Meta-Learning Q-Function Priors for Efficient Few-Shot Imitation

Alessandro Barro

***Abstract*— Manually designing reward functions is often infeasible, and standard inverse reinforcement learning methods incur high sample complexity by requiring nested RL solves. We propose meta-learning a soft Q-function prior across related tasks to enable few-shot imitation without an explicit reward network nor inner solves. During meta-training, we update task specific Q-parameters on a small set of demonstrations and optimize the initialization to minimize the inverse soft Bellman residual on held out data. At test time, a single gradient step on a few demonstrations yields both an approximate optimal policy and its underlying reward. We demonstrate the efficacy of this approach through a comprehensive experimental pipeline that includes procedurally generated pixel-based tasks, showing faster adaptation, particularly robust generalization, and low variance across tasks.**

I. Introduction and Motivation

The reward function is regarded as the protagonist when it comes to giving shape to the agent-environment interplay in reinforcement learning (RL) [1]. However, manually specifying rewards that accurately reflect decision-making criteria may be infeasible for certain tasks and prone to bias [2].

To overcome this, the inverse reinforcement learning (IRL) framework comes with methods that aim to recover a reward given observed, optimal behavior [3]. On the other hand, these observations typically require to cover an extensive chunk of the state–action space and suffer from high sample complexity, since each update involves solving a RL problem to evaluate the policy under the current reward estimate.

Recent advancements in imitation learning (IL) [4] have shown that, under certain assumptions, it is possible to derive both optimal policies and reward functions from a learned Q-function [5]. In particular, optimizing the latter provides a dynamically-aware way to completing the imitation process, enabling the agent to generalize more effectively across states by inferring the underlying intent of expert behavior.

In this paper, we relocate this concept to the few-shot domain [6], in which the agent has to adapt to a newly seen task with a limited number of adaptation steps, leveraging consolidated knowledge. This one takes the form of a meta-learned shared parametrization for the Q-function, across a distribution of structurally equivalent tasks. With this, we mean that the considered tasks share a meaningful portion of the state-action space, so that learning one task transfers knowledge to another (please, refer to Fig. 1) [7].

Fig. 1. LoL, the quintessential multitasking game, divides gameplay into distinct reward scenarios and policies while sharing state and action spaces.

Our approach departs from prior methods, such as MandRIL [8], which depend on an explicit reward network. Employing such a network can produce high-variance gradients and incur prohibitive computational costs, since it must be optimized within a double loop requiring nested RL solves, an overhead our method avoids. In fact, by inheriting the objective from IQ-learn, we can extract both optimal policy and reward after just few gradient steps on Q-values.

The intuition behind the proposed method comes from common exhibitions of human intelligence: this one is capable of inferring more or less rational strategies after just "a taste" of experience with a new task. This ability often relies on familiarity with similar tasks, and thus on prior knowledge of shared environment dynamics that can transfer to new instances. Learning a start (or prior) over the Q-function, which encodes both agent behavior (policy) and environmental feedback (reward), is then a natural choice for modeling this form of intelligence, one that is particularly valuable, even in practice.

II. Previous Work

IRL seeks to infer a reward function from expert demonstrations such that a corresponding optimal policy reproduces the expert's behavior. Early approaches include margin-based apprenticeship learning (matching feature expectations) [3]. A key advance was maximum entropy IRL [9], which frames IRL as probabilistic inference, yielding a globally normalized distribution over trajectories, accommodating suboptimal expert behavior and offering performance guarantees comparable to earlier methods.

Building on IRL, generative adversarial imitation learning (GAIL) [10] introduced an adversarial setup: a discriminator distinguishes expert versus agent behavior, while the policy learns to fool it by matching occupancy measures. GAIL

became widely used for its strong results, but suffers from unstable, sample-inefficient minimax training.

On a similar wavelength, Adversarial Inverse Reinforcement Learning (AIRL) [11] adapts GAIL's discriminator into a partition-function–based reward estimator that's disentangled from transition dynamics, yielding invariant rewards that transfer across environments. Agents trained with AIRL show markedly better reward recovery and transfer performance than earlier IRL methods [11].

