
Few-Shot Imitation Learning for Robotic Manipulation

Sebastian Kochman
sebastko@stanford.edu

Amal Mehta
amalm@stanford.edu

Soham Sinha
sohams@stanford.edu

In this work, we study meta imitation learning for robotic manipulation to achieve few-shot learning from expert demonstrations. In particular, we focus on inverse soft Q-learning (IQL) [6], a recent non-adversarial IL method showing promising results on a number of benchmarks. We explore how it scales to complex robotic tasks, as well as combine it with meta reinforcement learning algorithms to generalize to new tasks provided with as few as one expert demonstration. We evaluate the methods on the Meta-World ML1 benchmark [18], as well as a toy environment PointEnv.

We consider two imitation learning settings: offline IL, where the agent has access only to the expert demonstrations and is not allowed to learn by interacting with the environment, and online IL, where it can learn from both. When studying IQL’s performance on individual tasks (with fixed goals) sampled from the Meta-World benchmark, we observe that it struggles to learn robustly, especially on more challenging tasks, in the offline IL setting. However, it shows better results in the online IL setting. Hence, we focus on the latter in the subsequent experiments.

We frame the few-shot online imitation learning problem by following [19]: the agent first learns from K expert demonstrations (phase I), then interacts with the environment by running L trial episodes and learns from both demonstrations and trial experience (phase II) before finally being tested on a series of evaluation episodes. Similarly to [19], we focus on the case of $K = 1, L = 1$.

The main meta imitation learning algorithm that we propose is implicit MAML [12] (iMAML) with inverse soft-Q learning [6] (IQL) in the inner loop, or **iQLMAML**. Contrary to “standard” MAML [5], iMAML can handle optimization problems based on dynamic programming, like Q-learning, by estimating gradient of the whole optimization trajectory using a Conjugate Gradient (CG) solver. This allows us to find parameters θ of the actor and the critic (IQL leverages SAC [7] for continuous action spaces) that are good starting parameters for per-task adaptation using a single demonstration and a single trial epoch.

We also consider an algorithm based on RL² [4] which we call **IQL²**. It augments standard IQL by adding a recurrent layer to the actor network, which passes the hidden state across steps and episodes within the same task. When a new task starts, the hidden state is reset and a sequence representing the expert demonstration is ingested in phase I (output of the network is ignored). Then, the hidden state is passed to the subsequent steps of the trial episode in phase II and the evaluation episode. By backpropagating gradient through all the steps of the recurrent network, it learns how to encode the task’s goal in the hidden state, based on the initial demonstration.

Finally, a baseline approach we investigate is MAML with Behavioral Cloning (BC) and Trust Region Policy Optimization (TRPO) [14] in the inner loop.

We conclude our work by presenting partial experimental results we were able to obtain. iQLMAML, while well-principled, has proven to be challenging to implement. Our current implementation suffers from a memory leak which prevented us from evaluating it in longer experiments. Another challenge has been the algorithm’s latency - about $20\times$ higher than MAML with BC and TRPO. This may be caused by the dependency on the CG solver - if verified, this valuable insight shows that there may be still an important gap to fill in optimization-based meta-learning methods to handle algorithms like Q-learning in the inner loop.

1 Introduction

Robotics has been a promising field for development of Reinforcement Learning, especially as robots have become more ubiquitous in our daily lives. In particular, as robotics grow more complex in its internal parameters, the control and examination also grows in complexity as the inverse kinematics become more intractable to solve due to redundancy.

In recent years, there has been a push for rapid robot learning to make them adaptable to changing conditions and tasks, as not all environments in which robots work in are well known in advance and can become cost-prohibitive from building custom environments. In particular, a focus on learning from human demonstrations has been a key feature and promising area of development, in many ways unburdening the human operator from programming the robot to do specific tasks. However, there are key challenges in real world reinforcement learning - from not being able to reset the environment, not being able to access the low-level observations, and no strictly defined reward function.

Instead, we focus on using the simpler idea of evaluating the algorithms in a simulation modeling the robot. In particular, we focus on the Mujoco simulation of the Sawyer robot by Rethinks Robotics, a high-performance collaborative robot equipped with a gripper. Sawyer has industrial uses from dangerous situations such as metal processing, injection molding, and circuit board assembly.

One drawback of such robots is that they are deployed to do a single type of task, and as a result requires an equal number of robots to do an equal number of tasks, becoming cost-prohibitive at scale. Therefore, focus on algorithms that enable meta-learning is a key area of research, as it will enable fast-learning, adaptation at scale, and cost-reduction of a system of robots.

In this project, we combine these two ideas and evaluate whether we can teach the Sawyer Robot tasks from demonstrations, and furthermore, train the Sawyer Robot to meta-learn a series of tasks in a simulated environment. We focus in particular on a new algorithm for imitation learning - Inverse Q Learning or iQ-Learn that was recently published in 2021 [6], combine it with a resource efficient meta-learning algorithm Implicit Model Agnostic Meta Learning - iMAML, and compare its effectiveness to standard baselines.

