Offline Meta-Reinforcement Learning for Industrial Insertion

Tony Z. Zhao[*1] [†] **Jianlan Luo**[*2] **Oleg Sushkov**[3] **Rugile Pevceviciute**[3]

Nicolas Heess[3] **Jon Scholz**[3] **Stefan Schaal**[2]

Sergey Levine[45]

[1] X, The Moonshot Factory [2] Intrinsic [3] DeepMind [4] Google Brain [5] UC Berkeley

Abstract

Reinforcement learning (RL) can in principle make it possible for robots to automatically adapt to new tasks, but in practice current RL methods require a very large number of trials to accomplish this. In this paper, we tackle rapid adaptation to new tasks through the framework of meta-learning, which utilizes past tasks to learn to adapt, with a specific focus on industrial insertion tasks. We address two specific challenges by applying meta-learning in this setting. First, conventional meta-RL algorithms require lengthy online meta-training phases. We show that this can be replaced with appropriately chosen offline data, resulting in an offline meta-RL method that only requires demonstrations and trials from each of the prior tasks, without the need to run costly meta-RL procedures online. Second, meta-RL methods can fail to generalize to new tasks that are too different from those seen at meta-training time, which poses a particular challenge in industrial applications, where high success rates are critical. We address this by combining contextual meta-learning with direct online finetuning: if the new task is similar to those seen in the prior data, then the contextual meta-learner adapts immediately, and if it is too different, it gradually adapts through finetuning. We show that our approach is able to quickly adapt to a variety of different insertion tasks, learning how to perform them with a success rate of 100% using only a fraction of the samples needed for learning the tasks from scratch. Experiment videos and details are available at https://sites.google.com/view/offline-metarl-insertion.

1 Introduction

Industrial robotic manipulation tasks are gaining more traction than ever, partially due to the sharply increasing interest in AI-enabled applications and generally more suitable and affordable robot hardware [1, 2, 3]. Part mating and part insertion are very common in the manufacturing industry, and the goal to enable high-mix/low-volume scenarios (e.g., lot-size one) are among the most prominent research topics [4]. Current manufacturing methods often rely on human-engineered heuristics that must be redesigned for each new instance. In this context, reinforcement learning provides an appealing and automated alternative. Previous works have demonstrated that reinforcement learning can solve some of these manipulations tasks [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. However, lengthy exploration processes are often required. For example, it would usually take a RL algo-

[†]Equal contribution. Work done as an intern at X, The Moonshot Factory.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021

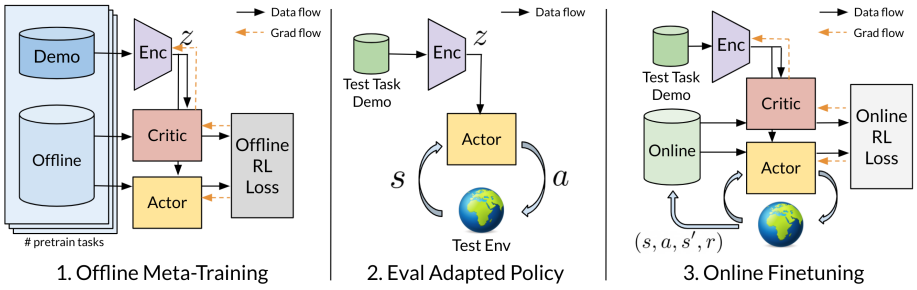

Figure 1: During offline meta-training (left), *ODA* trains the policy using offline data for each training task, with an encoder that learns to summarize the demonstrations into a latent variable z. Next, during adaptation to a new task (middle), we condition the encoder on demos for the new task, obtaining a policy that either solves the new task immediately, or else provides a initialization for the third stage (right), where the entire policy is finetuned end-to-end with online interactions.

rithm about four hours to solve one connector insertion task [6]. Moreover, this process needs to be repeated for each individual connector, even though these connectors can in principle bear similar insertion strategies; e.g., the knowledge to insert a USB-A connector should provide some guidance for inserting a USB-C connector.

Meta-RL algorithms can make it possible to adapt to new tasks quickly by pre-training on a distribution of tasks [16, 17, 18, 19, 20, 21, 22, 23, 21, 24]. However, the lengthy online meta-training phase these methods require can be prohibitive in industrial robotics settings. On the other hand, robots that are already deployed at manufacturing sites can already generate large amounts of potentially useful data from their existing controllers. This makes it appealing to consider incorporating ideas from offline RL [25, 26, 27, 28, 29], which can make it possible to utilize previously collected datasets in place of active data collection. Several works have proposed offline algorithms for meta-RL [30, 31, 32, 33]. Our offline meta-RL approach builds on Pong et al. [30], but incorporates a number of design choices that make it practical for industrial applications: (i) we leverage demonstrations for both training tasks and the test task, which reduces the need for lengthy exploration and accelerates adaptation; (ii) we combine meta-learned adaptation via a PEARL-style encoder [17] with full finetuning of the entire policy, such that even when the new task is very different from those seen in meta-training, the policy can still adapt gradually until it succeed.