Next, IQ-Learn [5] leverages a single soft Q-function to jointly model both reward and policy, recasting the adversarial minimax IRL problem into a single concave maximization with convergence guarantees. This non-adversarial formulation yields stable training and achieves state-of-the-art results in offline and online imitation, requiring over three times fewer environment interactions than prior methods and recovering rewards that closely match true rewards (e.g. Pearson correlation of 0.99 on Hopper).

Meanwhile, meta learning enables few-shot adaptation. Model agnostic meta learning (MAML) [12] trains parameters that can be rapidly fine tuned to new tasks with few gradient steps. This inspired meta-IL methods, often built on behavioral cloning (e.g. meta-DAgger [13]) due to its simplicity, despite IRL often providing better performance. To address this, SMILe [14], a meta-IRL method based on max-entropy IRL, learns context-conditioned policies that adapt quickly from limited demonstrations.

A related line of work seeks to meta-learn rewards or intentions behind expert behavior. For example, Xu et al. [8] use meta-IRL to learn a reward function prior, capturing shared structure across tasks in a latent reward space. With the transferable Q-function prior, our work is situated at the intersection of the latter meta-strategy and inverse soft Q-learning, overall contributing to the meta-IL paradigm [15].

III. Problem formulation

In our multi-task few-shot learning setting, each task $\mathcal{M}_i$ is a Markov decision process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote state and action spaces respectively, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ indicates the transition dynamics, $p_0 : \mathcal{S} \mapsto [0, 1]$ is the initial state distribution and $\gamma \in [0, 1]$ is the discount factor. In RL the goal is to learn a policy $\pi(a \mid s)$ that maximizes the expected total return [16]. In our framework, we adopt the MaxEnt return, as this guarantees that later IRL definitions are well behaved [17]

$$J(\pi) = \mathop{\mathbb{E}}_{\rho_\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\Big] + \alpha\mathcal{H}(\pi) \tag{1}$$

where $\mathcal{H}(\pi) = -\mathbb{E}_{\rho_\pi}[\log \pi(a \mid s)]$ denotes the policy entropy, ρ_π is the state-occupancy measure of $\pi(a \mid s)$ and $\alpha > 0$ is the temperature parameter (which we will omit from now on). We then introduce the soft Bellman equation

$$B^\pi Q^\pi(s, a) = r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(\cdot|s,a)}\big[V^\pi(s')\big] \tag{2}$$

where $V^\pi(s) = \log \sum_{a'} \exp\big(Q^\pi(s, a')\big)$. The operator B^π is a contraction and characterizes a unique soft Q-function $Q(s, a)$ for a given $r(s, a)$. This also means that at optimality $B^\pi Q^\star(s, a) = Q^\star(s, a)$ holds, with $Q^\star \in \Omega$. The respective optimal policy $\pi^\star \in \Pi$ can be recovered as an energy-based model of Q

$$\pi^\star(a \mid s) = \frac{1}{Z(s)} \exp\big(Q^\star(s, a)\big) \tag{3}$$

where $Z(s) = \sum_{a'} \exp\big(Q^\star(s, a')\big)$ is the normalization factor. Given the invertibility of B^π in the MaxEnt setting, inverse soft Q-learning tells us that $r \in \mathcal{R}$ can be rewritten in terms of $Q(s, a)$, and the IRL loss is given by **(A1)**

$$\begin{aligned}\max_{Q \in \Omega} \hat{J}(Q) = \mathop{\mathbb{E}}_{\rho_E}\Big[\psi\Big(Q(s, a) - \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(\cdot|s,a)}[V^\star(s')]\Big)\Big] \\ - (1 - \gamma) \mathop{\mathbb{E}}_{s_0 \sim p_0}\big[V^\star(s_0)\big]\end{aligned} \tag{4}$$

a single, concave one. Here, $\psi(x)$ is a concave function that shapes the residual to implicitly incorporate the divergence constraint between the learned policy and the expert.