2 Related Work

2.1 Demonstrations

The main type of learning that we are focused on are imitation learning, and in particular learning from demonstrations. There are various types of demonstrations (Human obtained (Expert, proficient, intermediate), simulation generated, etc), and all have their merits on usage. For Human level demonstrations, we refer to the RoboSuite and RoboMimic environments which are built around such demonstrations for industrial robots, including Sawyer [10]. For simplicity, we use expert demonstrations generated from the simulation environment, with no assumption on human usage.

2.2 Imitation Learning

A few of the most common forms of imitation learning are behavior cloning, inverse reinforcement learning, and generative adversarial imitation learning.

The simplest and most popular IL method is Behavioral Cloning (BC), which leverages supervised learning to minimize error between actions returned by the learned policy and expert actions, for every state represented in the expert demonstrations [8, 13]. This method can be efficient when applied to certain simpler tasks. However, it doesn't scale to long-horizon tasks, where mismatch between train and test state distributions can cause compounding errors which the agent often cannot recover from.

Inverse reinforcement learning [1, 11, 17] is a technique that recovers the reward function of the environment based on expert demonstrations. Such learned reward function can be then used to train the agent using any off-policy reinforcement learning algorithm.

Generative Adversarial Imitation Learning [9] (GAIL) is a model-free algorithm which combines ideas from GANs (generators and discriminators) and inverse-RL. GAIL tries to train generators that have similar behaviors to the experts and the discriminator tries to determine whether the behaviors

conducted by the agent in fact resemble expert behaviors. This strategy learns the policy directly and enables for better learning of sequential decisions. However, it is not sample efficient when it comes to interacting with the environment.

IQ-learn, a recent algorithm, doesn't use an adversarial method (what was introduced by GAIL), but instead learns a Q-function which is able to encapsulate both the reward and the policy. This method achieves state of the art results on many offline and online imitation learning benchmarks in Gym Environments such as Atari, Cartpole, while increasing sample efficiency.

2.3 Meta-Learning

Interestingly enough, there are some papers that utilize meta-learning to improve imitation learning methods. [19] proposes an algorithm to learn from both demonstrations and trial-and-error demonstrations. The paper applies its methods on vision-based single-arm simulated robotics control tasks.

Duan et. Al [3] proposed a meta learning setting for one-shot imitation learning of robotics tasks. The proposed solution includes an agent's network architecture accepting the demonstration sequence directly as an input, in addition to the current state the agent needs to act on. By using soft attention, the network can attend to appropriate parts of the demonstration to take the next step. The tasks used in the evaluation are limited to block stacking, where each task corresponds to a different permutation of blocks as a goal.

Cachet et. Al [15] extended the architecture used in Duan et. AL [3] by using transformers, and was able to show that transformers employ strong meta-learning transferability. Vaswani et. Al [16], as well as evaluated the transformer method on the Meta-World benchmark [18]. A similar approach of using meta-learning has been formalized as "Demonstration-Conditioned RL" in [2], and is focused on learning from relatively sub-optimal demonstration data-sets, as well as tackle the prevalent domain shift between a demonstrator protocol and the agent itself.

We focus on a simpler method, disregarding the domain shift between demonstrator and agent, of evaluating the validity and the limits of IQ-Learn as an imitation learning algorithm for a complex robot such as Sawyer. Furthermore, we aim to use relatively new meta-learning algorithms such as Model Agnostic Meta Learning [5], and its implicit variant [12] to see if demonstration-based imitation learning can be improved through meta-learning methods, with a focus on the Meta-World Environment and framework.

While Meta-World [18] is a good benchmark for meta RL in robotics, to our knowledge, it does not contain human expert demonstrations out of the box. However, it does contain some scripted policies which can be leveraged to generate expert demonstrations, enabling imitation learning across many tasks.

3 Problem statement

In the meta imitation learning setting, we are given a task distribution $p(\mathcal{T})$, where every task $\mathcal{T}_i$ is a Markov decision process (MDP), $\{\mathcal{S}, \mathcal{A}, r_i, \mathcal{P}, H\}$. In this work, we consider tasks with shared state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\mathcal{P}$ and horizon H , but with reward function $r_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that's unique per task. Set of train tasks $\mathcal{T}_i \sim p(\mathcal{T})$ and test tasks $\mathcal{T}_j \sim p(\mathcal{T})$ is given. Following [19], the goal is to *meta-train* an agent on tasks $\{\mathcal{T}_i\}$, such that it can quickly learn a new test task $\mathcal{T}_j$ in two phases:

Phase I The agent learns from $k = 1, \dots, K$ task demonstrations $\mathcal{D}_j^* := \{\mathbf{d}_{j,k}\}$. It can then attempt the task in $l = 1, \dots, L$ *trial episodes* $\{\tau_{j,l}\}$, for which it receives reward feedback.

Phase II The agent learns from those trial episodes *and* the original demos to succeed at $\mathcal{T}_j$.

That is followed by evaluating the trained agent on a series of evaluation episodes.