Concretely, our method first meta-trains a policy on available static datasets. For a new test task, we then adapt the policy using user-provided demonstrations. This phase may already solve most tasks, but if it does not, we then follow up with a fine-tuning phase until satisfactory performance is attained. The offline training dataset can comprise demonstration data from humans, replay buffer data from other RL policies, or even data from hand-designed ("scripted") policies. During meta-training, we first pass human demonstration data into an encoder to infer the task variable z; then we condition both the actor and critic on this task variable in addition to observations or actions. The policy, critic and contextual encoder will then be trained jointly with offline reinforcement learning. This encourages the encoder to pick up task information from demonstrations that is useful for actor and critic. During adaptation to a new task, we infer a new task variable from a few human demonstrations. We then evaluate the adapted policy and, if it does not solve the task well enough, we finetune the policy end-to-end with additional online samples.

Our contributions are summarized as follows: We present a novel offline meta-RL framework that integrates a number of design choices to make it practical in realistic industrial insertion settings. Our method can adapt to unseen tasks reliably and rapidly by leveraging static offline datasets and optional on-policy robot interaction. We evaluate it with a total of 13 industrial insertion tasks, including ones like RAM insertion that has tight tolerance and requires delicate handling. Our method is able to achieve 100/100 success for all 13 tasks within 30 minutes of adaptation.

2 Background

In this section, we give a brief background of RL and offline meta-RL. We consider a Markov Decision Process (MDP) associated with task $\mathcal{T}$ defined by $\mathcal{M}_{\mathcal{T}} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0, \gamma\}$. $\mathcal{S}$ is the

state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ governs the transition dynamics, $\mathcal{R}$ is the reward function, γ is the discount factor, $p_0(s)$ is the initial state distribution. The replay buffer for the task $\mathcal{D}_\mathcal{T}$ consists of state, action, reward, next state tuples (s, a, r, s'). We also define the associated Q function of the policy as $Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_\pi[\gamma^{t'-t} r(s_{t'}, a_{t'})|s_t, a_t]$, with $a_{t'}$ taken with the current policy π. We can learn this Q-function via Bellman backups in actor-critic methods [34, 35], and the objective of reinforcement learning is to find the policy that maximize cumulative discounted rewards $R = \sum_t \gamma^t r(s_t, a_t)$.

In the offline meta-RL setting, we assume a task distribution $p_\mathcal{T}(\cdot)$, and every task $\mathcal{T}$ is an aforementioned MDP. We slightly abuse the notion of $\mathcal{D}_\mathcal{T}$ to represent the static dataset linked to each task. During offline meta-training, we have N training tasks $\mathcal{T}_{i=1,\ldots,N}$ sampled from $p_\mathcal{T}(\cdot)$ and we assume access to dataset $\mathcal{D}_{\mathcal{T}_i}$ for each sampled task. We optimized the appropriate meta-training loss using the datasets to learn a meta-policy $\pi_{\theta^\star}$ and a meta-adaptation procedure $f_{\phi^\star}$. At meta-test time, we face a new task $\mathcal{T}'$ sampled from $p_\mathcal{T}(\cdot)$, and we assume access to the associated new dataset $\mathcal{D}_{\mathcal{T}'}$. The meta-adaptation procedure adapts the meta-policy to the new task, i.e. $\hat{\pi} = f_{\phi^\star}(\pi_{\theta^\star}, \mathcal{D}_{\mathcal{T}'})$, and we aim to maximizes the return of $\hat{\pi}$:

$$\theta^\star, \phi^\star = \text{argmax}_{\theta,\phi} \sum_{t=0}^{T} [\gamma^t r(s_t, a_t)]; \tag{1}$$

where s_t, a_t denote the state and action taken at step t when executing the adapted policy $\hat{\pi}$ in $\mathcal{T}'$.

