Bayesian Meta-Learning for Few-Shot Policy Adaptation Across Robotic Platforms

Ali Ghadirzadeh*, Xi Chen*, Petra Poklukar1, Chelsea Finn2, Mårten Björkman1 and Danica Kragic

Abstract—Reinforcement learning methods can achieve significant performance but require a large amount of training data collected on the same robotic platform. A policy trained with expensive data is rendered useless after making even a minor change to the robot hardware. In this paper, we address the challenging problem of adapting a policy, trained to perform a task, to a novel robotic hardware platform given only few demonstrations of robot motion trajectories on the target robot. We formulate it as a few-shot meta-learning problem where the goal is to find a meta-model that captures the common structure shared across different robotic platforms such that data-efficient adaptation can be performed. We achieve such adaptation by introducing a learning framework consisting of a probabilistic gradient-based meta-learning algorithm that models the uncertainty arising from the few-shot setting with a low-dimensional latent variable. We experimentally evaluate our framework on a simulated reaching and a real-robot picking task using 400 simulated robots generated by varying the physical parameters of an existing set of robotic platforms. Our results show that the proposed method can successfully adapt a trained policy to different robotic platforms with novel physical parameters and the superiority of our meta-learning algorithm compared to state-of-the-art methods for the introduced few-shot policy adaptation problem.

I. INTRODUCTION

Robot learning has emerged as a field in robotics with the goal of developing data-driven algorithms that can deal with the complexity and the diversity of robotic problems. The learning algorithms today achieve impressive results but mostly on a fixed robotic hardware, and to account for a minor change in the robot hardware the policy has to be retrained from scratch. Training samples are usually quite expensive in such setups and in most cases retraining policies from scratch is not data-efficient. This problem becomes more profound when learning complex visuomotor skills which typically require a significant amount of data. Therefore, the ability to adapt a policy to changes in the robot hardware using few training samples is an important property of a learning system. Likewise, sample efficiency of robot learning algorithms will be significantly improved if we can train a policy for a task on a robotic platform once and adapt it to all other similar robots with only few training samples. However, training general policies that adapt efficiently to various robots with different morphologies is a challenging problem for existing data-driven methods. Even though generic learning algorithms can train policies to acquire a wide range of manipulation skills [1–3] there is no way to transfer a trained policy to a new robot despite the common structure that is present in different robotic platforms.

Platform-independent policy training can be formulated as a few-shot learning problem where the goal is to adapt an action-selection policy to a new platform from small amounts of robot data [4], [5]. Such a few-shot learning problem can be addressed by meta-learning [6] - a learning to learn framework that leverages past knowledge to solve novel tasks more efficiently. In our case, a meta-learning agent can be used to learn a representation of the common structure of motor commands for a given manipulation task across different robotic platforms. However, ambiguities arising from the limited data make it difficult to find a unique representation of a robotic platform. Standard meta-learning approaches yield a single solution that is consistent with the given data but might be sub-optimal when the data do not carry sufficient information about the robotic platform. We therefore introduce a probabilistic meta-learning framework that models this uncertainty with a probability distribution from which multiple possible platform representations can be sampled. Our meta-learning framework is similar to [7] but instead models the uncertainty over the meta-learner using a low-dimensional latent variable. The low dimensionality of the introduced latent variable allows practitioners to readily interpret the behavior of the trained models.

The main contributions of this paper are: (1) introducing a learning framework to adapt a trained policy to a modified robot hardware or a novel but similar robotic platform, (2) introducing a probabilistic meta-learning framework that

1KTH Royal Institute of Technology, 2Stanford University [algh][xi8][poklukar][celle][dani]@kth.se, cbfinn@cs.stanford.edu
* denotes equal contribution.
proposes multiple potential action-selection policies by assigning a distribution over a low-dimensional latent variable given few demonstrated motion trajectories, and (3) experimentally evaluating the framework and benchmarking it against some meta-learning algorithms to adapt a trained policy to novel robotic platforms for a simulated reaching and a real-robot picking task. We experimentally evaluate our framework on 400 robots in simulation which are generated by modifying four existing robot platforms, namely ABB YuMi, Baxter, Franka Emika and Kinova robots. By training and evaluating a meta-learner on this setup, we show how a trained policy can be adapted to a new robot by demonstrating only a few motion trajectories. Our experimental results demonstrate superior performance compared to the prior meta-learning algorithms [6], [8] for a robotic reaching task on different target robots.

