Imperial College London

MSc Computing (Artificial Intelligence & Machine Learning)

Imperial College London

Department of Computing

PERIL: Probabilistic Embeddings for hybrid Meta-Reinforcement and Imitation Learning

Author:
Alvaro Prat Balasch

Supervisor:
Dr. Edward Johns

Second Marker:
Dr. Antoine Cully

September 4, 2020

Abstract

Traditional reinforcement learning algorithms require vast amounts of data collection which is often prohibitive in robotic setups. Moreover, robot learning in this setting results in highly task specific behaviours which require slow, data hungry training from scratch. In contrast, humans have the ability of exploiting past experiences to readily adapt to new conditions. In light of this, given the previous successes of deep reinforcement learning in dexterous manipulation, the field of meta-reinforcement learning (meta-RL) is becoming a major topic of interest in robot learning research. More specifically, probabilistic meta-RL aims to leverage past experiences with the goal of learning how to learn over a limited set of data retrieved from an unseen task.

Despite their success, current methods on meta-RL are highly sample inefficient and are typically conditioned on dense rewards which make them incompatible for real world robot learning. In this project we work on top of existing probabilistic meta-RL by folding learning from demonstrations into the meta-learning framework. To that end, we propose Probabilistic Embeddings for hybrid meta-Reinforcement and Imitation Learning (PERIL). By using probabilistic task beliefs conditioned on imperfect demonstrations, PERIL efficiently explores via posterior sampling to improve its belief of an unseen task. Furthermore, we demonstrate task inference over collected transitions in unseen tasks is directly linked to identification of previously learnt dynamics, thus, our method is capable of reasoning about its uncertainty by relating recent experiences through long-term memory triggers.

Extensive analysis over a set of multi-task and meta-transfer learning benchmarks validate that our approach greatly outperforms previous state-of-the-art methods in terms of sample efficiency and capacity. To the best of our knowledge, PERIL is the first algorithm capable of interpreting demonstrations to perform adaptation to different complex domains under continuous state-action spaces with sparse rewards. In addition, PERIL exhibits superior adaptation performances by performing zero-shot learning in unseen tasks of different dynamics. In light of this we extend on the topics of general artificial intelligence by developing task encoders which can adequately reason about its experience, and with this, condition a single policy to perform multiple tasks.

Our open source implementation of PERIL is available on `www.github.com/alvaroprat97/MetaRobot`.

Acknowledgements

I would like extend my gratitude to Dr. Edward Johns for all the hours of dedication and support. Working from home for the 5 months which define the thesis has imposed multiple practical and communication challenges. Nevertheless, your constant support and guidance has made this possible and for that I am very grateful. I look forward to discussing the results of this project in further detail.

In addition, I would like to thank Dr. Antoine Cully for your availability and your advice whilst co-supervising me for the past 5 months.

I would also like to thank my colleagues and friends Theophile Sautory, Ludovico Mitchener and Sacha Hu for the myriad of hours of enlightened discussions and brainstorming, and wish them the best in their future encounters.

Finally, I dedicate this thesis to my parents Jorge and Ines, as a sign of gratitude for providing countless and unconditional support which have made studying this incredible master possible.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Motivation

Alongside the expanding era of technological advancements, robots are becoming increasingly popular in social and industrial circles [1]. The idea of sentient machines capable of imitating human-like behaviour comes back centuries ago. For instance, Leonardo da Vinci sketched out the design for a humanoid robot in the form of a knight. Leaping a few centuries forward, Shakey, a robot developed by the Stanford Robotics Laboratory in 1972, was the first general-purpose mobile robot which could reason about its own actions [2]. After several decades of research and disruptive innovation, robots nowadays are equipped with competent programmable tools which allow them to adapt to different environments and complete a range of different tasks. Nonetheless, even though most robots are commonly misconceived as humanoid-like entities capable of taking decisions through reasoning, a major share are designed to blindly follow predefined instructions with minimal sensory feedback and no inference methods which help reason about the environment (Close to 500,000 new robots are installed on a yearly basis [3]).

Indeed the success of robotics in basic, monotonous tasks has opened an exceeding demand for robots which are evolved to solve problems of substantial complexity. As such, sectors like medicine, military service, and construction are becoming increasingly dependant on robotics. Leveraging robots to take control of an externalised situation which directly affects our physical world can present a myriad of economical and sociological advantages. Using robots instead, or in conjunction to, human input offers increased safety, scaled production speeds, and in some cases the ability to perform tasks which would be otherwise unattainable due to the vastly superior precision standards offered by contemporary control systems [3].

Other sectors including transportation and logistics are beginning to shift toward incorporating robotic frameworks, exploiting the repeatability and efficiency of single, and quite often multiple, robotic systems working in ensemble. For instance, the surge of autonomous vehicles is purely robot-driven, where sensory information is used to plan a path where the car is steered towards. In addition, robot swarms used in warehouse logistics have showcased the reliability and efficiency of multi-agent robotic systems [4].

Taking a step back from emerging applications, the predominant sector for robotic research falls within the scope of automation and manufacturing. In the latter, automation commonly exploits synergies from advances in computing, particularly machine learning and artificial intelligence, to increase flexibility and adaptability to perturbations: Because the robot is programmed to do a specific task, it must be

resilient to alterations in the environment which might otherwise jeopardise the robot's response. In such conditions, the robot must execute a series of actions depending on its interpretation of the environment, to repeatedly execute the desired task. The latter typically involves tool use or material handling [5].

The field of robotics research is generally twofold. In one hand, Robot Learning research aims at incorporating methods derived from machine learning and artificial intelligence to either aid the robot to perform a task by providing task-specific information [6], or directly provide the tools to let the robot learn how to solve the task autonomously. However, even though the field of Robot Learning is growing exponentially, most deployable robots nowadays still leverage the Traditional Approach: Manually tuning models and complex scripts which function the robot to succeed at performing what more often than not ends up being a highly specific and overfitted task [7]. In the following we will discuss the differences between the two approaches, and why Robot Learning is becoming predominant via the proliferation of advanced computing techniques.

1.1.1 Traditional Approach

Traditional robotic products heavily rely on hand-crafted engineered control systems which are typically devised to perform rudimentary tasks, without an understanding of the intrinsic or extrinsic laws of physics which govern its behaviour or surroundings [8]. These robots are commonly used in closed settings such as that in industrial lines, where the task at hand is purposely monotonic and highly repetitive. An example of typical robotic factory lines is presented in Figure 1.1, where each robot performs an independent yet similar task of spot welding.

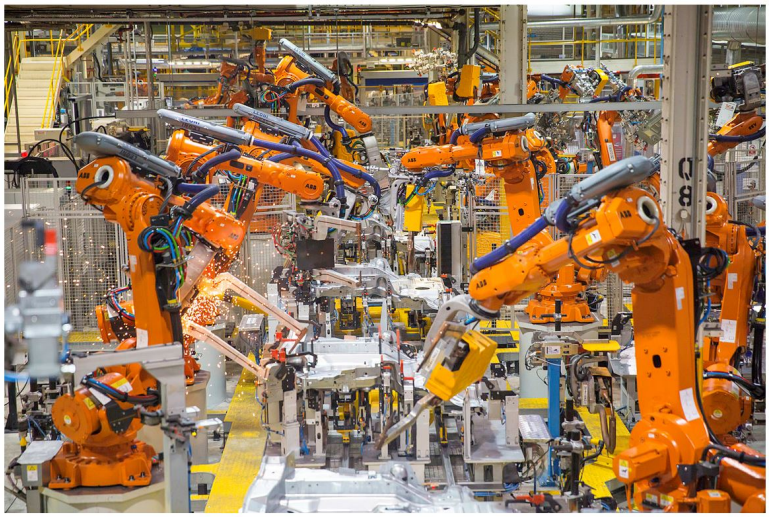

Figure 1.1: Typical setup for robotic manipulators in a factory line. Every robot in the factory line performs the exact same task. Extracted from [9].

Furthermore, advancements in the field of robotics have expanded the application domain by incorporating statistical methods of localisation such as Statistical Localisation And Mapping (SLAM) and the Extended Kalman Filter (EKF) [10, 11], as well as re-adjustable dynamical models leveraging Model Predictive Control (MPC) [12]. Through this, dexterous dynamical robots have also succeeded in other complex applications such as search and rescue, delivery drones, and healthcare [13].

In spite of this, other than requiring months if not years of tedious fine-tuning, a robot can only attempt to adequately perform a task once it has been accurately design to do so. In many cases this requires a deep expertise and understanding of the task it is trying to accomplish [13]. In complex settings where the robot must interact with objects in its surroundings, properly accounting for the complexities of the real world remains an unsolved conundrum [14]. Discontinuities in the environment, deformable objects, kinematic limitations, adverse lighting conditions, and pose uncertainty (proprioceptive uncertainty of the robot's position) are amongst a myriad of other factors which challenge manual tuning of a robot's behaviour.

Calibrating model parameters for a robot can lead to optimal results in protected environments but can result in highly brittle robotic behaviour [15, 16]. This typically incites task-specific robots which are susceptible to environmental changes or alterations in the dynamics. From a practical perspective, having a large number of robots which have been individually calibrated to perform different tasks does not scale well and can result in a prohibitive capital overhead for many industrial settings. Moreover, for safety and performance concerns, such robots must account for the event of experiencing unfamiliar occurrences which may result in catastrophic behaviour [16].