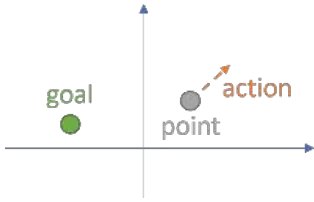
A demonstration is defined as a sequence of states and action pairs $\mathbf{d} = \{(s_t, a_t)\}$, while the trial episode includes reward signal obtained from the environment $\tau = \{(s_t, a_t, r_i(s_t, a_t))\}$.

We call such setting *offline meta IL* if $L = 0$ and *online meta IL* if $L \geq 1$. In the majority of this work, we focus on the case with $K = 1$ and $L = 1$, which can be summarized as *single-shot online imitation learning*.

4 Environments

We evaluated our methods on two environments: PointEnv and Meta-World (see summary in table 1). We chose PointEnv initially to test our algorithms on a simple environment before moving to a more complex robotics environment. We chose Meta-World due to its variety of tasks (50 distinct robotic manipulation tasks), and because it provides scripted policies that can be used to obtain expert demonstrations. Meta-World provides three different Meta Learning setups: ML1, ML10, and ML45. We chose ML1, which provides a selection of 50 “base” tasks creating a separate benchmark. Within each such benchmark individual tasks are sampled by modifying variable parameters of the environment, like, e.g., the starting or the goal positions of an object. During meta-test time, the robot is evaluated on tasks not seen previously during meta-train phase.

Table 1: Environments used in our experiments

		
Environment	PointEnv	Meta-World: ML1
Observation	$point, point - goal _2$	39 dimensions
Action	x, y where $-0.1 \leq x, y \leq 0.1$	4 dimensions
Reward	$- point - goal _2$	shaped (see [18] for details)
Tasks	sampled goals	sampled variable parameters (starting point, goal, etc.)
Demos	optimal policy	scripted policies

5 Methods

IQ-learn Before we tried more advanced few-shot learning setting, we evaluated IQL in the complex robotics environment. Before defining the objective of IQ-learn, it is helpful to define the inverse Bellman Operator. The inverse Bellman Operator is $\mathcal{T}^\pi : \mathbb{R}^{S \times A} \rightarrow \mathbb{R}^{S \times A}$ such that

$$(\mathcal{T}^\pi Q)(s, a) = Q(s, a) - \gamma \mathbb{E}_{s' \sim P(s, a)} V^\pi(s')$$

where $V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q(s, a) - \log \pi(a|s)]$

IQ-Learn is an adversarial-free algorithm that works by maximising the following energy-based objective [6], which is just rewritten inverse Bellman operator:

$$\mathcal{J}(\pi, Q) = \mathbb{E}_{\rho_E} [\phi(Q - \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} V^\pi(s'))] - (1 - \gamma) \mathbb{E}_{\rho_0} [V^\pi(s_0)]$$

The update step is two fold focusing on π and Q , the first step is to hold π constant, and maximize $\mathcal{J}(\pi, Q)$, to optimize Q . The second step is to optimize the policy π to be closer to the expert policy by holding Q constant, and applying a soft-actor critic update on the policy π to bring it closer to the expert policy held by the Q function π_Q . The pseudo code is shown in the following Figure 1.

In particular, we use IQ-learn to evaluate Sawyer on tasks from Meta-World Environment - drawer close, pick-place-v2, soccer, drawer open.

There are two modes of IQ-learn - offline and online learning. The difference between the two types of learning relate to the estimate of $\mathbb{E}_{\rho_0} [V^\pi(s_0)]$. In the online setting, we sample (s, a, s') from a

Algorithm 1 Inverse soft Q-Learning (both variants)

```
1: Initialize Q-function  $Q_\theta$ , and optionally a policy  $\pi_\phi$ 
2: for step  $t$  in  $\{1 \dots N\}$  do
3:   Train Q-function using objective from Equation 9:
      $\theta_{t+1} \leftarrow \theta_t - \alpha_Q \nabla_\theta [-\mathcal{J}(\theta)]$ 
     (Use  $V^*$  for Q-learning and  $V^{\pi_\phi}$  for actor-critic)
4:   (only with actor-critic) Improve policy  $\pi_\phi$  with SAC style
     actor update:
      $\phi_{t+1} \leftarrow \phi_t - \alpha_\pi \nabla_\phi \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\phi(\cdot|s)} [Q(s, a) - \log \pi_\phi(a|s)]$ 
5: end for
```

Figure 1: Pseudo-Code for IQ-Learn taken directly with no modifications from [6], the objective in red is the objective detailed in the subsection IQ-learn.

replay buffer, and get the estimate by $\mathbb{E}_{(s,a,s') \text{ replay}} [V^{\pi(s)} - \gamma V^{\pi(s')}]$. In the offline setting, we sample (s, a, s') from an expert buffer, and get the estimate by $\mathbb{E}_{(s,a,s') \text{ expert}} [V^{\pi(s)} - \gamma V^{\pi(s')}]$.