II. RELATED WORK

Transfer learning in robotics: Few-shot transfer learning has in recent years gained in popularity in robotics research because it addresses both the sample-inefficiency problem of reinforcement learning (RL) algorithms and high costs of recollecting real robot data for every new manipulation task. Prior work focused broadly on the simulation-to-real transfer learning problem [9–12], few-shot imitation learning [13–18], transfer of perception models across different domains [19–21], transfer of skills across tasks [14], [22], [23], dynamics [5], [24], and nonstationary environments [25]. However, limited work has been done on transfer learning across robotic platforms [26–30].

Devin et al. [26] proposed to decompose a policy network into robot-specific and task-specific modules to facilitate the transfer of knowledge to novel combinations of pre-trained tasks and robots. We similarly decompose a policy into task- and robot-specific components but in contrast to [26] our goal is to transfer manipulation skills to a novel robotic platform in a few-shot manner. Chen et al. [30] and Schaff et al. [29] proposed to learn a policy conditioned on an encoding of a robotic hardware, such as the kinematic structure of the robot. Their goal is to learn a general policy that transfers to a new robot either in a zero-shot manner or via fine-tuning [30] and optimizing the robotic hardware together with the policy at the same time [29]. We also learn a robot encoding but instead of feeding it to the policy we use the encoding to initialize a network that is further updated by gradient descent. Huang et al. [28] suggested to train one policy that control each limb of an agent and coordinate between different limbs through a message passing technique. Even though their approach demonstrated appealing results for walking gaits, it is not clear how to apply it to visuomotor policy training where perception and control are trained together. Dasari et al. [27] introduced a large-scale multi-task and multi-robot dynamic model training that can be used with model-based action-selection approaches such as model-predictive control [31]. We train policies given large-scale datasets constructed in simulation by adjusting physical parameters of different robotic platforms. However, we aim to meta-train an action-selection policy that can then be efficiently adapted to a novel robot given only few motion trajectories from the target robot.

Meta-learning: Meta-learning is a common approach to solve few-shot learning problems [32–35]. A popular optimisation-based meta-learning method is the model-agnostic meta learning (MAML) framework [6]. However, a critical issue in the few-shot learning setup is the lack of information contained in the few-shot training data to properly solve a given new task [7]. To account for such task ambiguities different probabilistic frameworks based on MAML [7], [36–38] were introduced as well as a variety of other Bayesian meta learning frameworks [8], [39–42]. Similar to [8], [38], [41], we introduce a low-dimensional meta-task latent variable which we embed into the gradient-based meta-learner following MAML. This combination improves the expressive power of the meta-learner while still enables generation of multiple solutions to a given few-shot adaptation task.

III. PRELIMINARIES

Policy training with generative models: Similar to [43], we consider a finite-horizon Markov decision process defined by a tuple $(S, U, P, p(s_g), p(r|s_g, \tau))$, where $S$ denotes both the set of (terminal) states $s$ and the set of goal states $s_g$, $U$ is a set of motor actions $u_m$ for the robot motor $m$, $P : U \times S \to \mathbb{R}$ is the state transition probability assuming a fixed initial state, $p(s_g)$ is the goal state distribution and $p(r|s_g, \tau)$ is the probability of the reward $	au$ conditioned on a goal state $s_g$ and a fixed-length sequence of open-loop motor actions $\tau = \{u_{t,m}\}$ at time index $t$. In order to complete a robotic manipulation task given a goal state $s_g' \sim p(s_g)$, we wish to find a goal-conditioned policy $\pi_{\Theta}(\tau|s_g')$, parameterized by $\Theta$, that assigns a distribution over the possible motion trajectories $\tau$ that maximize the expected reward $p(r|s_g', \tau)$. We follow [3], [43] and find the parameters $\Theta$ by first introducing an action latent variable $\alpha$ and a trajectory generative model $p_{\theta}(\tau|\alpha)$, parametrized by $\theta$, that maps a latent action sample $\alpha'$ into a full motion trajectory $\tau$. Given $\theta$, the policy parameters $\Theta = \{\theta, \theta\}$ are then found by marginalizing over $\alpha$ and maximizing the expected reward

$$J = \max_{\Theta} \mathbb{E}_{s_g' \sim p(\cdot), \alpha' \sim \pi_{\theta}(\cdot|s_g'), \tau' \sim p_{\alpha}(\cdot|\alpha')[p(r|s_g', \tau')]},$$