3 Related Work

In addition the works cited in Sec. 1, we discuss several previous works that focus on insertion problems. Heuristic-based methods combined with compliance-controlled robotic arms or even passive mechanisms [36, 37, 38, 39, 40, 41, 42] can solve some insertion tasks reliably, but such methods typically require manual tuning for every type of task and connector. Gerrit et al. [24] applied PEARL[17] for solving industrial insertion tasks in a sim2real fashion, while it has yet to be shown that such domain randomization approach could scale beyond the two insertion tasks considered in the paper. We posit that it would require a significantly larger family of parameters to randomize over in order to capture the complex physics like contact dynamics and deformation, making it computationally expensive and involves careful engineering. While our paper show that a broader range of connectors insertion problems can be solved by collecting the appropriate data in the real world and process them with offline meta-RL. Another relevant work is by Spector et al. [43]. This work learns a residual insertion policy on top of an existing PD controller through "backward learning." Unlike [44, 9], the residual policy is learned in a supervised fashion for sample efficiency. They generate the training data by a scripted policy producing random perturbations starting from the goal. They show this type of template policy can in principle tackle a series of industrial insertion tasks. However, as the authors pointed out, this method has trouble inserting connectors with small clearance, thus resulting in lower performance for connectors such as USB-C, whereas our method is able to consistently adapt to new connectors with a 100% success rate, as shown in our experiments. In addition, the paper focused on visual adaptation, which can be a different setting than ours.

4 Offline RL with Demonstration Adaptation

In our method, which is summarized in Fig. 1, an adaptive policy is meta-trained via offline RL on a user-provided dataset consisting of trials for a variety of different tasks, together with corresponding policies for each task. This adaptive policy is then used to solve new tasks, first by adapting to them using a small number of demonstrations, and then by finetuning with online RL. In the meta-training stage, we assume access to two buffers containing demonstrations and offline data respectively, for each of the pretrain tasks. Our goal is to meta-train the policy so that it can adapt to new tasks using user-provided demonstrations, which are easy and safe to provide for a new task in limited quantity, rather than adapting entirely via online exploration trials as in prior work. Our meta-learned policy consists of a policy network $\pi_\theta(a|s, z)$ that is conditioned on the current state s and a *latent code* z that denotes the task, as well as an encoder network $q_\phi(z|c)$ that predicts z for each task from a *context* c, which consists of the demonstrations. Intuitively, this encoder extracts the information necessary to determine how to perform the task from the demos and consolidates it into z, as we will

discuss in this section. This design resembles prior methods such as PEARL and MELD [17, 21], with the main difference that we specifically encode demonstrations rather than online experience. However, unlike meta-imitation methods [45], the policy and encoder are still meta-trained with RL.

When our method needs to adapt to a new task (e.g., insert a new type of connector), we assume that we are first provided with demonstrations, which form the context c for the new task and which we use to infer $z \sim q(z|c)$. Then, we can run $\pi(a|s, z)$. In many cases, this policy will already perform the task successfully. However, if the new task is substantially different from the tasks seen during meta-training, the encoder may be unable to infer a z for which the policy is consistently successful. In this case, we can finetune the entire policy and encoder with reinforcement learning and additional online data collection. Typically, this finetuning process is quite fast (5-10 minutes in our experiments), since $\pi(a|s, z)$ already provides a very good initialization. We describe the components of our method in more detail below.

4.1 Contextual Meta-Learning with Demonstrations

The design of our meta-reinforcement learning model resembles prior contextual meta-RL algorithms, such as PEARL and MELD [17, 21], with the principle differences being that we utilize offline data and adapt using demonstrations. In this design, adaptation is performed via an encoder network $q_\phi(z|c)$, which reads in a context c consisting of some trajectories for the current task, and infers a vector of latent variables z that is passed to the policy $\pi_\theta(a|s, z)$. In this way, z encapsulates the information needed to perform the current task. We follow prior work and use the same architecture for $q_\phi(z|c)$ as PEARL. Training is performed end-to-end via RL, which trains both $\pi_\theta(a|s, z)$ and $q_\phi(z|c)$ together to maximize performance on the meta-training tasks. We discuss the specific algorithm we use in Section 4.2. We deviate from prior work in our choice of context c: while PEARL and MELD [17, 21] use trials from the latest policy to generate trajectories for the context, our method uses user-provided demonstrations. Such demonstrations are typically easy to provide in small quantities, and spare the method from needing to explore via trial and error. In our experiments, we collect demonstrations using a space mouse to control the tool-center-point (TCP) pose. To meta-train $q_\phi(z|c)$ to utilize demonstrations, we maintain two buffers during meta-training: a demonstration buffer, which is used to form the context for each minibatch, and another buffer that consists of all offline data and is used to sample the transitions for actor-critic updates. Following Rakelly et al. [17], we add a KL-divergence loss to $q_\phi(z|c)$ to minimize unnecessary information content in z. This is summarized in Algorithm 1.