In the traditional approach, a robot is typically equipped with extereoceptive sensory data (visual feed, radar readings, infrared scans, contact forces, *etc.*), as well as proprioceptive data: information concerning the position of the robot in space, commonly retrieved from torque sensors and rotary encoders in the leg joints which count the frequency at which the joint angle varies. Through object detection methods and other computer vision techniques such as depth estimation [17], the robot is capable of locating itself (pose estimation) and finding the target. An example of pose estimation is illustrated in Figure 1.2.

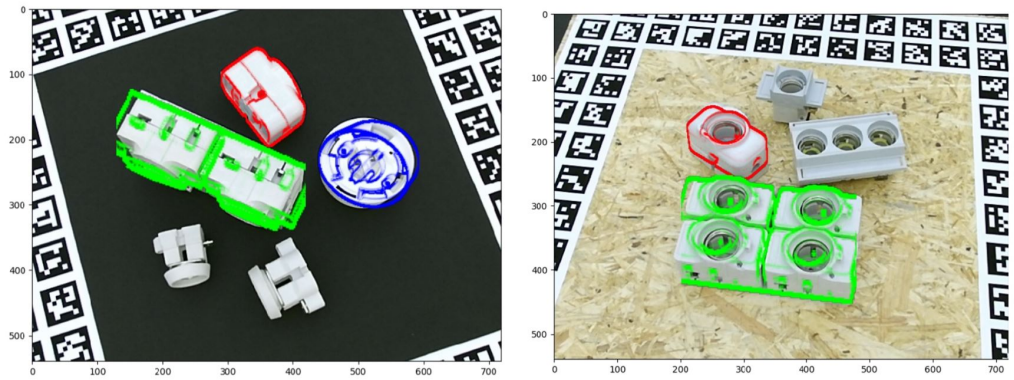

Figure 1.2: 6D pose estimation and object detection using visual cues. Extracted from [18].

From the robot's pose and the target's pose estimation, path planning takes place using previously programmed cues [19]. On top of this, intricate hand-crafted algorithms such as object collision detection and visual-extended Kalman filters are commonly used to improve pose estimation and safety measures [20]. An impedence controller consequently uses the current pose of the robot and the path planning algorithm to drive the robot to its desired way-point. Controller techniques such as positional-integral-

derivative error control (PID) are amongst other conventional ways to achieve smooth behaviours during execution.

Bar to highlighting the amount of engineering work required to set up a robot in this manner, three main sources of errors which severely hinder these methods can be distinguished. First, accomplishing complex manipulation tasks in the real world deteriorates the quality of the models used to define the environment used during path planning. Second, irregularities between "same" objects are natural in the real world and accommodating for randomness requires a higher level of perception and reasoning about the environment. Last, control systems rely on an accurate estimate of the robot's pose relative to the target. Unfortunately, the latter suffers from propagated uncertainty in measurement (joint angles, leg lengths etc.) and most importantly, noise (typically Gaussian distributed) in sensor readings.

1.1.2 Robot Learning

Although many efforts have been employed to mitigate modelling uncertainties, [21, 12], we need robots which are capable understanding their environment at a higher level of abstraction. In the last decade, research directions and exhaustive experimental validation, have indicated the suitability of machine learning (ML), namely reinforcement learning (RL) in robot control [22, 23, 24]. In contrast to the traditional approach, RL uses no prior knowledge of the environment. In order to learn how to solve a task, the learning algorithm explores the task domain and is rewarded when completing the task (sparse reward setting) or when taking actions which lead to states which are closer to solving the goal (dense reward setting). This leads to systems which can automatically learn policies (controllers) which are capable solving a task by inherently creating models of the environment. More details on the methods involved in RL will be presented in section 2.1.

This evidence has supported the abrupt shift in robotic research towards the search for more intelligent robots which learn how to solve tasks autonomously. By understanding the components which govern the definition of the task, ML methods can generalise to a greater extent, where robust changes in the environment could otherwise impede robot performance. In addition, learning from rewards in RL is proxy for a direct stimulus, which attempts to decouple the complexities behind the task to be solved. Particularly, multifarious manipulation problems involving contact-rich sub-tasks, such as solving a Rubix cube with a single robotic hand (Figure 1.3, [25]), or tool-use [23] have been accomplished via RL strategies. The latter tasks, amongst others, were otherwise deemed to be impossible with traditional controllers and have contributed to the aforementioned evidence.

It is certainly more practical to service fewer robots with sufficient perceptive tools and a higher level of intelligence such that these can adapt to different scenarios and thus operate in the real world without the need for constant supervision. As an example, domestic robots must be capable of performing different tasks in an infinite range of environments. Consider how many combinations of colour, textures, material properties, and geometries are present in what we may consider to be a simple task of cleaning the dishes, or cutting an assortment of vegetables. Engineered solutions could indeed succeed in the seemingly trivial

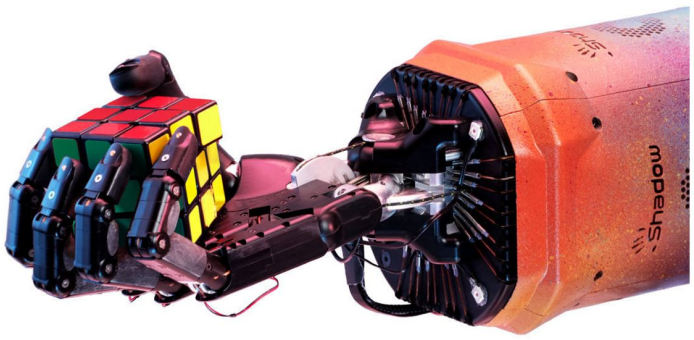

Figure 1.3: Solving a Rubix cube using model-based reinforcement learning. Extracted from [25].

sub-tasks of object approximation through pose estimation but would eventually struggle to accommodate for the difficulties present during contact-rich tasks [26].

In contrast, RL strategies are capable of building accurate and sufficiently general models of the environment which can be used internally to automatically learn which actions to take (model-free) or plan using a model of the robot dynamics and the environment (model-based). In practice we would require generalisation on the performance of a single or even a group of tasks such that the robot can automatically learn how to, for example, cut different sized vegetables or clean different utensils.

Although the representational power behind deep learning methods is vast, RL suffers from low sample efficiency [27]. That is, in order for the robot to converge to the optimal learning behaviour, otherwise referred to as the policy, the robot must theoretically cover all possible transition tuples infinitely many times [27]. Parametric function approximators such as Deep Neural Networks (DNNs) help mitigate this problem by interpolating between unseen state-action pairs, however these typically suffer from overestimation and catastrophic forget [28]. Moreover, mixed supervised learning through imitation learning may also help stabilise the policy by providing a task prior through human demonstrations [29]. Imitation learning not only reduces sample complexity by stabilising the policy but it also results in more natural behaviours [30].

Notwithstanding, stabilising deep neural networks for RL may require prohibitive amounts of training time, especially when working with rich sensory inputs such as images. This constraint usually induces the majority of learning to elapse during simulation engines, creating a reality gap between the learnt policy and the desired policy which results in distributional shift. To close the so-called reality gap, robots trained in simulation are usually fine-tuned before deployment via adaptation techniques such as domain randomisation [31] or Sim2Real [32]. These methods are based on the principle of introducing sufficient variability in the simulation training environment such that during deployment in the real world, the distributional shift in observations and the policy is minimised.

A heavily researched topic in RL relates to the exploration-exploitation trade-off: how to balance between exploring new states which might lead to higher rewards, and exploiting the state-action pairs which seemingly lead to high rewards. One of the problems is that exploration strategies are conventionally task-

agnostic [33, 34, 35], and thus might not exploit robot's understanding of the task. These exploration strategies require hours if not days of tedious hyper-parameter calibration - parameters such as noise which govern untrained exploration strategy variables - and result in sub-optimal highly task-specific learning algorithms.

Realistically, if we aim to develop practical robots which may continuously adapt to the environment and learn on the go, we must seek for systems which are flexible to different conditions rather than solely acting optimally in a specific setup. As humans, we are rarely confronted with the same exact situation: a bottle of water never looks or feels the same, or even holding and using different pens involves different control. Nonetheless, we are capable of understanding how to solve an unseen task given our past experiences by inferring from those tasks which are relatable.

1.2 Project Scope

Robot learning presents an interesting approach where, not only solutions to intricate problems can be found by automatically developing complex models of the environment, but generality can be formulated through inference and reasoning about a task. In particular, this is most interesting in situations where the traditional approach is prone to failure. For instance, tasks where online adaptation to changes in the the environment is required. Particularly, we find building robot learning tools capable of producing different behaviours to unknown environments paramount to the development of intelligent and generalisable machines.

In essence, we aim for machines capable of using prior knowledge from previously solved tasks (either in simulation or in the real world) and using its interpretation of the the occurrences it has processed in real time to adapt its behaviour accordingly [36]. For example, if a robot were to perform the task of chopping an onion, it could exploit this knowledge to learn how to chop a carrot too. Parallel to inciting continuous learning, this would attempt to replicate long-term memory functions in the prefrontal cortex [37], thus taking a step further into general artificial intelligence. In line to this train of thoughts, meta-reinforcement learning strategies were devised with the goal of "learning to learn". Through inference from prior task experiences, the system learns how to distinguish the type of task being solved and aims to quickly adapt to it [38].

Meta-learning in RL provides a framework to learn how to parametrise an arbitrary function, say an exploration strategy. By interacting with the environment and storing the actions, states and rewards covered during exploration, the robot can adapt its behaviour by linking the collected experiences with a set of previously solved tasks. Indeed, dynamically adapting exploration is a possible application for this framework and has already showed promising results [39].