Let $\mathcal{T}_i$ denote a dataset comprising N demonstrations τ for $\mathcal{M}_i$, which is drawn from a common task distribution $p(\mathcal{M})$ that we assume to be known. We additionally define $\phi_i \in \Phi$ as the Q-function parameters for i-th task. The idea behind meta-RL is to learn an algorithm

$$f(\mathcal{T}_i) : \big((\mathcal{S} \times \mathcal{A})^{K \times T}\big)^N \mapsto \Phi \tag{5}$$

to devise an initialization $\theta = \phi^{(0)}$ across tasks [18]. In our case, we can think of learning $f(\mathcal{T}_i)$ as paradigm to transform a cluster of demonstration datasets into IL algorithms, that in their turn learn a policy.

Let Q_{ϕ_i} denote the Q-network [19] and δ_{ϕ_i} the Bellman residual: extending (4) to the MAML formalism gives the following meta-gradient

$$\begin{aligned}\nabla_\theta \hat{J}(Q_\theta) = \sum_{\mathcal{M}_i \sim p(\mathcal{M})} (I - \eta \nabla^2_\theta \hat{J}_{\mathcal{T}_i}(Q_\theta))^\top (\mathbb{E}_{\rho_E}[\psi'(\delta_{\phi_i}) \\ \nabla_{\phi_i} Q_{\phi_i}(s, a)] - \mathbb{E}_{\rho_\pi}[\psi'(\delta_{\phi_i}) \nabla_{\phi_i} Q_{\phi_i}(s, a)])\end{aligned} \tag{6}$$

where $\mathcal{V}_i$ refers to unseen (or validation) data for task $\mathcal{M}_i$, $\nabla^2_\theta \hat{J}$ is the hessian of $\hat{J}$ w.r.t. θ, with η being the inner step size. In practice, we can use the first order approximation and drop the $I - \eta \nabla^2_\theta \hat{J}_{\mathcal{T}_i}(Q_\theta)$ term for efficiency **(A2)**. Thereafter, $f(\mathcal{T}_i)$ is implemented by setting $\phi_i^{(0)} = \theta$ and performing K adaptation gradient descent steps

$$f(\mathcal{T}_i) : \ \phi_i(\theta) = \theta + \eta \, \nabla_\theta \hat{J}_{\mathcal{T}_i}\big(Q_\theta\big) \tag{7}$$

This without any interleaved RL solve: each step is just a gradient on Q_{ϕ_i}. Finally, the policy $\pi_{\phi_i}(a \mid s) \propto$

$\exp\left(Q_{\phi_i}(s,a)\right)$ is implicitly recovered, while the reward function can be extracted via inverse Bellman operator T^π

$$r_{\phi_i}(s,a) \approx \delta_{\phi_i} = T^\pi Q_{\phi_i}(s,a) \tag{8}$$

$$= Q_{\phi_i}(s,a) - \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(\cdot|s,a)} \left[V_{\phi_i}(s')\right] \tag{9}$$

For state only reward retrieval **(A3)**, or for continuous control **(A4)** implementation details, we remind the reader to the appendix.

IV. Practical Algorithm

We implement our method as shown in Algorithm 1. In line 10 we use the first order approximation, i.e. we take the same gradient but evaluate it on held-out demos. Retaining the second derivative term instead injects curvature information on top of that direction shift.