There are several loss functions that can be used for IQ-learn, such as v0, value, value_policy, value_expert, and value mix. They each use different variations of expert and policy states. For our experiments, we primarily used v0 and value_expert.

iQL-MAML Implicit MAML [12] (iMAML) with Inverse soft-Q learning [6] (IQL) as the loss. The concept of just regular MAML is to have a shared weight between the meta-learner and the task-learner. We chose iMAML because it was stated in the paper [12] that iMAML has the opportunity to be more resource efficient and is able to be used in dynamic processes such as Q-learning situations, and we wanted to evaluate its effectiveness.

The algorithm has two parts - the outer meta-learning state, and the inner task-learning state. During the outer loop, the meta-learner shares its weights with the inner task-learner. The task-learner then learns a policy based on the number of steps given to that specific task it was learning, and can have its own algorithm to learn that policy. The task-learner learns across all the tasks given to it. Finally, the outer loop averages the gradient of all the tasks from the task-learner, and updates itself [?]. In an analogical sense, in which how towns connect to highways in form of exits - if we have tasks as towns, the highway is the iMAML meta-learner for a specific task, the meta-learner can be quickly adapted to the task at hand.

In a more technical sense, Implicit-MAML inner loop calculates a particular Hessian vector product of the loss which is the gradient of the task at hand. The gradient vector points in the direction of the minimum descent for the weights for the task. In a direction sense to weight optimization, iMAML shows the general vector direction, but not the specific path to the optimized weights.

The General Update Rule for iMAML is written as:

$$\theta \leftarrow \theta - \eta \frac{1}{M} \sum_{i=1}^M \frac{d \text{Alg}_i^*(\theta)}{d\theta} \nabla_\phi \mathcal{L}_i(\text{Alg}_i^*(\theta))$$

However, in reality it is rewritten as:

$$\frac{d \text{Alg}_i^*(\theta)}{d\theta} = \left(I + \frac{1}{\lambda} \nabla_\phi^2 \hat{\mathcal{L}}_i(\phi_i) \right)^{-1}$$

which further is used to calculate the meta-gradient in the inner loop (w) through solving a linear minimization problem:

$$\min_w w^\top \left(I + \frac{1}{\lambda} \nabla_\phi^2 \hat{\mathcal{L}}_i(\phi_i) \right) w - w^\top \nabla_\phi \mathcal{L}_i(\phi_i)$$

Particularly, it uses Conjugate Gradient Descent as the optimizer, and the Pearlmutter method of calculating Hessian vector products to avoid calculating the full Hessian and storing it memory, making it an efficient implementation to solve for the meta-gradient.

Algorithm 1 Implicit Model-Agnostic Meta-Learning (iMAML)

```
1: Require: Distribution over tasks  $P(\mathcal{T})$ , outer step size  $\eta$ , regularization strength  $\lambda$ ,  
2: while not converged do  
3:   Sample mini-batch of tasks  $\{\mathcal{T}_i\}_{i=1}^B \sim P(\mathcal{T})$   
4:   for Each task  $\mathcal{T}_i$  do  
5:     Compute task meta-gradient  $\mathbf{g}_i = \text{Implicit-Meta-Gradient}(\mathcal{T}_i, \boldsymbol{\theta}, \lambda)$   
6:   end for  
7:   Average above gradients to get  $\hat{\nabla}F(\boldsymbol{\theta}) = (1/B) \sum_{i=1}^B \mathbf{g}_i$   
8:   Update meta-parameters with gradient descent:  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \hat{\nabla}F(\boldsymbol{\theta})$  // (or Adam)  
9: end while
```

Algorithm 2 Implicit Meta-Gradient Computation

```
1: Input: Task  $\mathcal{T}_i$ , meta-parameters  $\boldsymbol{\theta}$ , regularization strength  $\lambda$   
2: Hyperparameters: Optimization accuracy thresholds  $\delta$  and  $\delta'$   
3: Obtain task parameters  $\phi_i$  using iterative optimization solver such that:  $\|\phi_i - \text{Alg}_i^*(\boldsymbol{\theta})\| \leq \delta$   
4: Compute partial outer-level gradient  $\mathbf{v}_i = \nabla_{\phi} \mathcal{L}_{\mathcal{T}}(\phi_i)$   
5: Use an iterative solver (e.g. CG) along with reverse mode differentiation (to compute Hessian  
   vector products) to compute  $\mathbf{g}_i$  such that:  $\|\mathbf{g}_i - (\mathbf{I} + \frac{1}{\lambda} \nabla^2 \hat{\mathcal{L}}_i(\phi_i))^{-1} \mathbf{v}_i\| \leq \delta'$   
6: Return:  $\mathbf{g}_i$ 
```

Figure 2: Pseudocode for Implicit MAML. In practice, parameters δ, δ' are replaced with number of computational steps to undertake. We used 10 inner steps δ , and 5 Conjugate Gradient Steps (δ')

To integrate with iQ-Learn, the meta-learner was initialized with iQ-learn models, and the appropriate iQ-Learn loss functions were used to calculate the meta-gradient.

The Pseudo-Code is given in Figure 2.

Furthermore for task sampling, we chose a single task and then sampled different goals for the task for our meta-learning experiments.