Dense reward functions are most common in the adaptation of said meta-learning algorithms as they provide enriched information about the specific task. However, recovering dense rewards in the real world requires a full understanding of the environment. Other than being overly intricate, if not impossible, this defeats the purpose of why robot learning exists in the first place. Although robot learning through

traditional reinforcement learning could still be used to solve an otherwise complex task like opening a door, if we had to provide the robot with information about the exact position of the door handle, then this scenario would be far from generalisable in the real world. Moreover, we could benefit from robots which attempt to adapt to the real world by providing them with tools to understand that everything it experiences is ambiguous by definition, as is the case in sensor readings [40]. In this case, robots would be capable of adapting to situations by capturing the overall representation which is inferred during adaptation rather than being overconfident of what the task at hand might be, thus imparting further flexibility.

In contrast to dense reward functions, demonstrations present a natural way of explaining what an unseen task might consist of: as humans we continuously learn from demonstrations. For instance, one can learn what a task such as striking a tennis ball from observing an expert perform said task. Nevertheless, an inherent problem of learning from demonstrations, or imitation learning, is that hundreds of independent demonstrations may be required to produce a system which is not too brittle to unexpected observations in the environment or alterations in the initial state of the robot [29]. However, via meta-learning, the robot can potentially learn how to understand the statistical similarities from demonstrations of different yet similar tasks (recall the analogy of chopping onions and chopping carrots), relaxing the constraint of requiring such prohibitive amount of demonstrations. Additionally, using expert demonstrations to condition policies during reinforcement learning training can result in significantly improved exploration, and thus, increased sample efficiencies and asymptotic returns.

In this project we will attempt to close the gap of of hybrid meta-reinforcement and imitation learning by conditioning exploration policies on expert demonstrations. As a result, we aim to devise a system which, when given a set of k demonstrations of an unseen task, has the ability of "learning to learn", *i.e.* adapting its policy such that it can eventually learn how to solve said task (Figure 1.4). This method promises a natural and compatible way of teaching a robot to complete a wide range of tasks in the real world by leveraging prior experiences.

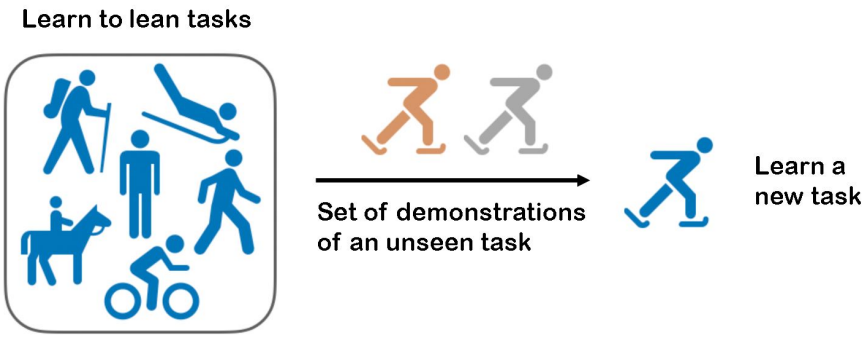

Figure 1.4: Learning how to learn by exploiting knowledge on how to solve a set of previously seen tasks, and adapting to an unseen task via a set of demonstrations. Adapted from [41].

1.2.1 Aims & Objectives

We aim to employ fast adaptation at test-time by exploring methods drawn from meta-reinforcement learning, namely those specific to meta-learning exploration strategies and long-term inference techniques. Additionally, we aim to incorporate learning from demonstration into the meta-learning loop, a novel technique which could potentially stabilise the meta-adaptation process further and generalise to complex tasks by reducing the need to incorporate dense hand-crafted rewards. Parallel to this, we aim to validate our proposed method by demonstrating its generalisation capabilities to multi-task meta-learning settings. With this, we push meta-learning methods a step further with the intention of providing a practical and robust algorithm for the real world.

It is key to distinguish the differences between learning from unseen tasks under the same task family, and learning tasks with inherently different dynamics. Our first primary objective for this project is to exploit meta-learning abilities in problems where the robot does not have any information about the conditions of the unseen task, and every task belongs to a single task family. Visualise the task where one attempts to screw a bolt in a threaded hole without visual information. You have screwed 100 bolts during training, where each bolt was geometrically different and each hole was at a different location. Notice we would describe the task family as "screwing a bolt". At test time you are presented with an unseen bolt and must screw it in an unseen location. The exploration strategy one would employ would be based on (i) Prior experiences one may recall, involving insertion tasks and searching; (ii) Exploratory interactions and demonstrations on the unseen task.

We hope that our method showcases that leveraging demonstrations is crucial in order to ensure that the exploration strategy is heuristically conditioned. Note that adapting in sparse reward conditions gives little information about the actions which lead to high rewards. For that matter, rather than solely basing the agent's task belief on sparse interactions, by meta-learning how to interpret demonstrations, adaptation performances are expected to be improved.

Our second primary objective is to devise a method which is capable of performing multi-task and out-of-task meta-learning. In this setup, we have a robot that learns how to perform tasks under diverse task families, such that it can effectively "learn how to learn". By doing so, we seek for a system that can adapt to (i) Any task from within the trained task families; (ii) Tasks from different task families. Ultimately we wish to verify that our approach allows reasoning about which task it is adapting to, and with this, bring state-of-the-art meta-RL one step further into general methods for artificial intelligence.

1.3 Contributions

The primary contribution to this project is an off-policy meta-learning algorithm for hybrid imitation and reinforcement learning (PERIL). This method achieves exceptional adaptation rates by adapting to unseen tasks with a single demonstration. Moreover, our method is capable of improving upon its performance through structured exploration.

In this project we provide with mathematical grounds for the first hybrid meta-reinforcement and imitation learning algorithm. To the best of our knowledge, this is the first algorithm to achieve zero-shot learning in complex multi-task meta-learning settings under sparse reward feedback. Furthermore, we verify the robustness of PERIL by using imperfect demonstrations, where the robot meta-learns to infer the key components which govern a demonstration. In addition, we present stabilisers for probabilistic meta-reinforcement learning and showcase their effectiveness via improved sample efficiency and convergence rates. Parallel to this we improve upon previous state-of-the-art inference models via attention-based Graph Neural Networks.

A secondary contribution to this project is a library for simple robot learning research. Designed as a tool-box for algorithmic investigation, this library allows for robot learning methods such as imitation learning and reinforcement learning. We also incorporate a set of Gym-compatible (OpenAI [42]) environments which consist of 7 2D robot-like contact-rich tasks. Last, we provide a set of 4 benchmarks for robot learning research.

1.4 Report Structure

The report begins with an introduction to the fields of reinforcement learning, meta-learning, and imitation learning, followed by a brief summary on the fundamentals of robotics. Sequentially, we discuss related works in the areas of complex manipulation, meta-reinforcement learning and imitation learning, with emphasis on the application of robotics. A thorough description of the mathematical background behind PERIL and the practical implementations carried out to execute said algorithm is presented in chapters 4 and 5. We evaluate and discuss our method's performance over different benchmarks and an ablation study is realised to break down the importance of each of the components behind PERIL, from which we draw our conclusions and suggest future works on chapter 7.

Chapter 2: Background

2.1 Reinforcement Learning

Reinforcement learning tasks are typically posed as Markov Decision Problems (MDPs). An MDP is defined as a tuple $\langle s, a, r, s', \gamma \rangle$ where $s \in \mathcal{S}$ corresponds to the current state of the agent in the environment, $a \in \mathcal{A}$ is the action taken at s, $r \in \mathcal{R}$ is the reward obtained by arriving to the next state $s' \in \mathcal{S}$ and γ is the discount factor, which serves as a reward decaying factor for future rewards and determines the priority of short-term rewards. A crucial property of MDPs is that the future only depends on the current state (2.1). This allows closed-form solutions to be devised as the current state contains all the statistical information required to decide the future.

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_t, ..., S_1] \tag{2.1}$$

It should be clarified that for most tasks, a state is not necessarily defined by the pose of a robot or a robot's observations through a camera feed. Due to the Markov property, a state must contain all the information required to predict the future. If this is not the case, the MDP is said to be partially observable (POMDP), and the process becomes significantly more complex. Although working with POMDPs accompanies additional difficulties, some POMDPs can be translated back into MDPs by including inference mechanisms which utilise previous incomplete states to output a proxy for a the full state s_t at time t [43].

In the RL framework (Figure 2.1), an agent acts in an environment $\delta(\cdot)$. At a discrete time step t the agent computes an action a_t drawn from a policy $\pi(a_t|s_t)$. Noteworthy is that the policy can be stochastic: different actions could be drawn from the same state by sampling from a distribution. On the other hand, policies can be deterministic $\pi(s_t) = a_t$. The next state s_{t+1} at which the agent will arrive to after performing action a_t depends on the transition probability $P(s_{t+1}|s_t, a_t)$ which is referred to as the model of the environment. At arrival to state s_{t+1}, the agent receives a reward r_{t+1} from the environment and the process repeats for a series T steps which define the *horizon* of the MDP.

The aim of the paradigm of RL is to maximise the return G_t, defined as the discounted sum of rewards (2.2) from time step t to a finite horizon T. Note that RL is conventionally formulated in the finite horizon setting: once an agent performs T steps in an environment or the agent reaches a terminal state (such as completing the task), we call this an *episode*. After each episode, the agent is re-initialised

under its initial state distribution $s_0 \sim p_0(s_0)$.