Algorithm 1 Meta-Learning a Q-Function Prior

Require: Task distribution $p(\mathcal{M})$;

1: support demos per task K, validation demos per task K';

2: inner steps L, inner learning rate η; outer learning rate ν;

3: batch size B

Ensure: Meta-initialization θ for Q-network Q_θ

4: Initialize $\theta \sim \mathcal{N}(0, I)$

5: **while** not converged **do**

6: Sample tasks $\{\mathcal{M}_i\}_{i=1}^{B} \sim p(\mathcal{M})$

7: **for** $i = 1, \ldots, B$ **do**

8: Collect support set $\mathcal{T}_i = \{(s,a,s')\}_{j=1}^{K}$

9: Collect validation set $\mathcal{V}_i = \{(s,a,s')\}_{j=1}^{K'}$

10: $\phi_i \leftarrow \theta$

11: **for** $l = 1, \ldots, L$ **do**

12: Compute inner loss:

$$\mathcal{L}_{\mathcal{T}_i}(\phi_i) = - \sum_{(s,a,s') \in \mathcal{T}_i} \psi\big(Q_{\phi_i}(s,a) - \gamma V_{\phi_i}(s')\big)$$

13: Update: $\phi_i \leftarrow \phi_i - \eta \nabla_{\phi_i} \mathcal{L}_{\mathcal{T}_i}(\phi_i)$

14: **end for**

15: **end for**

16: Compute meta-gradient:

$$\ell_{\mathcal{V}_i} \leftarrow - \sum_{i=1}^{B} \nabla_\theta \sum_{(s,a,s') \in \mathcal{V}_i} \psi\big(Q_{\phi_i}(s,a) - \gamma V_{\phi_i}(s')\big)$$

17: Meta-update: $\theta \leftarrow \theta - \nu\, \ell_{\mathcal{V}_i}$

18: **end while**

19: **return** θ

V. Experiments Plan

To validate our approach, we choose to adopt the same evaluation protocol of Xu et al. [8], and benchmark against two major baselines:

- **From Scratch**: each task is trained independently from a random initialization.
- **Average Gradient**: obtained via supervised pretraining across tasks.

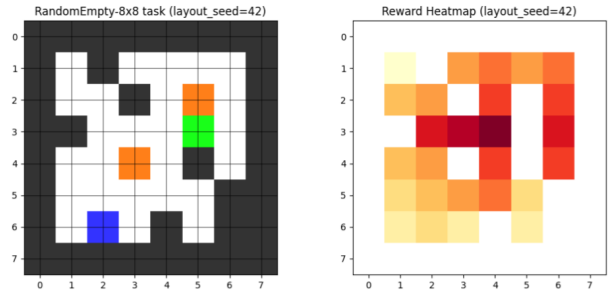

Fig. 2. Example map (on the left, agent is blue and goal is green) of the `MiniGrid-RandomEmpty-8x8-v0` task, Model estimated reward heatmap (on the right, darker color is higher reward)

We assess performance primarily using the expected value difference (EVD), a dependable IRL metric [20]:

$$\mathop{\mathbb{E}}_{\rho_E}\Big[\sum_{t=0}^{\infty} r(s_t,a_t)\Big] - \mathop{\mathbb{E}}_{\rho_\pi}\Big[\sum_{t=0}^{\infty} r(s_t,a_t)\Big] \tag{10}$$

Furthermore, we also measure the Pearson correlation between environment rewards r and estimated ones r_{ϕ_i}, throughout the rollout of a fixed policy.

We choose to infer reward functions directly from raw pixel observations, and to ensure that tasks share a common structure while still exhibiting meaningful variation, we adopt the **MiniGrid** suite [21], and procedurally generate grid-world environments. MiniGrid provides a lightweight, Gym-compatible interface in which the agent observes a small RGB patch of its local surroundings, yet must learn to navigate and solve a variety of objectives that depend on the layout.

Our pipeline focuses on navigating within a custom environment `MiniGrid-RandomEmpty-8x8-v0`, which is created from `MiniGrid-Empty-8x8-v0`. In each task generation epoch, the initialization function samples a configuration of obstacles, walls and lava, thereby fixing a specific grid arrangement that defines a single task $\mathcal{M}_i$. Each task is therefore identified by a layout seed, which remains constant for all demonstrations relating to that task. To collect proofs, the agent is repositioned to K different free spawn cells, determined by a distinct spawn seed, and a trajectory of interactions is executed using an expert policy.