In the offline setting, demonstrations also allow us to sidestep the train-test distribution shift problem in offline meta-RL pointed out by Pong et al. [30]. When using the meta-trained policy to collect data to form the context c, the data collected for a new task will differ systematically from the offline data used for training $q_\phi(z|c)$ during the offline meta-training phase, since the learned policy differs from the policy that collected the offline data, and as pointed out by Pong et al. [30], this can lead to very poor performance for standard offline meta-RL methods. Since we instead use demonstrations, which do not experience this distributional shift, our method performs well even with purely offline meta-training.

4.2 Offline and Online Reinforcement Learning

In order to meta-train on offline data, we require a suitable offline RL algorithm. Since our aim is to meta-train a policy that can then be fine-tuned to new tasks, we use Advantage-Weighted Actor-Critic (AWAC) [28], which has been shown to perform well in offline training followed by fine-tuning. We summarize AWAC in this section. AWAC aims to learn a policy that maximizes reward while bounding the deviation from the data distribution π_β:

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\pi_\theta(\mathbf{a}|\mathbf{s})}[Q_\phi(\mathbf{s}, \mathbf{a})] \,\mathbf{s.t.}\, D_{KL}(\pi_\theta \| \pi_\beta) \leq \epsilon$$

This constrained optimization can be solved via Lagrangian duality, and the solution can be approximated via weighted maximum likelihood:

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\beta}[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \exp(A^\pi(\mathbf{s}, \mathbf{a}))],$$

where $A^\pi(\mathbf{s}, \mathbf{a}) = Q_\phi(\mathbf{s}, \mathbf{a}) - E_{\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}[Q_\phi(\mathbf{s}, \mathbf{a})]$ is the advantage under current policy π. The critic Q_ϕ is trained to minimize the standard Bellman error. For further detail, we refer to prior work [28].

Algorithm 1 ***ODA*** **Meta-training**

Require: D^i_{demo} and $D^i_{offline}$ for each of N training tasks, learning rates η_1, η_2, η_3, temperature λ, KL weight β

1: Init. encoder q_ϕ, actor π_θ, critic Q_ψ
2: **while** not converged **do**
3: **for** task $i = 1, 2, ..., N$ **do**
4: Sample demo data as context $c_i \sim D^i_{demo}$
5: Sample offline data $(s, a, s', r) \sim D^i_{offline}$
6: Sample task variable $z_i \sim q_\phi(\cdot|c_i)$
7: $y = r(s, a) + \gamma\, \mathbb{E}_{s',a' \sim \pi_\theta(\cdot|s',z_i)}\, Q_\psi(s', a', z_i)$
8: $\mathcal{L}^i_{critic} = (Q_\psi(s, a, z_i) - y)^2$
9: $\mathcal{L}^i_{actor} = -\log \pi_\theta(a|s, z_i) \exp(\frac{1}{\lambda} A^\pi(s, a, z_i))$
10: $\mathcal{L}^i_{KL} = \beta D_{KL}(q_\phi(\cdot|c_i) \| \mathcal{N}(0, I))$
11: **end for**
12: $\phi \leftarrow \phi - \eta_1 \nabla_\phi \sum_i (\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL})$
13: $\theta \leftarrow \theta - \eta_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
14: $\psi \leftarrow \psi - \eta_3 \nabla_\psi \sum_i \mathcal{L}^i_{critic}$
15: **end while**

This RL algorithm is used to meta-train the policy, critic and encoder, and also to finetune it in the last stage of adaptation. The design of our approach resembles the method of Pong et al. [30], with several critical differences: (i) We condition the encoder on demonstrations, making it possible to adapt to new tasks without risky exploration. This is important in safety-critical industrial settings. (ii) Unlike Pong et al. [30], we do not require any online meta-training phase, because the conditioning demonstrations do not incur distributional shift. This makes our method significantly easier to meta-train, since all meta-training uses offline data with no online collection at all. (iii) We utilize a hybrid adaptation procedure where we *first* use the encoder to adapt to demonstrations, and then finetune further with online AWAC to maximize performance. As we show in our experiments, this finetuning stage can significantly improve performance on difficult tasks. This combination of design decisions is novel to our approach. Though it does combine a number of parts that are based on prior work, as we demonstrate in our experiments, the particular combination of parts we employ is important for good final performance. We describe both stages in detail in Algorithm 1 and 2. To avoid clutter, we describe the algorithm when batch size is 1 during data sampling (e.g. Algo. 1 line 4, 5), when in practice we sample a mini-batch. For reproducibility, we include network architectures, hyperparameters and other implementation details on our website.