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{2.2}$$

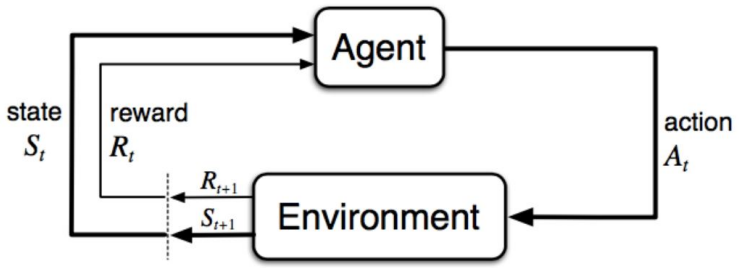

Figure 2.1: Reinforcement Learning framework illustration. Adapted from [27].

More specifically, in RL the ultimate goal is to come up with an optimal policy $\pi_*(a_t|s_t)$ which maximises the expected sum of rewards from any given state. In RL, we refer to this term as the value function. Value functions give a measure of how good a given state or state-action pair might be in the long run. State-value functions $V_\pi(s_t)$ (2.3) determine the suitabily of being at state s_t whilst action-value functions (Q-functions) $Q_\pi(s_t, a_t)$ (2.4) evaluate how promising taking an action a_t from state s_t is.

$$V_\pi(s_t) = \mathbb{E}_\pi[R_t|S_t = s_t] \tag{2.3}$$

$$Q_\pi(s_t) = \mathbb{E}_\pi[R_t|S_t = s_t, A_t = a_t] \tag{2.4}$$

Intuitively, the optimal policy is that which gives rise to the optimal value functions (2.5, 2.6).

$$V_*(s_t) = \max_\pi V_\pi(s_t) \tag{2.5}$$

$$\pi_* = arg\max_\pi Q_\pi(s_t, a_t) \tag{2.6}$$

Through dynamic programming, value functions can be decomposed to isolate the environment dynamics $p(s_{t+1}|s_t, a_t)$ (2.7).

$$\begin{aligned} Q_\pi(s_t) &= \mathbb{E}_\pi[R_t|S_t = s_t, A_t = a_t] \\ &= \sum_a \pi(a|s) \sum_{s\prime} \sum_r p(s\prime|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s\prime]] \\ &= \sum_{s\prime,r} p(s\prime|s, a)[r + \gamma \sum_a \pi(a|s) Q_\pi(s\prime, a\prime)] \end{aligned} \tag{2.7}$$

In practice the environment dynamics, or the *model* of the environment may or may not be known and these settings respectively correspond to two commonly known branches in RL, namely model-based and model-free RL. During model-based RL, $p(s_{t+1}|s_t, a_t)$ is learnt through supervised learning by sampling transition tuples from the environment.

Model-based methods allow for planning processes to occur iteratively through processes such as iterative least quadratic regulator (iLQR) and MPC [44]. MPC accounts for error accumulation in modelling by re-planning the next set of actions every k steps, whilst iLQR optimises planning via quadratic optimisation. Although model-based algorithms are more sample efficient than model-free methods, that is, require less transitions in the environment to achieve high rewards, the representational power of these methods is prone to be less than that in model-free methods. The sole reason for this is that in model-based learning, the transition probability is decoupled from the Q function and the policy π, reducing the representational capacity [45]. For that matter, Dyna-style algorithms which combine model-free and model-based methods (Dyna-style algorithms) are commonly implemented when seeking to exploit a model.

In the model-free setting we directly optimise the Q function. There are two common methods for optimising the policy. The first method named Q-learning was formulated by Watkins & Dayan in 1992 [46]. In essence it involves using temporal difference (TD) approximations to compute a target value for the value function, and regress on this value using an arbitrary optimisation algorithm (2.8). Here α is a hyper-parameter corresponding to the learning rate. The main idea behind TD is to update the value function with an estimated return $r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$, otherwise referred to as the TD target. The optimal policy subsequently takes actions which lead to maximal Q-value (2.6). In practice parametrising Q-functions in Q-learning with statistical learning tools such as deep neural networks allows Q-learning via bootstrapping on the TD target signal [43].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.8}$$

The second family of model-free methods is based on policy gradient theorem [27].

2.1.1 Policy Gradients

In the policy gradient formulation we try to optimise a policy $\pi_\theta(s|a)$ conditioned by parameters θ with the aim of finding an optimal behaviour strategy for the agent to obtain optimal rewards. In that sense, we define the conditioned reward function $J(\theta))$ as:

$$J(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a) \tag{2.9}$$

where $d^\pi(s)$ is the stationary distribution of the Markov chain for π_θ. In other words, it is the on-policy state distribution which accounts for the probability of being at state s after rolling out policy π for an infinite number of steps. Gradients of $J(\theta)$ can be computed via the following decomposition:

$$\begin{aligned} \nabla_\theta J(\theta) &\propto \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) \\ &= \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \\ &= \mathbb{E}_\pi[Q^\pi(s, a) \nabla_\theta \ln \pi_\theta(a|s)] \end{aligned} \tag{2.10}$$

Notice that the policy gradients defined in (2.10) are based on (i) On-policy rollouts, where states and actions are sampled from the current policy $a \sim \pi, (r, s) \sim \delta(\cdot)$; (ii) Having an independent Q-value function $Q(s, a)$. Sampling on-policy results in large sample inefficiences because every time you evaluate policy gradients, you must re-sample from the environment. This is particularly impractical in applications such as robotics where collecting data is slow. Importance sampling methods can be leveraged to resolve this problem by allowing expectancy to be taken from a different distribution, resulting in the following proxy for the policy gradient:

$$\nabla_\theta J(\theta) \propto \mathbb{E}_\phi\Big[\frac{\pi_\theta(a|s)}{\phi(a|s)} Q^\pi(s, a) \nabla_\theta \ln \pi_\theta(a|s)\Big] \tag{2.11}$$

where ϕ corresponds to the policy which was used to collect the data $a \sim \phi, (r, s) \sim \delta(\cdot)$. Moreover, we can leverage actor-critic methods to simultaneously learn policies $\pi_\theta(a|s)$ and value functions $Q_\omega(s, a)$, $V_\omega(s)$ to assist the policy update:

- The critic updates the value function parameters ω with update rules as in (2.8).
- The actor updates policy parameters θ for $\pi_\theta(a|s)$ in the direction suggested by the critic.

A key paradigm on RL methods is the exploration-exploitation tradeoff: we might need to explore regions of low rewards if we want to achieve high rewards in the future, rather than acting deterministically with the information we have. In light of this, entropy-regulated rewards (2.12) aim at inciting a stronger coverage of the state space, where α is the temperature and it controls the weight of the state-conditioned entropy $\mathcal{H}(\pi(\cdot|s))$.

$$J(\theta) = \sum_{s \in \mathcal{S}} \mathbb{E}_\pi[r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))] \tag{2.12}$$

Entropy-regulated rewards are especially effective in settings where the reward function gives insufficient information about the task. A clear example of this is sparse rewards settings. Here, agents are only rewarded positively when the agent successfully completes the task.

2.2 Deep Learning

2.2.1 Deep Neural Networks

An Artificial Neural Network (ANN/NN) can be represented as a directed acyclical graph, where the input of the graph receives a set of vectors and is processed by one or more hidden layers before computing the output. Although biologically inspired by the activation in the brain, ANNs are conventionally exploited as statistical models [47]. The simplest NN is the Multilayer Perceptron (MLP). It can be proven that an MLP with one hidden layer and an infinite number of neurons can approximate any continuous function and is therefore a valid universal function approximator [48].

The mathematical operations in a single chain of an MLP closely resemble that of linear regression, with the exception of a non-linear activation function. In essence, each layer k of an MLP consists of a set of m_k neurons densely connected via weights $W_{k+1} \in \mathbb{R}^{m_{k-1} \times m_k}$ to a set of m_{k+1} neurons on the following

layer (Figure 2.2). As in linear regression, each neuron also contains a bias b_k which is independent of the input from the previous layer. The input to the network is a is a set of n feature vectors $x_0 \in \mathbb{R}^f$ forming a batch input $X_0 \in \mathbb{R}^{n \times f}$. Each layer sequentially computes a linear combination Z_k of the vector product between the weights of that layer and the outputs of the previous layer, with the addition of the biases. Via a non-linear activation function ϕ, the input to layer the layer Z_k is transformed into a non-linear space (2.13). This chained process repeats for as large a number of hidden layers as designed for, hence the name deep neural networks.

$$X_k = \psi(Z_k) = \psi(X_{k-1} W_k + b_k) \tag{2.13}$$

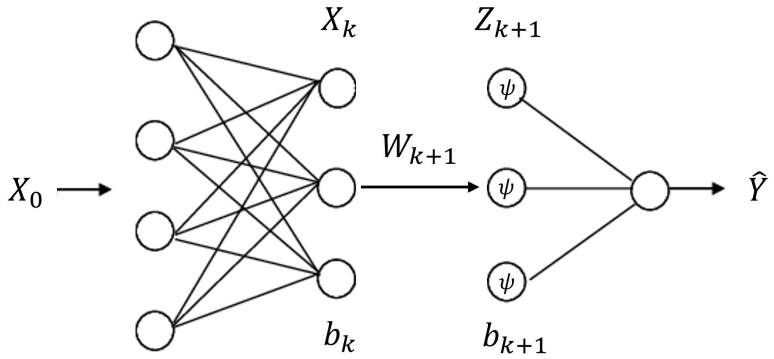

Figure 2.2: Illustration of a Multilayer Perceptron.