VI. Results and Discussion

In our experiments, we collected 550 demonstration datasets $(\mathcal{T}_i, \mathcal{V}_i)$ (500 for training, 50 for testing) by instantiating the tasks as aforementioned. We utilized a Q-Network with three Conv2D blocks. Within each inner loop, we performed one step of SGD on the inner loss, and with $K = 5$ demonstrations. After adaptation, we evaluated the adapted Q-network on the same number of held-out demos to compute the outer loss **(A5)**.

As for the testing phase, we implemented the adaptation

$f(\mathcal{T}_i)$ as 10 SGD steps with learning rate $\eta = 1 \cdot 10^{-3}$ on the single task's loss. All the other hyperparameters were kept the same, and no specific fine-tuning routine was conducted.

	From Scratch	Average Gradient	Meta Q-prior
EVD	0.75 ± 0.10	0.47 ± 0.13	$\mathbf{0.32 \pm 0.14}$
Correlation	0.32 ± 0.15	0.35 ± 0.11	$\mathbf{0.64 \pm 0.06}$

Tab. 1. EVD and correlation with different model initializations in the custom `MiniGrid-RandomEmpty-8x8-v0` environment.

Tab. 2 shows that a meta-learned Q-function prior offers a decisive adaptation advantage: after just a few gradient steps it achieves the lowest EVD (0.32 ± 0.14) and the highest Pearson correlation with the ground-truth rewards (0.64 ± 0.06), almost doubling the Average Gradient warm start (0.35 ± 0.11).

These numbers confirm that the prior successfully captures task-agnostic dynamics; when transplanted to a new 8×8 MiniGrid layout it needs minimal data to reconstruct a reward signal that closely matches the environment and drives a near-optimal policy, whereas the baselines either converge slowly or overfit (see the adaptation trends below).

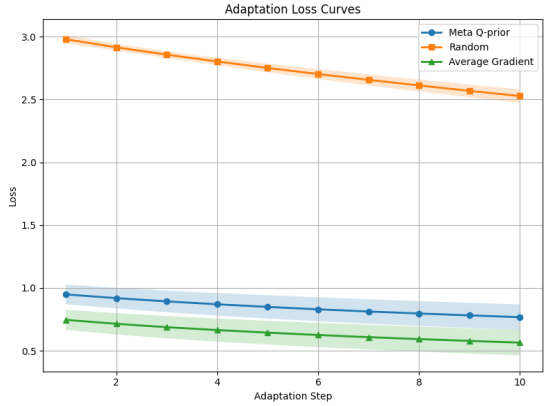

Fig. 3. Adaptation loss curves for different model initialization

VII. Conclusion

We presented a meta-learning framework that leverages inverse soft Q-learning to distill a soft Q-function prior. This prior effectively adapts to various algorithms, simultaneously recovering both reward and policy without the necessity of explicit reward models, occupancy measure matching, or computationally expensive inner RL loops. Our MAML-based methodology is computationally efficient compared to occupancy-matching approaches like MandRIL, thus offering a practical solution.

A concise evaluation on MiniGrid tasks shows our meta-learned prior adapts rapidly, outperforming pretrained and random initializations within just a few gradient updates, while maintaining low variance and strong generalization across procedurally generated scenarios. By unifying policy adaptation with implicit reward inference in a single Q-function prior, the framework delivers interpretable, scalable performance in high-dimensional, pixel-based environments.

Future research directions include evaluating the framework in more complex environments and continuous control scenarios. Immediate next steps should involve thorough experimental comparisons with established imitation learning, meta-IRL, as well as conventional IRL methods.

