stage method and needs to train the policy of each meta-training task with a large number of samples to obtain expert trajectories. For example, for one meta-training task in Pusher environment, GMPS requires about 0.8 million samples to train the expert policy. It is impossible for GMPS to train expert policies for all tasks drawn from task-distribution $\rho(\mathcal{T})$ in the whole meta-training. However, our proposed method only needs about 2 million samples to obtain the meta-level policy in the whole meta-training stage, which is far less than the amount of samples used by GMPS. Therefore, it is impractical to evaluate the GMPS algorithm according to the above evaluation method. For better comparison with the above three algorithms in sample efficiency, GMPS, in this work, only involves 2 specific tasks in the whole meta-training phase. That is, we train and evaluate GMPS on these 2 tasks in each meta-iteration. In this way, the GMPS method uses as many samples as our proposed SMRL-TO algorithm in the meta-training phase. We can easily compare the ability of meta-level policy trained by GMPS and SMRL-TO to adapt to new tasks. We employ soft actor-critic (SAC) [49] algorithm for GMPS to learn the optimal policies of these two tasks and the samples used for training are included in experimental evaluation.

We first evaluate SMRL-TO for sample efficiency on robotic manipulations. Figure 3 presents the meta-training comparisons in a series of robotic manipulation tasks. Learning curves are averaged over three replicas. The error-bars indicate the standard error across three different seeds. With the aid of iLQR-based trajectory optimization, SMRL-TO achieves improvements in sample efficiency, particularly in Pusher and Peg Insertion. In Door Opening, SMRL-TO achieves similar asymptotic performance with PEARL at the end of meta-training, although higher average return than MAML and PEARL at the beginning. It means that SMRL-TO can make it easy for the end-effector to locate the position of the door handle, but is not superior to PEARL in opening the door to the target angle. We analyze that under linear-Gaussian dynamics, it is relatively difficult to optimize the local policy for the latter process. MAML algorithm based on policy search easily falls into local optimum and requires numerous samples to escape from it in these tasks, while SMRL-TO is not. Compared with MAML, PEARL improves sample efficiency by learning an encoder of tasks. However, PEARL exploits the policy search method that cannot provide more direct supervision like imitation learning, resulting in slow convergence and high sample complexity. The comparisons have proved that using the online supervision from trajectory optimization instead of simply focusing direct policy search on trial-and-error data, does contribute to the learning of meta-level policy, hence SMRL-TO can meta-learn efficiently.

SMRL-TO performs well and presents similar asymptotic performance with GMPS at the meta-training. However, the task space $\rho(\mathcal{T})$ in GMPS is reduced to two tasks in the training and evaluation, which results in overfitting nicely and does not adapt to new tasks at the meta-testing. This means that when the same number of samples are used at the meta-training, SMRL-TO can robustly adapt to new tasks, while GMPS cannot. GMPS must require more samples at the meta-training if it can adapt to new tasks. Our proposed SMRL-TO avoids the use of global expert policy per task like GMPS, and then achieves the goal of high sample efficiency. It should be noted that when performing GMPS algorithm

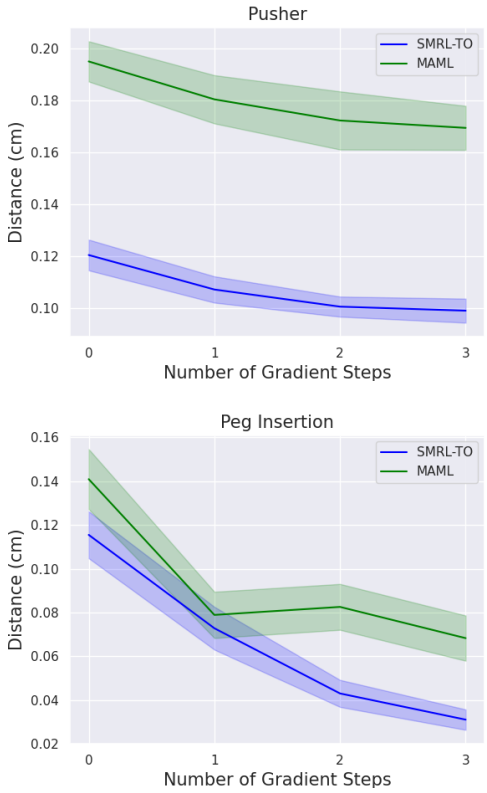

Figure 4. Meta-testing comparison between SMRL-TO and MAML for Pusher (up) and Peg Insertion (down). Learning curves are averaged over three replicas. The error-bars indicate the standard error. For Pusher, we plot the distance between the red block and the target position at last time step, while for Peg Insertion we plot the the distance between the peg and the hole at last time step. Both SMRL-TO and MAML can adapt their behaviours to new tasks by just few gradient steps, but SMRL-TO achieves smaller distance at the end of meta-testing.