Neural networks are built with the aim of statistically learning to reproduce a task by a set of examples, without considering any task-specific rules which may constrain the learning process. Effectively, the primary objective of a NN model $\mathcal{M}_\theta$ is to tweak the weights and biases (parametrised by θ) in order to maximise the likelihood of observing the dataset $\mathcal{D}^{train}$ it is trained on: $\max_\theta p_\theta(\mathcal{D}^{train}|\mathcal{M}_\theta)$ with the aim of statistically matching an independent test dataset $\mathcal{D}^{test}$. This optimisation method is performed via gradient descent on an objective commonly known as the *loss function* $\mathcal{L}(\cdot)$. Conventional loss functions include the Mean Squared Error (MSE) between the predicted outputs $\hat{y}$ and the ground truth outputs y^* (2.14) or the Entropy Loss defined by the log likelihood of matching a desired distribution (2.15).

$$\mathcal{L}_\theta(\cdot) = \sum_{i=1}^{N} (\hat{y}_i - y_i^*)^2 \tag{2.14}$$

$$\mathcal{L}_\theta(\cdot) = -\sum_{i=1}^{N} \log p(\hat{y}_i = y_i^* | x_i, \theta) \tag{2.15}$$

Using the chain rule, the Jacobian (gradients) of the loss computed after a *forward pass* (graphical computation from inputs to outputs) can be back-propagated from the output layers back to the hidden layers, distributing the loss gradients to each weight in the network (more on this on [47]). With this, steps of gradient descent (2.16) can be performed to tweak the network with the aim of reducing the loss in the next forward pass. The step size α controls how much to shift the parameters of the model

and is typically heuristically controlled by an optimiser. This process is performed for N training steps, where $0 < t \leq N$ is the training step and $\mathcal{L}_k$ is the back-propagated loss to layer k.

$$\theta_{t+1}^k \leftarrow \theta_t^k - \alpha(\nabla \mathcal{L}_k(\theta_t^k)) \tag{2.16}$$

2.2.2 Variational Encoders

Variational Encoders (VEs) are a form of non-linear feature extraction models which attempt to map an input space $x \in \mathbb{R}^f$ into a latent space $z \in \mathbb{R}^g$ via an encoder $q_\phi(z|x)$. The difference between an encoder and a VE is that, through an Information Bottleneck (IB), VEs assume that the latent space z can be represented by an underlying probability distribution, typically of Gaussian nature. In contrast, Variational Autoencoders (VAEs) are a form of generative models which are capable of generating statistically similar data by learning a mapping between the inputs x and a latent space z via an encoder $q_\phi(z|x)$ such that reconstructing $x\prime$ from z using a decoder $p_\theta(x|z)$ results in the least reconstruction error. In essence, VAEs attempt to perform Principal Component Analys (PCA) conditioned by an IB (Figure 2.3).

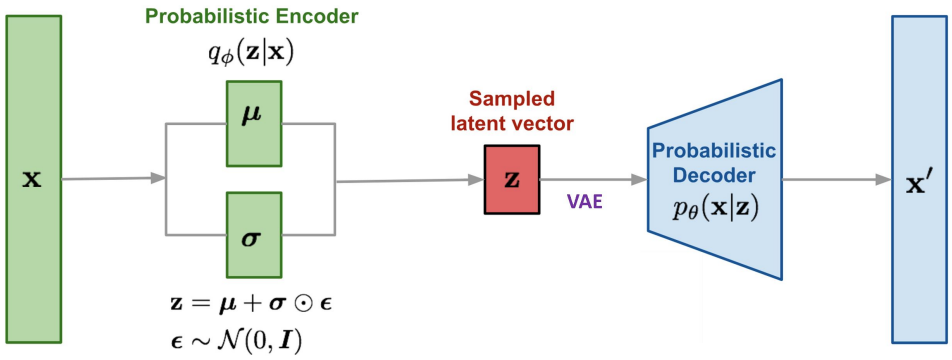

Figure 2.3: Variational Encoder and Variational Autoencoder Models. Variational encoders directly optimise z whilst autoencoders condition z on reconstruction via a decoder.

Both variational approaches approximate the posterior $q_\phi(z|x)$ to the real (unknown) posterior $p(z|x)$. This can be done by defining a Kullback-Liebler Divergence D_{KL} maximisation objective (2.17). Noteworthy is that the KL divergence is a form of distributional-matching measure and is high when the two distributions are statistically dissimilar. A key difference between VAEs and VEs is that VAEs perform reconstruction-based PCA, whilst VEs perform goal-conditioned PCA. From maximum likelihood estimation, we can derive a lower-bound for VAEs (2.18) which is directly related to the optimisation objective of VE by replacing the reconstruction loss term by a goal-conditioned term $\mathcal{G}$ (2.19), where $p(z)$ is an arbitrary distribution, typically the standard Gaussian $p(z) := \mathcal{N}(0, \mathbb{I})$, and $D_{KL}(q_\theta(z|x)||p(z))$ represents the IB term.

$$D_{KL}(q_\phi(z|x)||p(z|x)) = -\int q_\phi(z|x) \log\left(\frac{p(z|x)}{q_\phi(z|x)}\right) dz \geq 0 \tag{2.17}$$

$$\log p(x) \geq -D_{KL}(q_\phi(z|x)||p(z)) + \mathbb{E}_{z\sim q_\psi(z|x)}[\log p_\psi(x|z)] \tag{2.18}$$

$$\log p(x) \geq -D_{KL}(q_\phi(z|x)||p(z)) + \mathbb{E}_{z\sim q_\psi(z|x)}[\mathcal{G}(\cdot|z)] \tag{2.19}$$

We can then optimise the models with the lower bound *i.e.* the likelihood of matching the distribution of the data-set $p(x)$.

2.2.3 Graph Neural Networks

As defined in [49], a graph $\mathcal{G}$ is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices or nodes and $\mathcal{E}$ corresponds to the set of edges. We define $n = |\mathcal{V}|, m = |\mathcal{E}|$ as the number of nodes and edges in $\mathcal{G}$ respectively. For every node $\nu_i \in \mathcal{V}$ we assign an edge $e_{ij} \in \mathcal{E}$ which links ν_i to ν_j. The neighbourhood of a node ν can be defined as the set of nodes which are connected via edges to that node $N(\nu) = \{u \in \mathcal{V} | (\nu, u) \in \mathcal{E}\}$. We define the adjacency matrix A as an $n \times n$ matrix which defines the set of edges which link each node, with $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and 0 otherwise.

A graph may contain nodal attributes $X \in \mathbb{R}^{n \times d}$, where $x_\nu \in \mathbb{R}^d$ represents the feature vector of node ν. A graph can also have edge attributes $X^e \in \mathbb{R}^{m \times c}$, where $x^e_{\nu,u} \in \mathbb{R}^c$ represents the feature vector of an edge (ν, u). Directed graphs have asymmetric adjacency matrices, where all edges are directed from node ν to u. In contrast, undirected graphs have symmetric adjacency matrices, where each directed edge connecting ν to u has a corresponding edge connecting u to ν.

2.3 Meta Learning

The process of meta-learning is commonly understood as *learning to learn* [38]. This is different to traditional ML, where the objective is to train a set of parameters θ from a model $\mathcal{M}_\theta$ to match the distribution of the predictions of said model to that of its training data $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$. Here, $(x_i, y_i), i \in [0, ..., N]$ corresponds to a tuple of input data feature x and output label y, and $\mathcal{L}$ is the loss function we want to minimise (2.20).

$$\theta^* = arg \min_\theta \mathcal{L}(\mathcal{D}; \theta) \tag{2.20}$$

During meta-learning, an *outer loop* algorithm updates an *inner loop* ML algorithm such as deep policy gradients (2.11). The outer loop, or meta-controller, has access to the parameters in the inner loop and it can therefore modify these accordingly with the aim of improving a global *meta objective*. In practice, meta objectives may vary significantly, from generalisation performance, to learning speeds and inference time [50]. In ML research, meta-optimisation is conventionally hand-tuned through processes such as hyper-parameter tuning via cross-validation search. In contrast, meta-learning can become particularly useful when generalising to unseen tasks by tuning the inner loop algorithm along a set of tasks $\mathcal{T}$ from with a task family $p(\mathcal{T})$. By aiming to learn how to learn, meta-learning attempts to produce general purpose algorithms that can generalise across different tasks.