IQL² IQL [6] with the actor architecture leveraging a recurrent neural network (RNN) to pass the hidden state across episodes within the same task, similar to RL² [4]. This includes passing the hidden state from demonstrations to trial episodes in Phase I, and then to further episodes in Phase II. As described in the RL² paper, we use Gated Recurrent Units (GRUs) instead of a traditional RNN to mitigate the problem of vanishing and exploding gradients.

Initially, just the observation is passed as input to the first GRU cell, but for each subsequent timestep, the input to the GRU cell consists of the observation concatenated with the action, reward, and the done state (represented as an integer).

Behavior Cloning + Trust Region Policy Optimization Our baseline approach: MAML with Behavioral Cloning in Phase I and Trust Region Policy Optimization (TRPO) in Phase II of the MAML inner loop, as well as TRPO loss in the MAML outer loop.

Behavioral Cloning relies on performing supervised learning between the expert action-state pairs, and the agent action-state pairs, and minimizes the loss between the two. It is particularly useful for closed small tasks that have good expert demonstrations. Since the main problem with BC is the assumption of the i.i.d state-action pairs, which in reality is a Markov Decision Process, errors compound leading to quick failures. However, a limitation is that it will clone exactly the behavior, and will not necessarily adapt to other types of successful task execution.

We use Behavioral Cloning as our baseline, as it has shown to work reasonably well in the RoboSuite and RoboMimic environments [10] and with Sawyer.

Furthermore, we employ regular MAML with BC as the baseline for the meta-testing portion as MAML and BC have been shown to be quite robust algorithms individually.

We follow the same task-sampling method where we choose a single task and sample different goals in the task environment.

6 Analysis of Experiment Results

All experiments used Adam for the optimizer for the model parameters. For computation of the meta-gradient in iQLMAML, Conjugate Gradient Descent with 5 steps was used.

6.1 IQ-Learn Performance on Single Tasks with Sawyer

We first evaluated the IQ-learn’s ability to successfully perform imitation learning on a single task, with a fixed goal, from demonstrations pulled from scripted policies in Meta-world tasks with Sawyer. We use the default IQ-learn architecture - 2 hidden layers with 128 neurons each, with ELU nonlinearities in between. We tried to keep corresponding hyper-parameters similar to the Meta-RL baselines reported in [18]. The success rate per task is shown in figure 3.

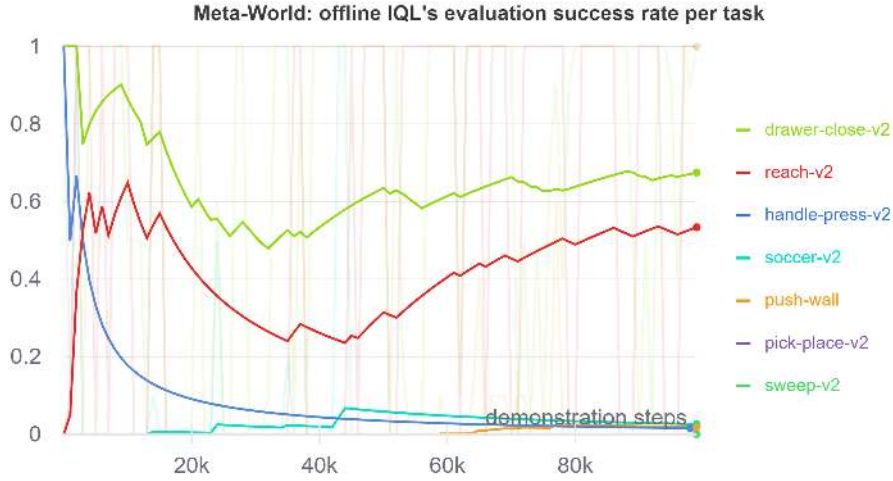


Figure 3: Offline IQL Performance (smoothing value=0.99) on Sawyer Robot Tasks on Meta World. Besides simple tasks, IQL does not help Sawyer learn complex robotics in the offline setting.

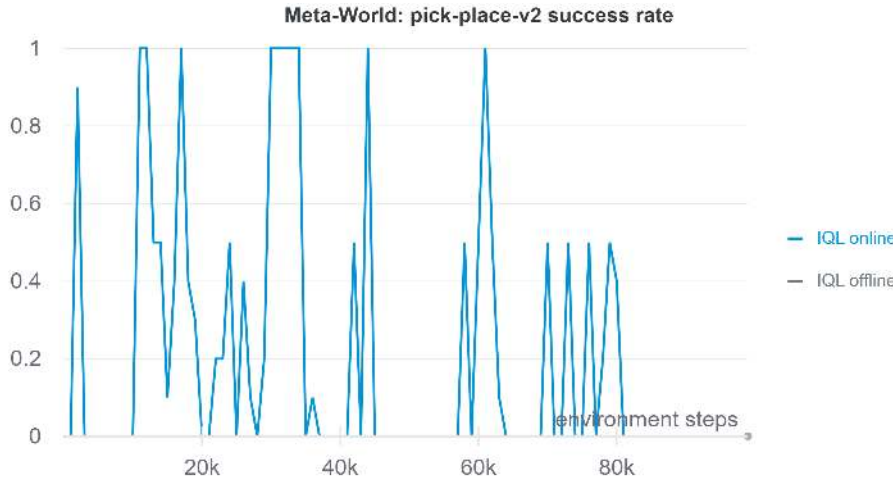


Figure 4: Online vs Offline IQL Performance on Sawyer Robot Task Pick-Place-v2 on Meta-World.