VIII. Appendix

(A1) From MaxEnt inverse reinforcement learning to inverse soft Q-Learning Assume that we are given demonstrations with policy E that follow the MaxEnt return, then MaxEnt IRL allows us to recover a reward function $r(s,a)$ in the functional space $\mathcal{R} \subseteq \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ by optimizing the following objective

$$\max_{r\in\mathcal{R}} \min_{\pi\in\Pi} L(\pi, r) = \mathbb{E}_{\rho_E}[r(s,a)] - \mathbb{E}_{\rho_\pi}[r(s,a)] - \mathcal{H}(\pi) - \varphi(r) \tag{11}$$

where $\varphi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \mapsto \overline{\mathbb{R}}$ is a convex regularizer that restricts $r(s,a)$ to another set $\overline{\mathbb{R}}$. In practice this allows to exclude ill-defined reward functions such as those yielding an unbounded objective. Next, we can re-formulate the above objective with a statistical distance metric measured between E and π, which is also parametrized by φ.

$$\max_{r\in\mathcal{R}} \min_{\pi\in\Pi} L(\pi, r) = \min_{\pi\in\Pi} d_\varphi(\rho_\pi, \rho_E) - \mathcal{H}(\pi) \tag{12}$$

Here, $d_\varphi = \varphi^\star(\rho_E - \rho_\pi)$, and $\varphi^\star$ is the convex conjugate of φ. In practice, the regularizer φ_g is formed with an additional function $g : \mathbb{R} \mapsto \overline{\mathbb{R}}$ defined as

$$g(x) = \begin{cases} x - \psi(x) & \text{if } x \in \mathcal{R}_\varphi \\ \infty & \text{otherwise} \end{cases} \tag{13}$$

where $\mathcal{R}_\varphi \subseteq \mathcal{R}$ is the constrained set of reward functions and $\psi : \mathcal{R}_\varphi \mapsto \mathbb{R}$ is a concave function. The choice of the regularizer φ_g is useful to guarantee properties such as limitedness, smoothness, etc., and ultimately leads to d_φ and ψ. The preferred choice, which is also the one employed in this work, is $\varphi_g(r) = \lambda r^2$ that results into the χ^2-divergence $d_\varphi(\rho, \rho_E) = \frac{1}{4\lambda}\chi^2(\rho, \rho_E)$ and $\psi(x) = x - \frac{x^2}{4\lambda}$.

It can be shown that, for a fixed $\pi \in \Pi$ the inverse soft Bellman operator T^π is bijective, and $\forall\ r \in \mathcal{R}$, $\exists!\ Q = T^{-\pi} r$ fixed point of the Bellman operator B^π. Thus, once we define

$$\hat{J}(\pi, Q) = \mathop{\mathbb{E}}_{\rho_E}[T^\pi Q^\pi] - \mathop{\mathbb{E}}_{\rho_\pi}[T^\pi Q^\pi] - \mathcal{H}(\pi) - \varphi(T^\pi Q^\pi) \tag{14}$$

we can see that $L(\pi, r) = \hat{J}(\pi, T^{-\pi} r)$ and equivalently $\hat{J}(\pi, Q) = L(\pi, T^\pi Q)$. Moreover, given that

$$\mathop{\mathbb{E}}_{\rho_\pi}[T^\pi Q^\pi] = -\mathcal{H}(\pi) + (1-\gamma) \mathop{\mathbb{E}}_{s_0\sim p_0}[V(s_0)] \tag{15}$$

we derive the objective function (4) in π and Q. Finally, in the MaxEnt setting, since $\pi(a \mid s) \propto \exp\left(Q(s,a)\right)$, we get

$$\max_{Q\in\Omega} \min_{\pi\in\Pi} \hat{J}(\pi, Q) = \max_{Q\in\Omega} \hat{J}(Q) \tag{16}$$

We remind the reader to the IQ-Learn paper for detailed proofs and theoretical results of this framework.