Consider a distribution over tasks $\mathcal{T} \sim p(\mathcal{T})$ where a task is defined as the set $\{\mathcal{D}, \mathcal{L}\}$. We introduce a variable λ which determines the learning conditions *i.e.* how to learn. The objective of meta-learning is to learn how to learn. Thus, the meta-learning is formalised as in (2.21), where $\mathcal{L}(\mathcal{D}; \lambda)$ marks the

performance of the model across tasks under learning condition λ.

$$\min_{\lambda} \mathbb{E}_{\tau \sim p(\mathcal{T})} \mathcal{L}(\mathcal{D}; \lambda) \tag{2.21}$$

Assume we can divide a set $\mathcal{K} \subseteq \mathcal{T}$ of training tasks into meta-train training and meta-train validation data $\mathcal{D}_{train} = \{(\mathcal{D}^{train}_{train}, \mathcal{D}^{eval}_{train})^{(i)}\}_{i=1}^{i \in \mathcal{K}}$. The meta-training step can therefore be formalised as maximising the likelihood of the learning condition λ (2.22).

$$\lambda^* = arg \max_{\lambda} \sum_{i \in \mathcal{K}} log\ p(\lambda | \mathcal{D}^{train}_{train}) \tag{2.22}$$

During adaptation in the testing stage, we consequently assume we can divide a set $\mathcal{E} \subseteq \mathcal{T}, \mathcal{E} \cap \mathcal{K} = \emptyset$ of testing tasks into meta-test training and meta-test training data $\mathcal{D}_{test} = \{(\mathcal{D}^{test}_{train}, \mathcal{D}^{test}_{test})^{(i)}\}_{i=1}^{i \in \mathcal{E}}$. At meta-test time we want to quickly adapt our parameters θ such that, given our meta-learnt knowledge λ, we maximise the likelihood of the test-time training distribution (2.23).

$$\theta^{*(i)} = arg \max_{\theta} \sum_{i \in \mathcal{E}} \log p\big(\theta | \lambda^*, \mathcal{D}^{train\ (i)}_{test}\big) \tag{2.23}$$

2.3.1 Meta-Reinforcement Learning

Meta-Reinforcement Learning, or Meta-RL, was introduced by Wang et al. at DeepMind [21] and Duan et al. at OpenAI [38]. In the basic meta-RL framework we consider a distribution over tasks $\mathcal{T} \sim p(\mathcal{T})$, where each task $\mathcal{T}$ corresponds to a unique MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ with horizon T, environment $\delta(\cdot)$, discount factor γ, state space $\mathcal{S}$, action space $\mathcal{A}$, model transition dynamics P and reward function R. Conventionally, it is the latter three which vary across tasks, whilst $\mathcal{A}, \mathcal{S}$ are fixed.

In essence, Meta-RL tries to learn a policy π_θ that can solve all training tasks. With this, the goal is to adapt to novel tasks, given training priors (behavioural policies conditioned on meta-training tasks) along a task distribution $p(\mathcal{T})$. The meta-objective for meta-RL can therefore be formulated as maximising the expected sum of rewards along the meta-training task distribution (2.24).

$$\theta^* = arg \max_{\theta} \sum_{\mathcal{T} \in p(\mathcal{T})} \mathbb{E}_{\pi_{\theta^*}} \big[\mathbb{E}_{s_t \sim \delta(\cdot), a_t \sim \pi_\theta(a_t | s_t)} \sum_{t \in T} \gamma^{t-1} R(s_t, a_t) \big] \tag{2.24}$$

Selecting the tasks for meta-training must be done in a thoughtful manner in order to avoid distributional shift during both meta-training and meta-testing. Indeed, incorporating diverse tasks allows for better generalisation as the distribution range during training is effectively augmented.

Notwithstanding, although techniques such as distillation have alleviated slight distributional shifts in meta-RL [41], test-time adaptation must occur along samples within the same distribution as in training. This problem remains an inevitable weak spot in generalisation across ML methods due to the fact that extrapolating behaviours results in policies which act with low confidence.

A different way to pose the problem of Meta-RL is the probabilistic approach, a setup which has shown major flexibility to other deterministic methods [51]. Probabilistic meta-RL can be viewed as a special case of a RL, where each task $\mathcal{T}$ is posed as a POMDP $\mathcal{M}^{PO}$ defined by the tuple $\langle \mathcal{S}, \mathcal{A}, , \Omega, \mathcal{O}, P, R, \gamma \rangle$. Due to partial observability, the agent cannot retrieve full states directly from the environment and must learn from observations. Note the inclusion of observation space $o \in \Omega$ and observation transition probability $\mathcal{O}(o_{t+1}|s_{t+1}, a)$ (probability of observing o_{t+1} and transitioning to state s_{t+1} after taking action a).

The goal is to meta-learn how to extract information from $\mathcal{T}$ in order to provide the observation space $o \in \Omega$ with a task belief $z \in \mathcal{Z}$ which can provide sufficient information to map the observational space into the state space, hence closing the POMDP to its stable MDP form (2.25).

$$\mathcal{M} \leftarrow \{\mathcal{M}^{PO}, \mathcal{Z} \cap \Omega\} \tag{2.25}$$

We can define the RL framework with a transition distribution, $P(o_{t+1}, z_t|o_t, z_t, a_t) = P^z(o_{t+1}|o_t, a_t)$, conditioned reward function $R(r_t|o_t, z_t, a_t) = R^z(r_t|o_t, a_t)$, and initial state distribution $p_0(o_t, z_t) = p(z)p_0(o_t|z_t)$. Each task $\mathcal{T} \in p(\mathcal{T})$ is specifically conditioned with its sampled task belief $z \sim p(z)$. With this, we define a meta-objective (2.26) which we can use to train a belief-conditioned policy $\pi_\theta(a|o, z)$.

$$\theta^* = arg\max_\theta \mathbb{E}_{b\sim p(b), \tau_{0:T}\sim\pi_\theta(a|o,z)}\Big[\sum_t^T \gamma^{t-1} R^b(r|o, a)\Big] \tag{2.26}$$

Notice we use the term $\tau_{0:T}$ to define a trajectory $(o_{0:T}, a_{0:T}, r_{0:T})$. It can be demonstrated that after several steps of manipulation, the posterior $p(z|\tau_{0:t})$ becomes independent of the policy which generated the trajectory (2.27), meaning that this method can be used for off-policy meta-training [52].

$$p(z|\tau_{0:t}) \propto p(z)p_0(o|z)\prod_{t=0}^{t-1} P^b(o_{t+1}|o_t, z)R^b(r_t|o_t, a_t, z) \tag{2.27}$$

This framework can be considered to be a meta-learnt variant to posterior sampling in RL [53], where updates of a set of MDP distributions are performed in order to find the optimal MDP. In contrast, probabilistic meta-RL uses inference to simplify the process of posterior sampling.

2.4 Imitation Learning

In the traditional RL setting, the goal is to learn policies $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to select actions from a given state such that the expected sum of rewards is maximised (formalised in section 2.1). In pure Imitation Learning (IL), agents do not receive a reward signal explicitly from the environment but instead have access to a set of expert demonstrations trajectories $\tau_{demo} = \{(s_t, a_t)\}_{t=0}^{T_{demo}}$ of the task from which they must seek to find a policy π that is capable of producing similar behaviours.

2.4.1 Behavioural Cloning

Behavioural Cloning (BC) is an imitation learning method which uses supervised learning on the states and actions retrieved from τ_{demo} to learn policies π_θ with parameters θ that produce behaviours (state-action

tuples) which are similar to those from the demonstrations (2.28).

$$\theta^* = arg\min_{\theta} \sum_{i \in \tau_{demo}} ||(s,a) - (s_i,a_i)||_2^2, \quad \text{where } a \sim \pi_\theta(a|s) \tag{2.28}$$

They idea of BC is to try to copy the expert's behaviour. Although this reduces the complexity of policy learning processes, BC can result in brittle policies due to distributional shift [54]: the fact that training is performed in sequential data produces non independent identically distributed samples which can diverge from small errors in the state or slight changes in the environment. In the meta-IL case, each task $\mathcal{T}$ has a set of demonstrations trajectories $\tau^d \in \mathcal{T}$. The meta-objective can be defined as trying to maximise the probability of observing the demonstrated trajectories under all tasks $\mathcal{T} \in p(\mathcal{T})$ (2.29).

$$\theta^* = arg\max_{\theta} \sum_{\mathcal{T} \in p(\mathcal{T})} \sum_{\tau^d \in \mathcal{T}} \mathbb{E}_{(s_t^d, a_t^d) \sim \tau^d}[\log \pi_\theta(a_t = a_t^d | s_t^d)] \tag{2.29}$$

2.5 Robotics Fundamentals

A robot can be described as a physical machine which is capable of grasping a sense of the its surroundings, typically through sensors, and exerting a force or change in the real world, typically through actuators. In essence, field of robotics is driven by the urge to create or synthesise machines that can replicate human function and capabilities.

A robotic system is generally comprised by one or more robotic arms. Each robotic arm typically consists of multiple semi-rigid links connected via joints which allow relative motion between neighbouring links. Each joint gives an additional degree of freedom (DoF). The degrees of freedom of a system is defined by the number of independent variations it can have in its space. For example, a rigid body in 3D has 6 DoFs in Cartesian space: 3 translations (x, y, z) and one rotation along each perpendicular plane $(\theta_{xy}, \theta_{xz}, \theta_{yz})$, whilst a rigid body in 2D has 3 DoFs: 2 translations (x, y) and 1 rotation (θ_{xy}).

At the end of the multi-linked arm lies the manipulator, or end effector. This component interacts with the surroundings, otherwise referred to as the *environment* to perform a specific *task* such as closing a door or screwing a bolt. In order to position the end effector of the robotic arm, where the a multi-linked robotic arm in 3D space, said arm must contain at least 3 joints to control translation and 3 joints to control orientation of the end effector (Figure 2.4). More often than not, robots may have additional joints to improve free-space navigation [5].