We then tried to do online imitation learning with just pick-place-v2 (see figure 4). While offline IQL completely fails at the task, the online variation shows some level of success, although not robust.

In particular, we note that online-training does hold potential - we hypothesise that some tweaks to the algorithm or some sort of stabiliser (regularisation, for example) will considerably help in training. For the meta-learning benchmarks, we stuck with the online IL setting.

6.2 Meta Learning Results

PointEnv We evaluated our algorithms on the PointEnv environment to be confident in our implementations. In particular, we focused on iQLMAML and compared with two forms of loss value_expert and v0 loss to MAML with Behavioral Cloning and TRPO, and just MAML with TRPO (the latter learning only from online experience). The success rate of the algorithms are shown in Figure 5. We see that MAML with BC and TRPO perform the best (we note that Negative Log Likelihood loss worked better than Mean Square Error for the Behavioral Cloning part), and with MAML and just TRPO perform the worst. iQLMAML, especially with the value_expert loss, performs slightly better than MAML TRPO.

Meta-World We compared MAML with TRPO (learning only from online experience), MAML with BC and TRPO (with Negative Log Likelihood as well as Mean Square Error loss in the BC part), and iQLMAML with value_expert loss on the reach-v2 ML1 benchmark in Figure 6 and the pick-place-v2 ML1 benchmark in Figure 7.

We see that for the reach-v2 task, the BC-based algorithms outperform the IQL-based algorithms in the initial steps of the training run. However, we weren’t able to complete full training due to a memory leak bug in the iQLMAML algorithm at runtime. We haven’t been able to detect and fix the bug in time for the final project report.

Just solely focusing on the first 1 million steps, we see that MAML with BC types still strongly outperform iQLMAML algorithm. However, we are curious if more training steps would result in close or better performance for iQLMAML with MAML(BC+TRPO) with mean squared error loss, and we aim to focus more on the memory leak issue in future work.

Overall, we do not see any indication that IQL with implicit MAML offer any tangible benefits other than being more resource efficient (at least in theory) than standard MAML with BC.

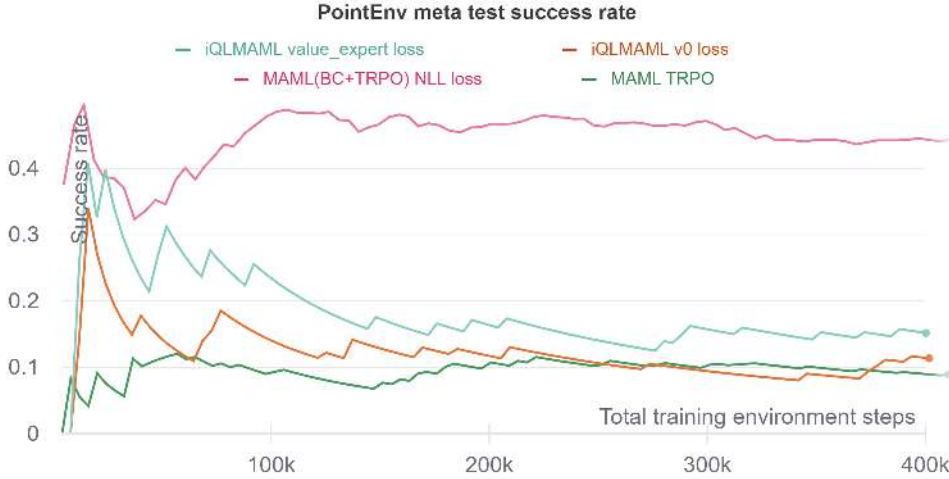


Figure 5: Meta test performance (smoothing value=0.99) across sampled goals in the toy PointEnv environment, measured as a function of environment steps consumed by the training process.

We ran IQL² on single-tasks and ML1 from Meta-World. The chosen tasks were pick-place-v2, sweep-v2, and reach-v2. We also experimented with different IQ-learn loss functions such as v0 and value-expert loss.