(1)

where a sub-policy $\pi_{\theta}(\alpha|s_g')$, parametrized by $\theta$, assigns a distribution over the action latent variable $\alpha$ conditioned on the goal state $s_g'$. Intuitively, the action latent variable $\alpha$ captures the high-level objective of the policy $\pi_{\Theta}$, while the generative model $p_{\theta}$ translates this objective into the low-level motor commands.

The complete policy network is illustrated in Figure 2 with blue color. A goal state $s_g'$ is first mapped into a latent action sample $\alpha'$ by the sub-policy $\pi_{\theta}$ and then to a sequence of motor actions $\tau'$ by the generative model $p_{\theta}$. The two parametric models $\pi_{\theta}$ and $p_{\theta}$ together form the policy $\pi_{\Theta}$.

The sub-policy $\pi_{\theta}$ can be trained independently of the robotic hardware platform since the structure of $\alpha$ can be fixed such
that the same sub-policy can be used for all robotic platforms without any model adaptation (Sec. [IV-C]). However, the generative model \( p_\theta \) generates platform-dependent low-level motor actions which requires the parameters \( \theta \) to be adapted to different platforms. The generative model \( p_\theta \) can be trained on a dataset consisting of trajectories \( \tau \) demonstrated on a given robotic platform based on variational autoencoders (VAEs) [3], [44] which maximize the following variational lower bound

\[
\max_{\alpha'} \mathbb{E}_{\alpha' \sim q_\psi(\alpha)} \left[ \log p_\theta(\tau | \alpha') - D_{KL}(q_\psi(\alpha | \tau) || p(\alpha)) \right],
\]

where \( q_\psi(\alpha | \tau) \) denotes the approximate posterior distribution and \( p(\alpha) \) the prior over the action latent variable which is set to the standard normal distribution.

**Multi-task meta learning:** Meta learning can be used to model the common structure shared among different robotic platforms in a way that enables few-shot transfer to related tasks. Let a task \( T_i \) be defined as the task of generating valid motion trajectories for the \( i \)th robotic platform. Let \( p(T) \) be an unknown distribution from which we sample infinitely many tasks \( T_i \sim p(T) \) each of which is represented by a dataset \( D_{T_i} \) consisting of motor action trajectories demonstrated on the platform \( i \). A common representation of the tasks \( \{ T_i \} \) can be learned by training a meta-model using the MAML framework. MAML learns a common feature representation or network initialization to achieve an optimal solution on a novel task with only a small number of gradient steps. MAML consists of (a) a meta-train phase in which the meta-model is trained, and (b) adaptation or meta-test phase in which the meta-model is adapted to a novel task using gradient descent. The meta-model \( \phi \) in phase (a) is trained by optimizing the following objective

\[
\min_\phi \sum_{T_i \sim p(T)} \mathcal{L}(\phi - \lambda \nabla_\phi \mathcal{L}(\phi, D_{T_i}^{\text{tr}}, D_{T_i}^{\text{ts}})),
\]

where \( \lambda \) is the learning rate which itself is a trainable parameter and \( D_{T_i}^{\text{tr}}, D_{T_i}^{\text{ts}} \) are the observed support set and the hidden query set, respectively, such that their union forms the entire task \( T_i \) dataset \( D_i = D_{T_i}^{\text{tr}} \cup D_{T_i}^{\text{ts}} \).

**IV. FEW-SHOT POLICY TRAINING BASED ON META-LEARNING**

In this section, we introduce our probabilistic meta-learning algorithm that outputs a probability distribution over the meta-model parameters. Using this distribution, we sample the parameters of the trajectory generative models that are used to model a given set of few demonstrated trajectories. We formally define the model setup in Sec. [IV-A] and introduce the framework in Sec. [IV-B]. Finally, we explain the training of the sub-policy \( \pi_\alpha(\alpha | s_q) \) in Sec. [IV-C].