Algorithm 2 ***ODA*** **Adaptation and Finetuning**

Require: Test task demo D_{test}, learning rates η_1, η_2, η_3, temperature λ, KL weight β, Pretrained π_θ, Q_ψ, q_ϕ

1: Init. empty online buffer $\mathcal{B}$
2: Sample demo data as context $c \sim D_{test}$
3: Sample task variable $z \sim q_\phi(\cdot|c)$
4: Evaluate policy $\pi_\theta(\cdot|s, z)$, **exit** if policy solves the task
5: **while** not converged **do**
6: collect trajectory τ by executing policy $\pi_\theta(\cdot|s, z)$
7: add τ to $\mathcal{B}$
8: Sample demo data as context $c \sim D_{test}$
9: Sample offline data $(s, a, s', r) \sim \mathcal{B}$
10: Sample task variable $z \sim q_\phi(\cdot|c)$
11: Calculate $\mathcal{L}_{actor}$, $\mathcal{L}_{critic}$, $\mathcal{L}_{KL}$ same as Algo.1
12: $\phi \leftarrow \phi - \eta_1 \nabla_\phi (\mathcal{L}_{critic} + \mathcal{L}_{KL})$
13: $\theta \leftarrow \theta - \eta_2 \nabla_\theta \mathcal{L}_{actor}$
14: $\psi \leftarrow \psi - \eta_3 \nabla_\psi \mathcal{L}_{critic}$
15: **end while**

Figure 2: **Offline training tasks.** For each of the 11 training tasks, we run DDPGfD [5] to collect data for offline training.

	US-AC-plug	NEMA14-30P	Metal-peg-rec	Metal-peg-rd	UK-AC-plug	Car-plug-4p	Metal-peg-sq	Car-plug-3p	EU-AC-plug
Ours	**100/100**	**100/100**	**100/100**	**100/100**	**100/100**	**100/100**	**99/100**	**75/100**	99/100
AWAC	87/100	93/100	96/100	99/100	**100/100**	**100/100**	90/100	64/100	**100/100**

Figure 3: **Test tasks and adaptation performance.** We evaluate adaptation with 9 test tasks and compare it to AWAC [28]. Our method *ODA* achieves 100% success rate for 6/9 tasks, and outperforms AWAC for 8/9 tasks. Adaptation only uses the demos for each task, without additional online interaction.

5 Experiments

We evaluate our algorithm by adapting to 12 new tasks (see Fig. 3 and **??**), studying the following questions:

1. How well does our method perform compared to standard offline RL?
2. Can finetuning provide fast adaptation even for very different novel tasks, where just running the meta-trained encoder fails?
3. Can our method handle the challenging tasks in Fig. **??**, which are outside of the training distribution?
4. Does our meta-adaptation improve with more offline training data?

Experiment setup In our insertion tasks, the robot starts holding an object (e.g., a connector), and must position it into a goal pose where it is inserted into a socket. The reward is a binary indicator for whether the insertion is successful, and the episode terminates either when the robot succeeds, or the episode length exceeds a maximum time limit. We run experiments with a KUKA iiwa7 robot. The agent controls the TCP twist of the robot at 10Hz, which is tracked and interpolated by a downstream impedance controller at 1000 Hz. The observation to our agent consists of the robot's TCP pose, velocity, and the wrench measured at the tool tip. For both training and test tasks, we express robot TCP information in a relative coordinate system, with its origin located at the TCP, as in prior work [6]. We add perturbation noise from a uniform distribution $\mathcal{U}$[-1mm, 1mm] at the beginning of each episode, at both training and test time. The policy does not have access to this noise, and therefore must be robust to the perturbations.

Offline dataset We collected data for 11 different plug-socket pairs, as shown in Figure 2. The data was collected by running DDPGfD [5] and saving the entire replay buffer as offline data for our method, as well as demonstrations for each task. On average, we obtain 500 episodes of RL interaction data and 20 episodes of human demonstration data for each of the training tasks. To further improve the diversity of the dataset, we also reran the final DDPGfD policies with added noise to provide good coverage of the workspace. We list dataset details in our website.

5.1 Adaptation via Learned Encoder

In our first set of experiments, we evaluate how well our learned encoder can adapt to new tasks using only the demonstrations for those tasks. We use 9 test tasks as shown in Fig. 3. We collect 20 demonstrations for each task and feed them into our encoder to infer the task information z. We compare to standard AWAC pretrained on the same data, with demos added into the offline data. As