Training the recurrent actor was tricky, and resulted in low eval success rate for the pick-place-v2 and sweep-v2 tasks (consistently 0), but had a much higher success rate on the reach-v2 task. Moreover, for all three environments, after an initial dip in reward within the first few episodes, train reward and

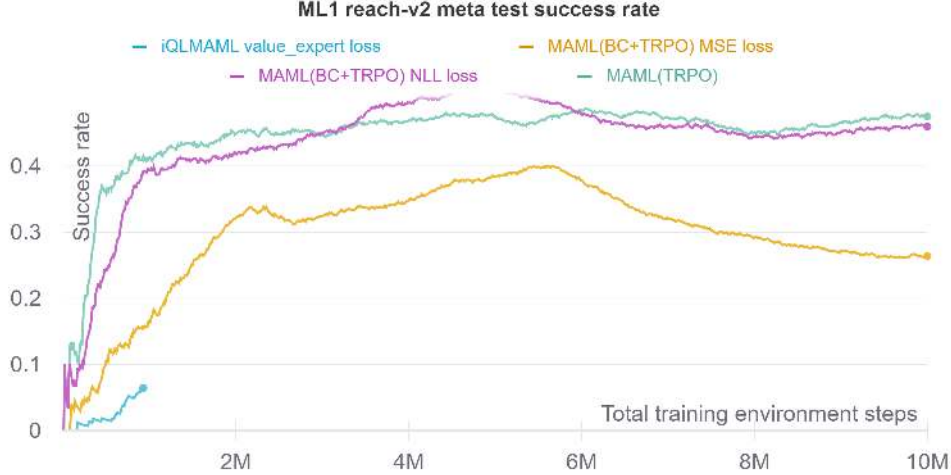


Figure 6: We show how the Meta-learning algorithms perform on the success rate (smoothing value=0.99) for the reach-v2 task in the meta world, where we sample different goals.

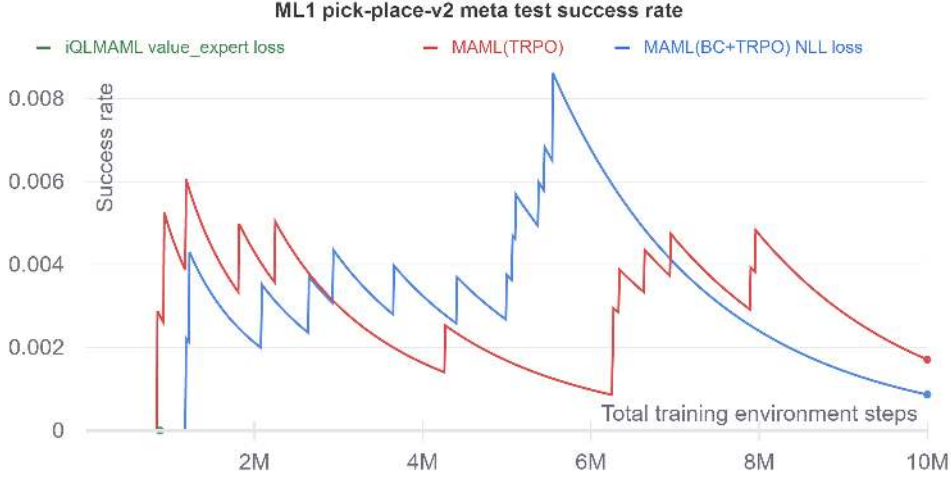


Figure 7: Shows how meta-learning algorithms perform on the success rate (smoothing value = 0.99) for the pick-place-v2 task in meta world.

eval reward tended to have a positive trend. We also found that value-expert loss always outperformed v0 loss.

Our best results were on the reach-v2 Meta-World environment, and the eval MW success is shown in Figure 8.

7 Conclusions and Future Work

We were not able to find hyper-parameters for which IQ-Learn performs complex robotics tasks successfully, especially in the offline case. Even though the online setting of IQ-Learn did perform relatively better, without a robust criteria, it didn't perform well in the single-task setting. However, we note that online IQ-Learn is definitely an avenue that warrants further exploration, and further fine-tuning to the algorithm to establish more stable training will go a long way in solving complex robotic tasks.

We do think that Q-learning still poses significant challenges to work effectively with MAML, but to the best of our knowledge this work features the first implementation of Q-learning with implicit MAML. Complexity of the implementation, as well as latency imposed by the Conjugate Gradient

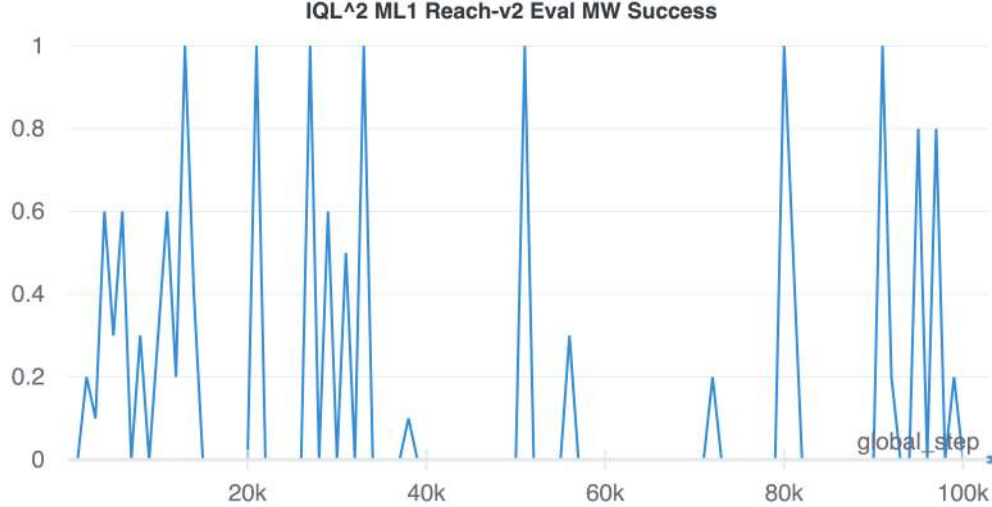


Figure 8: Reach-v2 Eval MW Success (smoothing value = 0) for IQL² applied on ML1. IQL² performed the best on reach-v2 out of the 3 chosen environments.

solver, suggests that there is still a lot of potential for inventing new optimization-based meta-learning methods for Q-learning.