**A. Platform-independent policy training setup**

Our goal is to devise a meta-learning algorithm that adapts the policy \( \pi_\alpha \), more specifically, the trajectory generative model \( p_\alpha(\tau | \alpha) \), shown in Figure 2 to an unseen robotic platform, i.e., a novel meta-task \( T_{\text{novel}} \sim p(T) \) in a few-shot manner given the set of support motion trajectories \( D_{T_{\text{train}}}^i \). The meta-model is trained on meta-train datasets \( D_i \) containing valid motion trajectories to run on the \( i \)th robot. The datasets are divided into support and query sets contain support motion trajectories \( \tau_{i}^{\text{tr}} \in D_{T_{i}}^{\text{tr}} \) and query motion trajectories \( \tau_{i}^{\text{ts}} \in D_{T_{i}}^{\text{ts}} \).

The platforms are generated in simulation by choosing a robot model and randomly adjusting the length of different robot links. Using existing motion planners, such as rapidly-exploring random tree (RRT) [45], we then generate several motion trajectories \( \tau \) that complete the desired goal in simulation. The generated trajectories \( \tau \) and the label of the robotic platform \( i \) form one meta-task dataset \( D_i \) while their union over all meta-tasks forms the meta-train dataset \( \{ D_i \} \).

**B. Our proposed probabilistic meta-learning framework**

Since the small size of a dataset representing a new meta-task gives rise to ambiguities in the meta-model parameters, we extend MAML to a probabilistic setting. The idea of our probabilistic meta-learning algorithm is to represent meta-tasks in a low-dimensional latent space. We obtain such low dimensional representation of the meta-tasks by introducing a meta-task latent variable \( z \) and a network generative model \( p_\psi(\phi | z) \), parametrized by \( \psi \), that generates the high dimensional meta-parameters \( \phi \) from a given representation \( z \). Therefore, using the meta-task latent variable \( z \) we can first generate different initial meta-models that are well-suited for the few-shot gradient descent updates and then choose the one that results in the best performance on the given novel platform. The variable \( z \) is sampled from a variational distribution \( q_\psi(z | \tau) \), parametrized by \( \psi \), which is trained jointly with the network generative model \( p_\psi(\phi | z) \) using the meta-train dataset \( \{ D_i \} \) as explained next. Note that we choose \( q_\psi \) to be a Gaussian distribution.

During the meta-train phase (a) the goal is to find the parameters of the variational distribution \( q_\psi \) and network generative model \( p_\psi \) that yield a distribution over the possible meta-model parameters \( \phi \). By sampling from this distribution, different platform-dependent trajectory generative models \( p_\phi \) are generated that can be efficiently optimized similar to MAML by one or few gradient updates. More precisely, \( \psi \) and \( \phi \) are optimized by maximizing the following
variational lower bound \[44\]

\[
\max_{\varphi, \psi} \sum_i \mathbb{E}_{z_i \sim q_\psi(\cdot|\tau^t_i), \varphi_i \sim p_\varphi(\cdot|z_i)} \left[ \log p_\varphi(\tau^t_i) \right] - \beta D_{KL}(q_\psi(z|\tau^t_i)||p(z)),
\]

(4)

where, \(p(z)\) is the prior over the meta-task latent variable \(z\) and \(\varphi_i = \varphi_0 - \lambda \nabla_{\varphi} \mathcal{L}(\varphi, D^T_i)\) is the gradient step as in the MAML adaptation step and \(\beta\) is a scalar parameter that balances the two loss terms. Note that we omitted the action latent variable \(\alpha\) from the notation \(p_\varphi(\cdot|\cdot)\) in Eq. (4) for the sake of clarity. Please refer to Sec. [IV-C] for more details.

During the meta-test phase (b), illustrated in Fig. [2] with green color, a few motor action trajectories from a new robotic platform are given to the algorithm which then returns a distribution over the parameters of the trajectory generative model \(\vartheta\). More specifically, the motor action trajectories \(\tau' \in D^T_{\text{novel}}\) are first recorded given some demonstrations. The trajectories are then given to the variational distribution \(q_{\varphi_0}(\cdot|\tau')\) from which we sample several different latent task representations \(\{z_j \sim q_\psi(\cdot|\tau')\}\). These are further mapped to the meta-parameters \(\{\varphi_j\}\) by the network generative model \(p_\varphi\). Finally, a set of task parameters \(\{\varphi_j\}\) is found by applying the gradient descent rule on the loss functions \(\mathcal{L}(\varphi_j, D^T_{\text{novel}})\).