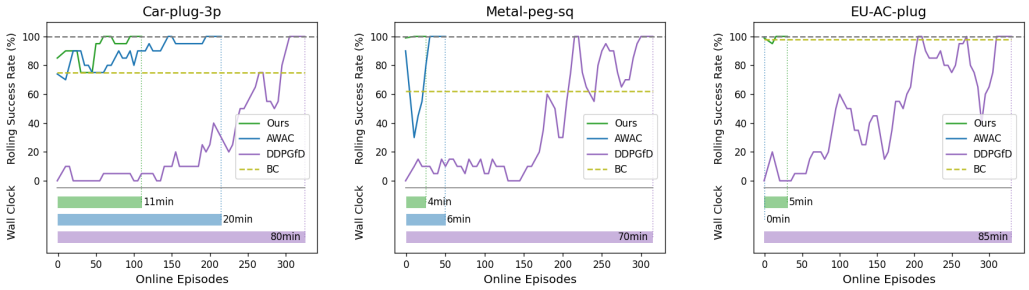

Figure 4: **Finetuning performance.** For tasks where adapting with demos does not reach 100/100, we finetune with online interaction. We plot the training success rate and wall-clock time for *ODA*, finetuning AWAC[28], DDPGfD[5], as well as behavioral cloning. Our method reduces wall-clock time by 11.75x compared to DDPGfD, and outperforms AWAC. We note that the length of bar plot reflects the number of episodes, and successful episodes are significantly shorter wall-clock.

shown in Fig. 3, our method performs well across all of the tasks, succeeding 100/100 times on 6 of the 9, while AWAC attains comparatively lower success rates.

AWAC learns a single policy for all connectors, and relies on this policy to generalize effectively, whereas our approach can adapt to each task using a small amount of demonstration data. Let's look at a motivating example to further illustrate this difference. We tested pretrained policies both from our method and AWAC on one of the training tasks, "E-Model-10p" (see Fig. 2). Since this is a training task, we expect both methods to do well, but AWAC only has a success rate of 49%, while our method gets 100%. This task requires fine-grained adjustments to insert, which are quite distinct from other connectors in the dataset, making it hard for the non-adaptive AWAC policy to succeed on this task and all the others with the same strategy. On the other hand, our method succeeds by determining the right strategy from the provided demonstrations. Of course, we could finetune AWAC with additional data, which would make for a more fair comparison. We study precisely this in the next set of experiments.

5.2 Handling Out-of-Distribution Tasks via Finetuning

If the task at test-time is out-of-distribution relative to tasks seen during offline training, simply utilizing the demos with the encoder might not be enough to succeed. In this section, we study how finetuning can alleviate this issue, using the three tasks where our method fails to achieve 100% performance from only the demos. In Fig. 4, we show finetuning results for (1) our method, (2) finetuning AWAC with demonstrations added to initial buffer, (3) training DDPG from demos, and (4) the performance of behavioral cloning with the demos. (4) fails to reach 100% because imitating human demonstrations will not produce a policy that is robust to starting pose randomization. (1), (2), and (3) all attain a success rate of 100/100 after finetuning, while our method finetunes significantly faster as shown in the bar plot below each graph in Fig. 4. On average, we finish training 11.75 times faster than DDPGfD. This is not completely surprising because the power of offline training: it allows us to leverage the rich information in offline datasets. We also find that in the two tasks where our method outperforms AWAC (left and middle plot of Fig. 4), we finish training 1.73 times faster than AWAC. For the third task (right plot of Fig. 4), our algorithm takes 5 mins to account for the performance difference. Note the speedup comparison here is conservative: we do not include the tasks where AWAC fails but our method achieves 100% without any finetuning at all (i.e., the first 4 tasks in Fig. 3). It's also worth noting that we have tried to pretrain a policy with DDPG on the offline datasets, but it showed 0% success rate on test tasks. We then finetune the policy pretrained with DDPG loss, while it takes similar amount of time to succeed as randomly initialized networks. This validates the necessity of offline RL training. We did not include it in the figure for presentation clarity.

5.3 Out-Of-Distribution Challenge Tasks

In realistic industrial assembly tasks, the robot may be tasked with problems that differ significantly from the conditions seen in training. To study how our method performs in these settings, we evaluate three additional challenge tasks that differ significantly from the training connectors and present a particular physical challenge (see Fig. 5 left): (1) RAM insertion, (2) network card insertion, and

	 RAM	 Network Card	 Complex Shape
Meta Adaptation	82/100	**100/100**	77/100
Finetune	**100/100**	-	**100/100**
Time	25min	-	27min

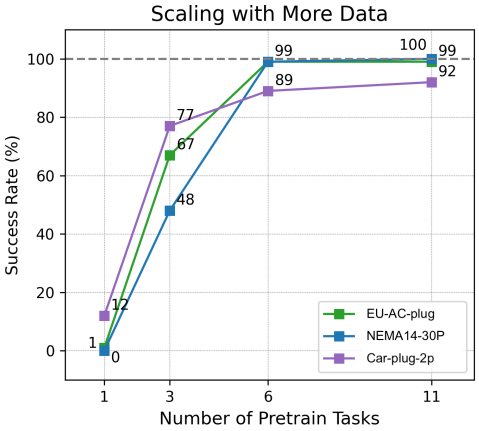

Figure 5: **Left: Challenge tasks** We challenge our method with three additional out of training distribution tasks. *ODA* is able to solve all three with 100/100 success within 30 minutes. **Right: Scaling with more data.** We plot the adaptation performance of three tasks as the number of training tasks increases. The success rate increases steadily as the algorithm has access to more data.