Joints can be *actuated*, which means that they can be controlled by means of a motor, hydraulics, *etc.*. Analogous to human muscles, actuators allow the robot links to move around and exert forces in the environment. The amount of torque (rotational force) that an actuator exerts depends on the *action* that the robotic controller or *policy* decides to take. In essence, the robotic controller can be considered the "brains" of the machine. Through a set of sensors which provide environmental cues (images, proprioception, *etc.*), the *state* of the robotic agent can be described. This state is typically processed

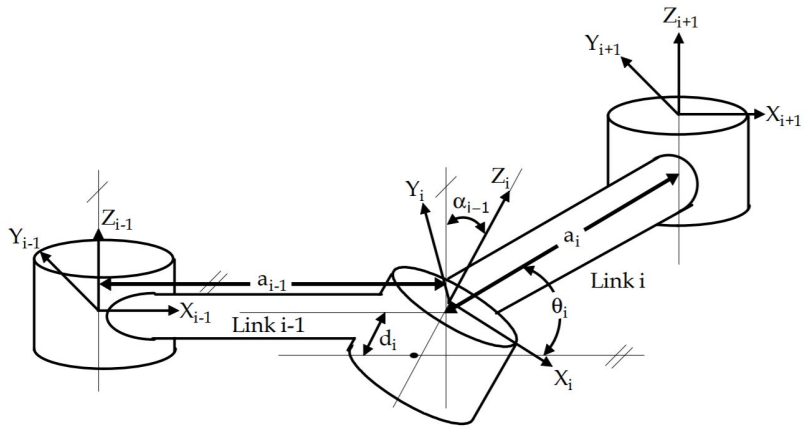

Figure 2.4: Coordinate frame for a multi-armed robotic arm. Each joint has its independent coordinate system, followed by a link of length a_i poisitioned at angles $\theta_i, \alpha_i, \gamma_i$ in the XY, YZ, ZX planes respectively. Adapted from [55].

by the controller in order to perform actions with the aim of performing the designed task.

Tasks are typically defined *a priori* task completion, either by a set of defined instructions or by a set of models (policies) which condition the continuously updated states observed by the robotic agent. Typically, controllers directly command the end effector to navigate to a specific position, or inflict specific forces at a given location.

Commands given to the effector are conventionally given in Cartesian space. Because the arm has multiple joints and actuators, these commands must be mapped to all actuators. Methods derived from Inverse Kinematics (IK) can be used to translate a given position in Cartesian space to desired angles in joint space $J(\theta_i), i \in n$, where n is the number of joints in the system 2.30. In contrast, Forward Kinematics (FK) uses the joint angles to find the position (x, y, z) and orientation $(\theta_x, \theta_y, \theta_z)$ of the manipulator 2.30. Similarly to how back-propagation projects NN outputs towards the parameter space from the input layer, forces and torques at the end effector can also be translated back into the joints.

$$J(\theta_1, ..., \theta_n) \xrightarrow{FK} (x, y, z, \theta_x, \theta_y, \theta_z) \xrightarrow{IK} J(\theta_1, ..., \theta_n) \tag{2.30}$$

Chapter 3: Related Work

3.1 Complex Manipulation

In light of mechanical and hardware advancements for dexterous manipulators, there has been an increasing need to develop algorithms capable of controlling these complex structures in a generalisable manner. Moreover, multi-bodied dexterous manipulation capable of tackling contact-rich tasks such as complex grasping or tool use is crucial to develop robots capable of operating in human-centred environments [23, 22].

On one side of the spectrum, engineered solutions have successfully covered a range of manipulation tasks by proposing task-specific solutions: creating geometric meshes of the environment to plan gating manoeuvres [56]; using contact force mechanics to develop a motion planning algorithm to re-position an object into a stable grasp state [57]. Other hand-crafted solutions to tasks involving re-location, such as peg insertion, include search algorithms such as spiral search or raster search [6]. Although these methods can eventually work in particular settings, they are not general enough to adapt to different tasks and may require models of the environment which can quickly become too complex. As a proxy to this statement, we may consider the works performed in [57], where the authors demonstrate that reasoning about contact model identification during deployment scales exponentially with the number of contacts.

We can quickly draw comparisons between the aforementioned methods and state-of-the-art deep RL methods such as TRPO or SAC [35, 58]. Although manual approaches can scale exponentially, RL methods typically scale linearly with increasing task complexity: task complexity and state-action space dimensions are related in a linear fashion, increasing computational efforts accordingly. Furthermore, deep RL allows increased re-usability: from manipulating Baoding Balls and digit handwriting [23], to opening doors and tool use, the same exact algorithms can be used to learn different tasks [22].

For these reasons, deep RL is quickly superseding hand-crafted solutions and is becoming a strong tool in extracting high-level perception of the environment in an increasingly large number of applications for manipulation in robotics. Via RL, dexterous manipulation has shown extraordinary results in simulation through model-based trajectory optimisation methods such as PPO or TRPO [44, 58]. However, model-based approaches often suffer in conditions where the dynamics are highly complex and the transfer from simulated contact dynamics to the real world becomes too dissimilar. On the other hand, model-free methods provide with a model-agnostic strategy which can circumvent modelling errors.

By capturing the transition probability intrinsically, model-free methods are naturally prone to be equipped with higher representational power than model-based methods. Indeed, building models in discontinuous contact-rich environment can make planning highly erratic. On the other hand, a major caveat in using model-free methods is that these are naturally more sample inefficient [45]. However, recent studies in dexterous manipulation settings have showcased the suitability of model-free algorithms by speeding up training via asynchronous updates [59], meta-learning representations [51], or learning from demonstrations [60].

3.2 Meta-Reinforcement Learning

Meta-RL methods can be divided into three branches: RNN-based, gradient-based, context-based, where RNN stands for Recurrent Neural Network. Initially, meta-RL was conceptualised in the RNN-based family. The latter method was coined as "learning to reinforcement learn", or RL^2, and was developed by Wang et al. [21] and Duan et al. [38]. In their approach, they use a long short-term memory (LSTM) in order to feed a history into the model such that the policy can subsequently internalise the dynamics between the states as well as the actions and rewards visited along the episode.

Ritter et al. [37] expand this idea one step further by tackling catastrophic forget. Inspired by hippocampal memory triggers, their study focuses on augmenting the LSTM with an episodic memory system (epLSTM) that stores the cell state along with a contextual cue. The memory module in epLSTM corresponds to a differential neural dictionary (DND) used in previous works on Neural Episodic Control (NEC) [61]. DNDs store highly rewarding contexts (transition tuples) via key-value pairs experienced during during exploration. These contexts are then used during meta-testing, allowing context embeddings extraction using k-nearest neighbours (KNNs). Despite demonstrating that epLSTM learns state-of-the-art exploration policies by capturing time-variant properties, this framework suffers from unstable learning: RNNs aim to learn the entire meta-learning algorithm at once, thus, stabilising RNN-based policies can become highly intricate and sample inefficient. In order to mitigate this, Humplink et al. [52] suggest an auxiliary loss for meta-RL to stabilise recurrent systems by providing decoupled optimisation goals.

The second line of work adopts a learning-to-learn strategy. Finn et al. developed model agnostic meta learning (MAML) [62], a novel gradient-based meta-learning framework. The latter method meta-learns an initialisation which adapts the parameters of the policy network and fine tunes it during meta-test training. MAML, like other meta-learning algorithms, updates the parameters in an outer loop by minimising the loss along a distribution of tasks. Although promising results simple goal-finding tasks, MAML-based methods fail to produce stochastic exploration policies and therefore adapt to complex tasks [39].

In [39], meta-adaptation through stochastic noise (MAESN) attempts to alleviate this problem by meta-learning a representation of the latent space z which accounts for structured, time-correlated, exploration noise. It does so by conditioning the policy $\pi_\theta(a|s,z)$ on z and meta-training $\pi_\theta(a|s,z)$ to adapt to different parameters of z in order to produce sensible structured exploration strategies at meta-test time.

In their works, MAESN efficiently adapts robust exploration strategies to unseen robotic manipulation tasks such as selective block pushing (learning to push a randomly active block with a robot arm into a target position) or locomotion (navigating a quadruped robot in different speeds and directions).

Meta-learning robust exploration strategies is key in order to improve sample efficiency and allow for fast adaptation at test time. In light of this, context-based RL was developed with the aim of reducing the uncertainty of newly explored tasks. State-of-the-art context meta-RL algorithms such as PEARL [51] and CASTER [63] map transitions $\tau_{0:T}$ collected from an unseen task into a latent space z via an encoder $q_\phi(z|\tau_{0:T})$, such that, similarly to MAESN, the conditioned policy $\pi_\theta(a|s,z)$ can solve said task (Figure 3.1).

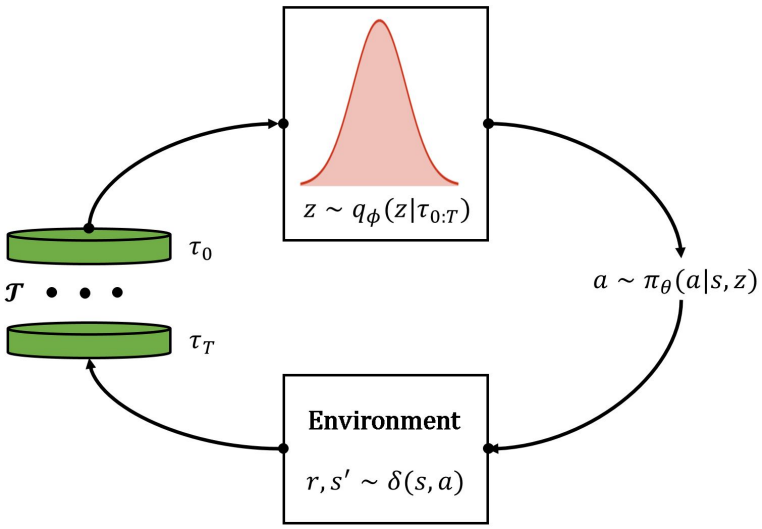

Figure 3.1: Probabilistic meta-reinforcement learning reasoning. From a set of experiences $\tau_{0:T} \sim \mathcal{T}$, an encoder $q_\phi(z|\tau_{0:T})$ produces latent task beliefs z which, by conditioning the policy $\pi_\theta(a|s,z)$, can subsequently solve $\mathcal{T}$.