in each task, we intercept the learning curve (red solid line) of the first few iterations, and use the red dotted line to indicate the final result. Because GMPS algorithm suffers from instability in the next iterations. It is that due to distribution mismatch, states from the adapted policy can not be correctly labeled with optimal actions from previously learned expert policies. Thus bad data increases over iteration, resulting in an oscillating and diverging behavior. Compared with GMPS, our proposed algorithm SMRL-TO including off-policy RL setting, can meta-learn stably without any demonstrations by leveraging online supervision from trajectory optimization.

In meta-testing, both MAML and our SMRL-TO adapt to new tasks by vanilla policy gradient. From the curves in Figure 4, we observe that SMRL-TO can quickly adapt to new tasks and achieve better performance than MAML by just few gradient updates. This shows that our approach can train a reinforcement learner more efficiently. We report the numerical results of meta-testing in Tables III and IV.

C. Ablation Study

In this section, we conduct an ablation study to evaluate the contribution of each component of our proposed approach. We compare our algorithm with its three ablated variants on Peg-Insertion. The first variant uses fixed rather than meta-learned

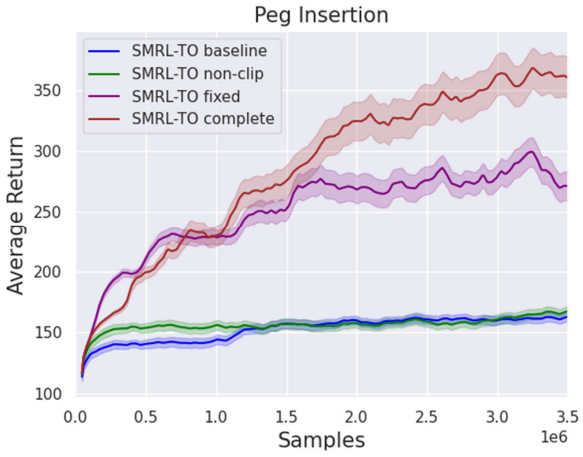

Figure 5. Comparison of our approach with ablated variants for Peg Insertion. Learning curves are averaged over three replicas. The error-bars indicate the standard error. The plot shows the results with meta-learned α and with clipping trick ("SMRL-TO complete"), with meta-learned α and without clipping trick ("SMRL-TO non-clip"), without meta-learned α and with clipping trick ("SMRL-TO fixed"), as well as without meta-learned α and without clipping trick ("SMRL-TO baseline"). Our method ("SMRL-TO complete") presents the best performance.

fast adaptation learning rate (α) in the inner loop, which we label "SMRL-TO fixed". The second variant, referred to as "SMRL-TO non-clip", only removes clipping trick on the basis of our complete algorithm "SMRL-TO complete" and uses vanilla importance-weighted policy gradient in the inner loop. The third variant uses fixed fast adaptation learning rate and removes clipping trick at the same time, which we label "SMRL-TO baseline". To reduce the affects of stochastics, we also re-run three repetitions for each ablated variant with random initializations.

The results are shown in Figure 5 in terms of the average return of 30 meta-testing tasks at training samples. Our proposed algorithm "SMRL-TO complete" uses both meta-learned fast adaptation learning rate and the proposed clipping trick. The comparison to the "SMRL-TO baseline" variant shows that both of the above components are crucial for obtaining the good overall performance. In addition, we interestingly observed that the "SMRL-TO baseline" and "SMRL-TO non-clip" variants show similar performance in the experiment. This means that using the meta-learned learning rate alone can not effectively solve the instability problem introduced by the off-policy optimization, while it can contribute the performance at the aid of clipping trick, as shown in comparison of "SMRL-TO fixed" and "SMRL-TO complete" variants. It is proved that our proposed clipping trick can alleviate the instability problem and is a particularly important component of our algorithm.