(3) complex shape insertion. (1) and (2) are require handling delicate electronics and significant application of force. Training from scratch would be impractical here, since an untrained policy would easily damage the circuit board. (3) is challenging because of the small clearance and the very different contact shape compared to training tasks. We do **not** inject uniform noise to the starting pose to make these three tasks more tractable.

As shown in Figure 5 left, our method is able to solve all three tasks 100/100 within 30 minutes. For network card insertion, we are able to succeed without any finetuning. We posit that inserting the RAM is significantly harder than the network card because it is much longer (288 vs. 18 pins), requires high precision along the yaw axis, and requires considerable application of force. These characteristics makes it very different from all the training tasks, and generally difficult for a robot to perform. During the 25 minutes required to adapt the policy to perform this task 100% of the time, no damage is caused to the RAM or the motherboard.

Lastly, we analyze how the number of training tasks affects adaptation performance of our method, shown in Fig. 5 right. We evaluate the meta-adaptation performance across three tasks, and it continues to improve as we increase the number of tasks in the dataset from 1 to 11.

6 Discussion

We introduced an offline meta-RL algorithm, ODA, that can meta-learn an adaptive policy from offline data, quickly adapt based on a small number of user-provided demonstrations for a new task, and then further adapt through online finetuning. We show that this approach is highly effective at adapting to new connector insertion tasks, learning to insert many connectors with a 100% success rate after just the initial demonstrations, and finetuning other connectors to a 100% success rate with minutes of additional online training. Although our algorithm is composed of parts proposed in prior work, the particular combination and the use of demonstrations for adapting in contextual meta-RL is novel to our method. Promising future directions include incorporating visual perception, extending the approach to other tasks, and extending it into a lifelong learning framework where each new task is included into a lifelong meta-training process.

References

[1] *Global Robotics Market - Growth, Trends, COVID-19 Impact, and Forecasts (2021 - 2026)*. 2021. URL: https://www.yahoo.com/now/global-robotics-market-growth-trends-125900343.html.

[2] *Introducing Intrinsic*. 2021. URL: https://blog.x.company/introducing-intrinsic-1cf35b87651.

[3] *Alphabet to launch robotics firm Intrinsic under its other bets unit*. 2021. URL: https://www.reuters.com/technology/alphabet-launch-robotics-firm-intrinsic-under-its-other-bets-unit-2021-07-23/.

[4] B. Wang. "The Future of Manufacturing: A New Perspective". In: *Engineering* 4.5 (2018), pp. 722–728. URL: https://www.sciencedirect.com/science/article/pii/S2095809918300614.

[5] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *arXiv preprint arXiv:1707.08817* (2017).

[6] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz. "Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study". In: *Proceedings of Robotics: Science and Systems*. Virtual, July 2021.

[7] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz. "A practical approach to insertion with variable socket position using deep reinforcement learning". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 754–760.

[8] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. "Reinforcement learning on variable impedance controller for high-precision robotic assembly". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3080–3087.

[9] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine. "Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5548–5555.

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[11] S. Levine, N. Wagener, and P. Abbeel. "Learning Contact-Rich Manipulation Skills with Guided Policy Search". In: *CoRR* abs/1501.05611 (2015). arXiv: 1501.05611. URL: http://arxiv.org/abs/1501.05611.

[12] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *CoRR* abs/1709.10087 (2017). arXiv: 1709.10087. URL: http://arxiv.org/abs/1709.10087.

[13] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. "Composable Deep Reinforcement Learning for Robotic Manipulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6244–6251.

[14] T. Inoue, G. D. Magistris, A. Munawar, T. Yokoya, and R. Tachibana. "Deep Reinforcement Learning for High Precision Assembly Tasks". In: *CoRR* abs/1708.04033 (2017). arXiv: 1708.04033. URL: http://arxiv.org/abs/1708.04033.

[15] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel. "Learning from the Hindsight Plan – Episodic MPC Improvement". In: *ArXiv e-prints* (Sept. 2016). arXiv: 1609.09001 [cs.RO].

[16] C. Finn, P. Abbeel, and S. Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: https://proceedings.mlr.press/v70/finn17a.html.

[17] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 5331–5340. URL: https://proceedings.mlr.press/v97/rakelly19a.html.