C. Training the sub-policy

In this section, we explain how we can train a single sub-policy \(\pi_\theta(\alpha|s_g)\) that can be used with different trajectory generative models on different robotic platforms. The sub-policy is trained on only one robot and the training is performed independently and prior to the training the meta-model in phase (a).

The training of the sub-policy \(\pi_\theta(\alpha|s_g)\) as described in [43] requires a trained generative model \(p_\varphi(\tau|\alpha)\) which in turn requires a training dataset of the motor trajectories. We therefore choose one robotic platform \(i_s\) for which we can obtain a training dataset of motor action trajectories \(D_{T_{i_s}}\). We train a VAE on \(D_{T_{i_s}}\) by optimizing Eq. (2).

The obtained generative model \(p_{\theta_{i_s}}\) is only used to train the sub-policy \(\pi_\theta(\alpha|s_g)\) with the Expectation-Maximization algorithm introduced in [43].

However, to be able to use the obtained sub-policy with different trajectory generative models on different robotic platforms, we need to ensure a consistent structure of the action latent variable \(\alpha\). We define a set of trajectories \(\{\tau^t_i\}\) to be consistent, if they result in the same end state \(s\) after executing them on the respective robotic platform. For a set of consistent trajectories, the goal is then to obtain the same latent action sample \(\alpha'\) for all robotic platforms. This is achieved using \(D_{T_{i_s}}\) and the previously trained VAE encoder \(q_{\psi_{i_s}}\) introduced in Eq. (2). In particular, a given trajectory \(\tau'\), regardless the robotic platform, is matched with a consistent trajectory \(\tau^t_i \in D_{T_{i_s}}\) by comparing the end states. The latter is mapped to a latent action sample \(\alpha' \sim q_{\psi_{i_s}}(\alpha|\tau^t_i)\). The resulting \(\alpha'\) and considered trajectory \(\tau'\) can then be used to train the meta-model in Eq. (4).

V. Experiments

We apply our proposed learning framework to a robotic reaching task in which the end-effector of the robot has to reach to different 3D target positions, and a visuomotor picking task similar to [21] in which a YuMi robot is trained to pick different objects in cluttered background given image pixels as the input. We experimentally evaluate (1) the generality of our learned meta-model by training it on three different scenarios described below, (2) the benefits of the probabilistic formulation of our method by benchmarking it to MAML, and (3) the benefits of gradient-based optimization by benchmarking it to two gradient-free methods described in Sec. [V-B]. We experimentally evaluate the generalizability of our meta-learning method when adapting the policy to (i) a slightly modified robotic hardware generated by changing the length of a link of the robot, and to (ii) a completely new robotic platform which is not used during the meta-training phase. We created 400 different robots in simulation by adjusting the mechanical parameters of four 7 degree-of-freedom robotic platforms, ABB YuMi, Kinova, Franka Emika and Baxter, shown in Figure [1]. We consider three different scenarios: (a) meta-training given data from robots built on one of the platforms (100 robots), and adapting to novel robots built on the same platform, (b) meta-training given all of the 400 simulated robots and adapting as in (a), and (c) meta-training given data from robots built on only three platforms (300 robots) and adapting to novel robots built on the excluded platform. The scenarios (a) and (b) are used to evaluate criteria (i), while scenario (c) is used to evaluate criteria (ii).

At meta-test time, we sample \(n = 3\) novel robots for each platform introduced in Figure [1]. We used 5 demonstrated motion trajectories to adapt the trajectory generative model \(p_{\theta_{i_s}}\), produced by the meta-learning agent, to the platform \(n\). The demonstrated trajectories are consistent across platforms as defined in Sec. [IV-C]. We evaluate the performance of the obtained policy \(\pi_{\theta_0}\) on the novel robot \(n\) using Eq. (1).