(A2) Derivation of the inverse soft Q-Learning meta-gradient Consider a Q-network Q_θ of parameters θ, and demonstrations subdivided into train $\mathcal{T}_i$ and validation $\mathcal{V}_i$ for the task $\mathcal{M}_i$. We denote with $\phi_i \in \Phi$ the parameters obtained by adapting θ to some task, which is done through (8). Following MAML, the outer objective is given by

$$\max_{\theta} \sum_{\mathcal{M}_i \sim p(\mathcal{M})} \hat{J}_{\mathcal{V}_i}\left(Q_{\theta + \eta\, \nabla_\theta \hat{J}_{\mathcal{T}_i}\left(Q_\theta\right)}\right) \tag{17}$$

Thus, we can proceed by integrating the objective function in (4) with the above expression to obtain the meta-gradient. For a single task, we have

$$\nabla_\theta\ \hat{J}_{\mathcal{V}_i}\ =\ \left(\frac{\partial \phi_i}{\partial \theta}\right)^\top\ \nabla_{\phi_i}\ \hat{J}_{\mathcal{V}_i}(Q_{\phi_i}) \tag{18}$$

By differentiating the algorithm devised in (8) which, recall, coincides with the adaptation step, we get a closed form expression for the jacobian

$$\frac{\partial \phi_i(\theta)}{\partial \theta}\ =\ I\ +\ \eta\, \nabla^2_\theta\ \hat{J}_{\mathcal{T}_i}(Q_{\phi_i}) \tag{19}$$

where $\nabla^2_\theta\ \hat{J}_{\mathcal{T}_i}$ is the hessian of $\hat{J}$ w.r.t. θ. Provided the empirical Hessian is non-singular in the neighbourhood of θ, $I + \eta\ \nabla^2_\theta\ \hat{J}_{\mathcal{V}_i}$ is full-rank. In practice we ensure this numerically (damping) rather than relying on theoretical rank guarantees. Substituting into the single level objective $\partial\phi_i(\theta)/\partial\theta$, we get

$$\nabla_\theta \hat{J}_{\mathcal{V}_i} = \left(I + \eta\, \nabla^2_\theta\ \hat{J}_{\mathcal{T}_i}\right)^\top \ell_{\mathcal{V}_i}, \quad \ell_{\mathcal{V}_i} = \nabla_{\phi_i} \hat{J}_{\mathcal{V}_i}(Q_{\phi_i}) \tag{20}$$

Let's differentiate $\hat{J}$ w.r.t. Q

$$\frac{\partial \hat{J}}{\partial Q(s,a)} = \left[\rho_E(s,a) - \rho_\pi(s,a)\right] \psi'\big(\delta_{\phi_i}(s,a)\big) \tag{21}$$

Thus, in expectation notation

$$\ell_{\mathcal{V}_i} = \mathop{\mathbb{E}}_{\rho_E}\big[\psi'(\delta_{\phi_i})\, \nabla_{\phi_i} Q_{\phi_i}(s,a)\big] - \mathop{\mathbb{E}}_{\rho_\pi}\big[\psi'(\delta_{\phi_i})\, \nabla_{\phi_i} Q_{\phi_i}(s,a)\big] \tag{22}$$

where the state occupancy measures are related to $\mathcal{V}_i$ and the policy is parametrized by ϕ_i. In conclusion, summing over the meta-batch results in (5).

(A3) Retrieval of state-only rewards with marginalization State-only rewards cannot be simply retrieved once the model is trained, they need to come from a modified objective, which also marginalizes over actions.

$$\max_{Q\in\Omega} \hat{J}(Q) = \mathop{\mathbb{E}}_{\rho_E}\Big[\ \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\ \Big[\ \psi\Big(\ Q(s,a) - \tag{23}$$

$$\gamma\ \mathop{\mathbb{E}}_{s'\sim\mathcal{P}(\cdot|s,a)}\big[V^\star(s')\big]\Big)\Big]\Big] - (1-\gamma)\ \mathop{\mathbb{E}}_{s_0\sim p_0}\big[V^\star(s_0)\big]$$

Here, π is only used to sample actions, but does not contribute to the gradient's computation. This objective the be used for observation only IL, as it doesn't rely on π_E.

(A4) Extension to the continuous control setting In environments with continuous action spaces the normalising constant $Z(s)$ is generally intractable. To avoid repeatedly approximating the latter, we introduce an auxiliary stochastic policy $\pi_\xi(a\,|\,s)$, parameterised by ξ, that serves as a surrogate for the exact energy-based policy π.