[18] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. "Meta-Reinforcement Learning of Structured Exploration Strategies". In: *NeurIPS*. 2018.

[19] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2016. arXiv: 1611.02779 [cs.AI].

[20] 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: *Proceedings of the Conference on Robot Learning*. Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 30 Oct–01 Nov 2020, pp. 1094–1100. URL: https://proceedings.mlr.press/v100/yu20a.html.

[21] T. Z. Zhao, A. Nagabandi, K. Rakelly, C. Finn, and S. Levine. "MELD: Meta-Reinforcement Learning from Images via Latent State Models". In: *CoRR* abs/2010.13957 (2020). arXiv: 2010.13957. URL: https://arxiv.org/abs/2010.13957.

[22] I. Clavera, A. Nagabandi, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. "Learning to Adapt: Meta-Learning for Model-Based Control". In: *CoRR* abs/1803.11347 (2018). arXiv: 1803.11347. URL: http://arxiv.org/abs/1803.11347.

[23] A. Nagabandi, C. Finn, and S. Levine. "Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL". In: *CoRR* abs/1812.07671 (2018). arXiv: 1812.07671. URL: http://arxiv.org/abs/1812.07671.

[24] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. "Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 9728–9735.

[25] S. Levine, A. Kumar, G. Tucker, and J. Fu. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. 2020. arXiv: 2005.01643 [cs.LG].

[26] A. Kumar, A. Zhou, G. Tucker, and S. Levine. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: 2006.04779 [cs.LG].

[27] A. Kumar, A. Gupta, and S. Levine. "Discor: Corrective feedback in reinforcement learning via distribution correction". In: *arXiv preprint arXiv:2003.07305* (2020).

[28] A. Nair, M. Dalal, A. Gupta, and S. Levine. *Accelerating Online Reinforcement Learning with Offline Datasets*. 2020. arXiv: 2006.09359 [cs.LG].

[29] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. *COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning*. 2020. arXiv: 2010.14500 [cs.LG].

[30] V. H. Pong, A. Nair, L. Smith, C. Huang, and S. Levine. *Offline Meta-Reinforcement Learning with Online Self-Supervision*. 2021. arXiv: 2107.03974 [cs.LG].

[31] E. Mitchell, R. Rafailov, X. B. Peng, S. Levine, and C. Finn. "Offline Meta-Reinforcement Learning with Advantage Weighting". In: *ICML*. 2021.

[32] R. Dorfman and A. Tamar. "Offline Meta Reinforcement Learning". In: *CoRR* abs/2008.02598 (2020). arXiv: 2008.02598. URL: https://arxiv.org/abs/2008.02598.

[33] L. Li, R. Yang, and D. Luo. "Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization". In: *ArXiv* abs/2010.01112 (2021).

[34] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning". In: *CoRR* abs/1509.02971 (2016).

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

[36] S. Chhatpar and M. Branicky. "Search strategies for peg-in-hole assemblies with position uncertainty". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)* 3 (2001), 1465–1470 vol.3.

[37] D. Whitney. "Force Feedback Control of Manipulator Fine Motions". In: *Journal of Dynamic Systems Measurement and Control-transactions of The Asme* 99 (1977), pp. 91–97.

[38] D. E. Whitney. "Quasi-Static Assembly of Compliantly Supported Rigid Parts". In: *Journal of Dynamic Systems, Measurement, and Control* 104.1 (Mar. 1982), pp. 65–77. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/104/1/65/5777806/65_1.pdf. URL: https://doi.org/10.1115/1.3149634.

[39] H. Park, J. Bae, J.-H. Park, M. Baeg, and J. Park. "Intuitive peg-in-hole assembly strategy with a compliant manipulator". In: *IEEE ISR 2013* (2013), pp. 1–5.

[40] K. Sharma, V. Shirwalkar, and P. K. Pal. "Intelligent and environment-independent Peg-In-Hole search strategies". In: *2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE)*. 2013, pp. 1–6.

[41] W. Newman, Y. Zhao, and Y.-H. Pao. "Interpretation of force and moment signals for compliant peg-in-hole assembly". In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. Vol. 1. 2001, 571–576 vol.1.

[42] T. Tang, H.-C. Lin, Y. Zhao, W. Chen, and M. Tomizuka. "Autonomous alignment of peg and hole by force/torque measurement for robotic assembly". In: *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. 2016, pp. 162–167.

[43] O. Spector and D. D. Castro. "InsertionNet - A Scalable Solution for Insertion". In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 5509–5516.

[44] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. "Residual Reinforcement Learning for Robot Control". In: *CoRR* abs/1812.03201 (2018). arXiv: 1812.03201. URL: http://arxiv.org/abs/1812.03201.

[45] 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.