VI. Discussion and Future Work

In this paper, we introduced a novel meta-RL algorithm called SMRL-TO, which can meta-learn more efficiently and stably with the aid of iLQR-based trajectory optimization. Compared with the existing meta-RL methods, our approach has several benefits. Through trajectory optimization, it can acquire more information from a small amount of on-policy

Table III
PUSHER, DISTANCE (CM)

num. grad steps	0	1	2	3
MAML	0.1950 ± 0.0078	0.1804 ± 0.0093	0.1722 ± 0.0112	0.1694 ± 0.0085
SMRL-TO (ours)	$\mathbf{0.1204} \pm 0.0059$	$\mathbf{0.1072} \pm 0.0051$	$\mathbf{0.1006} \pm 0.0039$	$\mathbf{0.0990} \pm 0.0046$

Table IV
PEG INSERTION, DISTANCE (CM)

num. grad steps	0	1	2	3
MAML	0.1409 ± 0.0137	0.0789 ± 0.0106	0.0825 ± 0.0105	0.0682 ± 0.0103
SMRL-TO (ours)	$\mathbf{0.1154} \pm 0.0107$	$\mathbf{0.0728} \pm 0.0098$	$\mathbf{0.0430} \pm 0.0062$	$\mathbf{0.0309} \pm 0.0047$

data for policy learning. It makes policy learning more robust and efficient through imitation learning rather than direct policy search. More importantly, due to the small sample complexity, our approach is likely to solve tasks in real-world situations without human-provided demonstrations. So our next step would be to apply our method to physical robotic systems. In effect, our work has actually given a paradigm that combines MAML algorithm with local optimization method to speed up training of reinforcement-learner. While putting less the pressure on needing perfect supervision, our approach is also affected by the choice of policy optimization method.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No. 61973066, 61471110), Foundation of Key Laboratory of Equipment Reliability (WD2C20205500306).

REFERENCES

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] D. Ye, G. Chen, W. Zhang, S. Chen, B. Yuan, B. Liu, J. Chen, Z. Liu, F. Qiu, H. Yu *et al.*, "Towards playing full moba games with deep reinforcement learning," *arXiv preprint arXiv:2011.12692*, 2020.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] Y. Lin, M. Wang, X. Zhou, G. Ding, and S. Mao, "Dynamic spectrum interaction of uav flight formation communication with priority: A deep reinforcement learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 3, pp. 892–903, 2020.

[7] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[8] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation*. IEEE, 2016, pp. 528–535.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[10] M. Ramicic and A. Bonarini, "Augmented memory replay in reinforcement learning with continuous control," *IEEE Transactions on Cognitive and Developmental Systems*, 2021.

[11] F. Zeng, C. Wang, and S. S. Ge, "Tutor-guided interior navigation with deep reinforcement learning," *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[12] X. Huang, W. Wu, and H. Qiao, "Computational modeling of emotion-motivated decisions for continuous control of mobile robots," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 1, pp. 31–44, 2020.

[13] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Advances in Neural Information Processing Systems*, 2018, pp. 5302–5311.

[14] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RI^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning-Volume 70*, 2017, pp. 1126–1135.

[16] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[17] R. Mendonca, A. Gupta, R. Kralev, P. Abbeel, S. Levine, and C. Finn, "Guided meta-policy search," in *Advances in Neural Information Processing Systems*, 2019, pp. 9653–9664.

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

[19] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[20] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[21] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[22] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever, "Some considerations on learning to explore via meta-reinforcement learning," *arXiv preprint arXiv:1803.01118*, 2018.

[23] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[24] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "Promp: Proximal meta-policy search," *arXiv preprint arXiv:1810.06784*, 2018.

[25] Z. Xu, H. P. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018.

[26] R. Houthooft, R. Y. Chen, P. Isola, B. C. Stadie, F. Wolski, J. Ho, and P. Abbeel, "Evolved policy gradients," 2018.

[27] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on robot learning*. PMLR, 2017, pp. 357–368.

[28] D. Rengarajan, S. Chaudhary, J. Kim, D. Kalathil, and S. Shakkottai, "Enhanced meta reinforcement learning using demonstrations in sparse reward environments," *arXiv preprint arXiv:2209.13048*, 2022.
[29] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.
[30] W. H. Montgomery and S. Levine, "Guided policy search via approximate mirror descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 4008–4016.
[31] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 156–163.
[32] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, 2014, pp. 829–837.
[33] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.
[34] S. Levine, *Motor skill learning with local trajectory methods*. Stanford University, 2014.
[35] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning." *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.
[36] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010, pp. 1607–1612.
[37] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004, vol. 133.
[38] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.
[39] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," *Advances in Neural Information Processing Systems*, vol. 27, 2014.
[40] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *International Conference on machine learning*, 2011, pp. 465–472.
[41] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 703–711.
[42] S. Levine, "Optimal control and planning," http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_8_model_based_planning.pdf, pp. 27–37, 2017.
[43] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[45] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.
[46] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
[47] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1094–1100.
[48] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.
[49] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.