A. Network architectures

In this section, we introduce the network architectures of the sub-policy, the trajectory generative model, and our proposed meta-learner model. Following [21], the sub-policy \(\pi_\theta(\alpha|s)\) consists of four convolutional layers with \([64, 32, 32, 1]\) channels and \([5 \times 5]\) filters followed by three fully connected dense layers with \([1058 \times 16, 16 \times 16, 16 \times 6]\) neurons in each layer. For the picking task, the sub-policy maps an image of the size \(60 \times 105\), and for the reaching task a 2D target position into a 6D action latent variable \(\alpha\). In the latter case, the convolutional layers are excluded. The generative model \(p_{\theta_0}(\tau|\alpha)\), consisting of two dense layers with \([6 \times 8, 8 \times 98]\) neurons, then maps the 6D action latent variable into a trajectory of motor actions of the size \(7 \times 14\), i.e., 7 motor actions over 14 time-steps. The motor actions are joint level position commands sent to a position controller.

Our proposed meta-model consists of an encoder \(q_{\psi}(z|\tau)\) and a network generative model \(p_{\varphi}(\cdot|z)\). The encoder \(q_{\psi}\)
maps 5 demonstrated motor trajectories of the size $7 \times 14$ into the 2D task latent variable $z$ using 4 dense layers with $[490 \times 128, 128 \times 64, 64 \times 32, 32 \times 2]$ neurons, respectively. The network generative model $p_{\phi}$ consists of 5 dense layers of size $[2 \times 32, 32 \times 64, 64 \times 128, 128 \times 256, 256 \times 938]$. It generates the parameters of the meta-model neural network (having 832 weights and 106 biases) given the 2D task latent variable $z$ as the input. The meta-model has the same architecture as the generative model, and is adapted to a platform-specific generative model after one or few gradient updates. The Rectified Linear Unit (ReLU) is used as the non-linearity in our network architecture.

### B. Benchmarking methods

We benchmark our method with MAML [6], VERSA [8] and a method based on amortized variational inferences (AVI) [39] using three different meta-train and meta-test datasets constructed according to the three scenarios. These methods are used to adapt the parameters of the trajectory generative model given few demonstrated motion trajectories during the meta-test phase.

**MAML:** The meta-model in MAML is obtained with the same architecture as the trajectory generative model (introduced in Sec. V-A) and is adapted to a platform-specific generative model using one gradient update.

**VERSA:** We adopt the VERSA model used for few-shot view reconstruction in [8] to obtain the motor trajectory generative model. The meta-learner consists of an encoder $q_{e}(z|\tau)$ that similarly to our approach maps 5 motor trajectories into a 2D task latent variable. However, the task latent variable is directly given as an input to the trajectory generative model as opposed to the network generative models as in our method. Therefore, in VERSA, the parameters of the trajectory generative model are platform independent.

**AVI:** In order to study the benefit of the gradient descent component of our method, we exclude the gradient optimization step from our method. The meta-learner in this case consists of a task encoder and a network generative model.

For both VERSA and AVI, we use the same network architecture as described in Sec. V-A. We train all the methods, including ours, for 1000 epochs using ADAM optimizer [46] with learning rate $1e^{-4}$. For our method we use $\beta = 5e^{-3}$. All hyper-parameters are found by searching for optimal values.

### C. Evaluating the meta-learning agent

For our method, VERSA and AVI, we sampled $j = 20$ policies from the learned distribution based on the 5 motion trajectories. In Fig. 3 [4] we provide the average and 95% confidence interval of the reaching error on the obtained 20 policies for the robotic reaching task. For MAML we report the reaching error of the obtained single solution. Our experimental results find that our method outperforms MAML, VERSA, and AVI in most cases for all of the three scenarios. In scenario (a), in which the meta-train and meta-test datasets include data from only one platform, we observe that in most cases our method performs considerably better than the others, also with lower variance (Fig. 3). This experiment suggests that our few-shot learning approach performs well when adapting the policy to changes in the same robot hardware.

![Fig. 3: Scenario (a): The average and the 95% confidence interval of the reaching error (in cm) of the adapted policies for all methods. The meta-learner is trained and tested on the data generated by the same platform. Y stands for YuMi, B for Baxter, K for Kinova, and F for Franka Emika, while the numbers denote the index of the test robot.](image1)

![Fig. 4: Scenario (b): The average and 95% confidence interval of the reaching error as in Fig 3. The meta-learner is trained and tested on the data generated by all four platforms.](image2)

![Fig. 5: Scenario (c): The average and 95% confidence interval of the reaching error as in Fig 3. The meta-learner is trained on the data from all platforms except the one that is used for meta-testing.](image3)
achieves significant gains compared to prior methods on the YuMi and Kinova platforms. These results demonstrate the effectiveness of adaptation to a wider range of meta-tasks.