Furthermore, we see that IQL² also faces a lack of a robust stopping criteria, and suffers through unstable training - the evaluation set never converges in its success rate. We recognize the need for stronger imitation learning algorithms that can perform in complex robotics environments.

Moving forwards, we personally will evaluate our imitation learning algorithms against such a RoboSuite/RoboMimic baseline as it is more accurate reflection of imitation learning limitations than standard Cartpole/Acrobat/Gym environments.

7.1 Future Work and Directions

We note that convolutional neural networks have enjoyed a boom from 2010, and with hardware acceleration and recent promising results such as a DQN learning to play Atari from just pixels makes us wonder about the use of images in imitation learning. Going back to the challenge of real-life reinforcement learning, we hypothesize that using high dimensional data such as RGB images to capture the robot environment at each state and developing an imitation learning algorithm to learn from robot poses might perform well. In particular, we are excited about the prospect of developing a learning-based controller for complex robots with images as its input (Soham is currently working on a variant of this idea for his thesis).

We do think that IQ-Learn has promise in its online version, and potentially an area to adapt IQ-learn is to use raw images as its expert demonstration, and potentially combine with existing state of the art control methods such as Model Predictive Control to come up with a combined method to tackle complex robotics.

We also note that many of these tasks in meta-world are single task and often do not worry about trajectory control. Many situations such as 3D printing, or moving in tightly confined spaces such as in clinical settings, or human collaborative environments require robots to be more precise in their movements. At some point, evaluating trajectory mapping and control is a goal of ours when it comes to learning imitation algorithms - essentially asking not only if it solves the task, but how well does it solve the task (with the least time, movement, etc.)

Furthermore, we posit the opposite meta-learning challenge - given a single task, can a robot learn to adapt itself if its sensors are also dynamically placed in different location?

8 Team Contributions

We adjusted the team contributions based on the shifting directions of the project. Initially Soham and Amal focused on Robomimic, while Sebastian focused on Meta-World, but we decided to all shift focus completely to Meta-World in the second half of the course. We also came up with new and more specific algorithms to try, such as IQL² and MAML(TRPO), hence the reallocation of the workload.

Sebastian Kochman implemented code extracting Meta-World demonstrations for IQ-Learn, integrated with the garage toolkit, implemented MAML(BC+TRPO), ran Meta-World and PointEnv Experiments using the aforementioned algorithms.

Amal Mehta ran single-task IQ-Learn Meta-World experiments, implemented and ran experiments for IQL², assisted with iMAML, contributed to writeups.

Soham Sinha wrote and implemented iQLMAML, ran Meta-World and Point Env experiments on iQLMAML, assisted with IQL², and initially implemented IQ-Learn in RoboSuite.

9 Acknowledgements

Divyansh Garg came up with the original project idea (applying IQL to the RoboMimic benchmark) and he graciously agreed to advise the team.

10 Code

Private due to proprietary iQ-Learn code by Divyansh Garg: Code Link.

Contact Divyansh Garg for access.

References

- [1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery.
- [2] Christopher R. Dance, Julien Perez, and Théo Cachet. Demonstration-conditioned reinforcement learning for few-shot imitation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2376–2387. PMLR, 18–24 Jul 2021.
- [3] Yan Duan, Marcin Andrychowicz, Bradley C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning, 2017.
- [4] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL²: Fast reinforcement learning via slow reinforcement learning, 2016.
- [5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.
- [6] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *CoRR*, abs/2106.12142, 2021.
- [7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.
- [8] Gillian Hayes and Yiannis Demiris. A robot controller using learning by imitation. *Citeseer*, 676, 06 1995.
- [9] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

- [10] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.
- [11] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.
- [12] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients, 2019.
- [13] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Proceedings of the Ninth International Workshop on Machine Learning*, ML92, page 385–393, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.
- [14] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.
- [15] Julien Perez Théo Cachet and Seungsu Kim. Transformer-based meta-imitation learning for robotic manipulation, 2020.
- [16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- [17] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning, 2016.
- [18] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.
- [19] Allan Zhou, Eric Jang, Daniel Kappler, Alex Herzog, Mohi Khansari, Paul Wohlhart, Yunfei Bai, Mrinal Kalakrishnan, Sergey Levine, and Chelsea Finn. Watch, try, learn: Meta-learning from demonstrations and reward, 2020.