Consequently, the inner objective on the critic Q remains identical to (4), but each inner-loop iteration is now augmented with a soft actor–critic (SAC) update. For a fixed critic Q and a given task $\mathcal{M}_i$ we solve

$$\max_{\pi_\xi}\ \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D}_i \\ a\sim\pi_\xi(\cdot|s)}}\big[\,Q(s,a) - \log \pi_\xi(a \mid s)\big] \tag{24}$$

where $\mathcal{D}_i$ denotes the task-specific replay buffer (or state distribution). The entropy term encourages exploration and simultaneously drives π_ξ towards the policy induced by Q while remaining properly normalised and differentiable. Specifically, in Algorithm 1 we

1) initialize an additional step size for the actor
2) insert a gradient update on ξ after line 7.

(A5) Further experimental details All experiments were implemented in PyTorch and used the MiniGrid environment via the Gym interface. Demonstrations were generated using a BFS expert and varying spawn seeds for K times (both for support and query datasets). Training and adaptation relied on a convolutional Q-network (whose architecture is described below) and MAML optimization with the higher library. A summary table follows.

Parameter / Setting	Value
Number of Tasks (Train / Test)	500 / 50
Demonstrations per Task (K)	5
Inner Learning Rate (η)	$2e-5$
Outer Learning Rate (ν)	$2e-5$
Adaptation Learning Rate	$1e-3$
Discount Factor (γ)	0.99
Temperature (α)	0.5
Loss Regularization (λ)	1.0
SGD Steps (Inner Loop)	3
SGD Steps (Test-Time Adaptation)	10
Q-network Architecture	3L Conv2D-BN-ReLU + 2L MLP head
Total Parameters	151015

Tab. 1. Hyperparameters and other details about training and test conduction.

Our code is available at the following GitHub repository https://github.com/alessandrobarro/meta-q.git.

REFERENCES

[1] J. Eschmann, *Reward Function Design in Reinforcement Learning*, pp. 25–33. Cham: Springer International Publishing, 2021.

[2] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016.

[3] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, (San Francisco, CA, USA), p. 663–670, Morgan Kaufmann Publishers Inc., 2000.

[4] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 6322–6337, 2022.

[5] D. Garg, S. Chakraborty, C. Cundy, J. Song, M. Geist, and S. Ermon, "Iq-learn: Inverse soft-q learning for imitation," 2022.

[6] H. Gharoun, F. Momenifar, F. Chen, and A. H. Gandomi, "Meta-learning approaches for few-shot learning: A survey of recent advances," *ACM Comput. Surv.*, vol. 56, July 2024.

[7] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13344–13362, 2023.

[8] K. Xu, E. Ratner, A. D. Dragan, S. Levine, and C. Finn, "Learning a prior over intent via meta-inverse reinforcement learning," *CoRR*, vol. abs/1805.12573, 2018.

[9] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, "Maximum entropy inverse reinforcement learning.," in *Aaai*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[10] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[11] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, pp. 1126–1135, PMLR, 2017.

[13] A. E. Sallab, M. Saeed, O. A. Tawab, and M. Abdou, "Meta learning framework for automated driving," *arXiv preprint arXiv:1706.04038*, 2017.

[14] S. K. Seyed Ghasemipour, S. S. Gu, and R. Zemel, "Smile: Scalable meta inverse reinforcement learning through context-conditional policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Proceedings of the 1st Annual Conference on Robot Learning* (S. Levine, V. Vanhoucke, and K. Goldberg, eds.), vol. 78 of *Proceedings of Machine Learning Research*, pp. 357–368, PMLR, 13–15 Nov 2017.

[16] A. G. Barto, "Reinforcement learning: An introduction. by richard's sutton," *SIAM Rev*, vol. 6, no. 2, p. 423, 2021.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, Pmlr, 2018.

[18] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," *Advances in neural information processing systems*, vol. 24, 2011.

[21] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks," *CoRR*, vol. abs/2306.13831, 2023.