Next, we evaluate our method based on criteria (ii) using scenario (c), in which the meta-learner should adapt to out-of-distribution tasks (Fig. 5). We observe that the methods do not yield satisfactory performance. The results of our method are slightly better for the Baxter and YuMi robots, potentially due to the similarities between these two robots. To further analyze the results, we also studied the latent structure of our method for models trained in scenarios (b) and (c). In Fig. 6, we illustrate the mean values of the meta-task latent variable $z$ obtained from the Gaussian encoder $q_\psi$ corresponding to all the robots used to generate the meta-train datasets in scenarios (b) and (c). The left figure illustrates the latent space of a model trained on scenario (b). We see that our method well separates the four robotic platforms in the meta-task latent space. The right three figures in Fig. 6 illustrate the meta-task latent space of three independently trained models for scenario (c) in which Baxter robots are excluded from the meta-train phase. The figures also show the Baxter robots encoded at meta-test time. The results suggest that the Baxter robots (at meta-test) look similarly to the YuMi robots that were used to train the meta-learner, which is consistent with the results visualised in Fig. 5.

Beyond these visualizations, we add one new experiment to understand how we might mitigate the challenge of adapting to out-of-distribution tasks. In particular, to avoid meta-testing on completely out-of-distribution tasks as in scenario (c), we included 10 and 20 meta-task samples of the excluded platform in the meta-train dataset and repeated the experiment. The results (Fig. 7) are significantly better than the results reported for scenario (c), suggesting that a relatively small amount of meta-training data on these platforms is particularly helpful for alleviating out-of-distribution challenges.

Finally, we present our experimental results for the visuo-motor picking task on a real YuMi robot. First, we built 4 different robot grippers that are $-1$ CM, $0$ CM, $+1$ CM and $+2$ CM longer than the original YuMi robot gripper. Using our method, we meta-trained given 100 simulated robots as described in scenario (a) and meta-tested on the real YuMi robot with the four different grippers. On average, we obtained 90.8% picking success rate after 5-shot adaptation to the YuMi robot with the different gripper sizes. These results validate that our approach can scale to a real robot with image observations.

![Fig. 6: Visualization of the meta-task latent space. Left: meta-task latent samples corresponding to the training dataset of a model trained for scenario (b). Right: meta-task latent samples corresponding to the training dataset of a model trained for scenario (c) as well as to the testing dataset of the excluded robot (in red).](image)

**Fig. 6**

![Fig. 7: The average and 95% confidence interval of the reaching error as in Fig. 5. Our meta-learner is trained on the data from all platforms except the one used for meta-testing of which 0, 10 and 20 meta-task samples are used during meta-training.](image)

**Fig. 7**

The primarily goal of this study was to answer the following question: is it possible to obtain a meta-learner that captures the common structure shared across different robotic platforms such that an action-selection policy for performing a given robotic task can be adapted to a novel platform in both unambiguous and data-efficient manner? We approached this problem by introducing a MAML-based probabilistic meta-learning framework where the meta-tasks are represented by a low-dimensional meta-task latent variable. We exploited the latent meta-task variable for modeling the meta-task uncertainties originating from the few-shot formulation of the policy adaptation problem. We studied the performance of our proposed framework on a reaching task performed in simulation on 400 robots generated with various physical parameters of the existing robotic platforms. Our method obtained superior results compared to the benchmarking MAML, VERSA and AVI frameworks. The adaptation of the policy was particularly successful to the robotic platforms whose variations of the physical parameters were present in the meta-train dataset. However, the performance of all considered frameworks dropped when the adaptation was done on a novel robotic platform without a similar representative in the meta-train dataset. We leave such out-of-distribution cases as well as evaluation on more complex visuo-motor scenarios using several more platforms as future work. Given the promising results in this study, we ultimately aim to apply our framework on a multi-robot multi-task scenario.

**VI. Conclusion**

This work was supported by Knut and Alice Wallenberg Foundation, the EU through the project EnTimeMent and the Swedish Foundation for Strategic Research through the COIN project.
REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” The Journal of Machine Learning Research, 2016.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” ArXiv:1707.06347, 2017.

[3] A. Ghandiradze, A. Maki, D. Kragic, and M. Bjrko, “Deep predictive policy training using reinforcement learning,” in IROS, 2015.

[4] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, “Efficient adaptation for end-to-end vision-based robotic manipulation,” arXiv preprint arXiv:2004.10190, 2020.

[5] I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning,” in ICRA, 2019.

[6] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” ICML, 2017.

[7] C. Finn, K. Xu, and S. Levine, “Probabilistic model-agnostic meta-learning,” in NeurIPS, 2018.

[8] J. Gordon, J. Bronskill, M. Bauer, S. Nosozin, and R. Turner, “Meta-learning probabilistic inference for prediction,” in ICLR, 2019.

[9] K. Arndt, M. Hazara, A. Ghandiradze, and V. Kyrki, “Meta reinforcement learning for sim-to-real domain adaptation,” in ICRA, 2020.

[10] K. Arndt, A. Ghandiradze, M. Hazara, and V. Kyrki, “Few-shot model-based adaptation in noisy conditions,” arXiv preprint arXiv:2010.08397, 2020.

[11] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, G. Powell, A. Ray et al., “Learning dexterous in-hand manipulation,” The International Journal of Robotics Research, 2020.

[12] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, “Continuous adaptation via meta-learning in nonstationary and competitive environments,” in ICLR, 2018.

[13] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, “Learning modular neural network policies for multi-task and multi-robot transfer,” in ICRA 2017.

[14] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, “Robonet: Large-scale multi-robot learning,” in Conference on Robot Learning, 2019.

[15] W. Huang, I. Mordatch, and D. Pathak, “One policy to control them all: Shared modular policies for agent-agnostic control,” ICML, 2020.

[16] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, “Jointly learning to construct and control agents using deep reinforcement learning,” in ICRA, 2019.

[17] T. Chen, A. Murari, and A. Gupta, “Hardware conditioned policies for multi-robot transfer learning,” in NeurIPS, 2018.

[18] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in ICRA, 2017.

[19] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in NeurIPS, 2017.

[20] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in CVPR, 2018.

[21] M. A. Jamal and G. Qi, “Task agnostic meta-learning for few-shot learning,” in CVPR, 2019.

[22] H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B.-G. Hu, “Lgm-net: Learning to generate matching networks for few-shot learning,” ArXiv:1905.06331, 2019.

[23] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, “Recasting gradient-based meta-learning as hierarchical bayes,” in ICLR, 2018.

[24] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, “Bayesian model-agnostic meta-learning,” in NeurIPS, 2018.

[25] A. A. Rusu, D. Rau, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, “Meta-learning with latent embedding optimization,” in ICLR, 2019.

[26] S. Ravi and A. Beatson, “Amortized bayesian meta-learning,” in ICLR, 2019.

[27] J. Harrison, A. Sharma, and M. Pavone, “Meta-learning priors for efficient online bayesian regression,” in International Workshop on the Algorithmic Foundations of Robotics, 2018.

[28] J. Kaddour, S. Sæmundsson, and M. P. Deisenroth, “Probabilistic active meta-learning,” ArXiv:2007.08949, 2020.

[29] Y. Zou and X. Lu, “Gradient-em bayesian meta-learning,” in ICRA 2017.

[30] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, “Meta-learning with latent embedding optimization,” in ICLR, 2019.

[31] S. James, M. Bloesch, and A. J. Davison, “Learning one-shot imitation from humans without humans,” IEEE Robotics and Automation Letters, 2020.

[32] J. Büttepape, A. Ghandiradze, Ö. Öztürm Karadag, M. Björkman, and D. Kragic, “Imitating by generating: Deep generative models for imitation of interactive tasks,” Frontiers in Robotics and AI, 2020.

[33] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, “Sim2real viewpoint invariant visual servoing by recurrent control,” in CVPR, 2018.

[34] A. Hämäläinen, K. Arndt, A. Ghandiradze, and V. Kyrki, “Affordance learning for end-to-end visuomotor robot control,” in IROS, 2019.

[35] X. Chen, A. Ghandiradze, M. Björkman, and P. Jensfelt, “Adversarial feature training for generalizable robotic visuomotor control,” in ICRA, 2020.

[36] L. Pinto and A. Gupta, “Learning to push by grasping: Using multiple tasks for effective learning,” in ICRA, 2017.