The benefit of having different parameters for the encoder and the policy networks is that it disentangles task inference from reward maximisation [39]. Whilst gradient-based and RNN-based meta-RL algorithms are on-policy, this decoupling introduces a further advantage of allowing off-policy training, increasing sample efficiency by a factor of 20-100 [51]. Moreover, incorporating an encoder incites fast adaptation as it extracts the principle components of the information which characterise task $\mathcal{T}$.

In practice, context-based meta-RL methods such as PEARL have shown promising results in fast adaptation to unseen contact-rich tasks such as peg insertion [6], as well as in diverse OpenAI-gym environments [51]. Worth mentioning is that in [6], the authors use meta-learning by training on similar insertion tasks during simulation and demonstrate that meta-learning can also be used as a powerful tool to close the Sim2Real reality gap.

A key aspect to consider when proposing solutions for test-time adaptation is how to use both long-term and short-term memory triggers to condition the exploration strategy. In light of this, CASTER [63] attempts to improve PEARL's lack of short-term memory contribution by posing the problem as a

POMDP: PEARL assumes all the information collected during test time is i.i.d. whilst this is not the case since, during exploration at test time, data is collected in a sequential manner. Moreover, in CASTER, the authors give empirical evidence to support their claim that decoupling the exploration policy from the reward-seeking exploitation policy achieves better exploration strategies and faster adaptation. They conjecture that the reason for this is that CASTER optimises for what we want in both exploration and exploitation policies rather than mixing reward maximisation with exploration in the same objective.

An important remark regarding training conditions of the discussed meta-RL methods is that they are typically meta-trained using dense reward functions [51, 21, 63]. The reason for this is that training is usually performed during simulation where the task is perfectly defined and dense rewards such as cosine similarity to the goal state can be computed. In contrast, considering that the ultimate goal is to allow robots to adapt in the real world, adaptation during test time is performed via sparse reward settings [39, 6]. Notwithstanding, this naturally presents the problem of distributional shift as the online policy rollouts are inherently from a different MDP than observed during training via dense rewards.

3.3 Imitation Learning

In IL, expert demonstrations are used in an attempt to incite an agent to act in a similar manner. Demonstrations serve as a reference by either (i) conditioning the policy to produce similar state-action transitions (BC), or (ii) inferring a reward function from which standard RL algorithms can train on to maximise rewards. The latter method is commonly referred to as Inverse-RL (IRL).

In the first case, a well known algorithm for adapting policies through BC is dataset aggregation (DAgger) [64]. In DAgger, the demonstration trajectories are stored in memory and the policy classifier iteratively trains online by taking actions, saving the observed transitions and aggregating the experience with the previous demonstrated transitions. Furthermore, differentiable IL - AggreVaTeD [65] extends DAgger by incorporating continuous action spaces and parametric differentiability. Nevertheless, these methods are only capable of imitation learning are not capable of improving the expert's behaviour [29].

Differently from methods derived from DAgger, which attempt to match the demonstration by training from the its trajectory transitions, GAIL [66], OptionGAN [67] and other IRL-based approaches do not directly interact with the expert [68]. For instance, in GAIL, the policy learns to select actions which bring the policy's occupancy measure towards the expert's trajectories. However, these methods focus on IL and do not accommodate the reward maximisation framework of RL.

In the context of RL, incorporating demonstrations to the training process has proven successful in helping exploration strategies find the goal, stabilising learning and increasing sample efficiency [60]. In recent works, demonstrations have been used to, instead of matching the policy directly to the demonstration as in BC, pre-train DQNs via supervised learning on the TD error [29]. Other approaches as that proposed in DDPGfD involve incorporating IL into deep dynamic policy gradients (DDPG) by initialising the replay buffer with transitions from the expert demonstration.

In the field of dexterous manipulation in robotics, Rajeswaran et al. [23] extend the works in DDPGfG by bootstrapping the policy using BC and consequently using on-policy policy gradients to fine-tune to the task at hand. They propose Demo Augmented Policy Gradients (DAPG) and produce state-of-the-art performance in contact-rich dexterous manipulation tasks involving tool use, object grabbing and relocation, and in-hand manipulation (re-positioning of a stylus). In DAPG, the authors argue that, even though on-policy algorithms are known to be very sample inefficient (prohibitive in robotic applications), given that demonstrations significantly increase sampling efficiency, on-policy methods become plausible as they also benefit from lower variance and thus higher stability than other off-policy algorithms [23].

3.3.1 Meta-Imitation Learning

Learning expressive policies from a defined set of demonstrations typically requires a vast amount of expert trajectories, particularly in high dimensional state-action spaces [23]. However, in practical few-shot settings, there is an identifiable problem that it may not be possible to precisely determine a policy from one or just a few demonstrations, especially in unseen environments. In light of this, meta-IL aims to provide the policy with tools to leverage priors on how to interpret structures amongst demonstrations of different tasks, rather than treating each demonstration as an independent distribution [69].

Meta-IL can be rapidly implemented in inference meta-RL methods such as PEARL or CASTER, by conditioning the agent with demonstration trajectories [69]. Although this sets a good prior for the latent space representation z of a task, it does not condition exploration or even trajectory inference during training, thus the perception of the demonstration is not optimised. Furthermore, this approach does not mitigate the efforts to reduce dense rewarding.

On another hand, Zhou et al. propose a different meta-IL approach named watch-try-learn (WTL) by simply averaging the objective across demonstrations (2.29) from within a single task and optimising a meta-controller. The latter learns to adapt at test-time by receiving binary rewards [69]. The caveats of this approach remain that of traditional IL: (i) Imitating expert trajectories hinders the policy from doing better than the demonstrations; (ii) Cloning behaviours reduces flexibility and generalisation capacity.

Due to the inherent problem of reward engineering in RL training and especially meta-RL adaptation in real tasks, methods such as meta-IRL [70, 71] aim to recover a task-conditioned reward function $r(\cdot|z)$ using structural similarities amongst different tasks. With this information, the policy aims to adapt its task belief within the inferred MDP. Although suitable for real tasks, these methods result in extremely slow and unstable training (on-polcy RL with dual generative processes on (i) reward functions and (ii) latent space task descriptors z) as well as slow test-time adaptation in complex domains [70].

Chapter 4: Methods

In this chapter we present our proposed method, Probabilistic Embeddings for hybrid meta-Reinforcement and Imitation Learning (PERIL). PERIL is designed with the objective of meta-learning how to interpret demonstrations from different tasks in order to adequately condition structured exploration strategies derived from maximum-entropy RL. Through mutual information constraints, PERIL aims to minimise distributional shift from demonstrations presented during meta-test adaptation. Conditioning exploration policies on both expert demonstrations and recently collected experiences aims to provide fast online adaptation to unseen tasks.

We provide an overview of the background and the general objectives encompassed in our approach section 4.1 and incrementally dive into the specific modules which altogether define PERIL in the following. In section 4.2.1 we develop the concept of task inference for meta-RL further and propose an inference strategy to recover latent space embeddings. Subsequently, based on state-of-the-art methods for meta-RL (PEARL [51]) we expand the notion of probabilistic meta-RL in section 4.2.2 and build from that by including decoupled auxiliary objectives. Finally, in section 4.2.3 we disclose our proposal for linking off-policy meta-RL with off-policy meta-IL by introducing a constrained objective on mutual information between demonstration trajectories and latent task embeddings.

4.1 Problem Statement

Adaptability is a desirable attribute in deployable systems such as robots, which are continuously encountered with new tasks. With this as our main objective, we base the foundations of our approach on the ultimate goal of leveraging past experiences to solve new tasks and exploiting demonstrations to improve adaptation.

We assume access to set of tasks $\mathcal{T} \in p(\mathcal{T})$, where each task is a POMDP construed as the tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{O}, P, R, \gamma, T \rangle$ defined as in section 2.3.1. Each task $\mathcal{T} = \{p(o_0|z), p(o\prime, z\prime|o, a, z), r(o, a|z)\}$ has an independent initial state distribution $p(o_0|z)$, transition distribution $p(o\prime, z\prime|o, a, z)$ and reward function $r(o, a|z)$. By leveraging task beliefs, PERIL exploits belief-enriched observational space ($s \sim \{o \cap z\}$) with the objective of closing said POMDP into a stable MDP.

By definition, the task distribution $p(\mathcal{T})$ can inherit tasks with diverse dynamics and reward functions. From a theoretical point of view, this brings coverage to a whole spectrum of robotic-based task families which may be meta-trained on. For example, different dynamics may involve task alterations such as

changes in the angle and position of insertion of an object inside a clearance hole of different geometries (change in environment $\delta(\cdot)$). In contrast, altering reward functions maintains the dynamics of the task, but may bring noticeable behavioural changes such as navigating a robot to different positions, or unscrewing instead of screwing a bolt.

With the aim of supporting continuous task inference, we condition probabilistic embeddings z on a set of recently collected transitions, referred to as the context c. In particular, we are trying to find the true context-conditioned posterior $p(z|c)$ over the task belief $z \sim p(z|c)$ conditioned on c. We define a contexton $c_t^{\mathcal{T}} = (o_t, a_t, r_t, o_{t+1})$ as a transition collected on task $\mathcal{T}$ at time-step t such that the context $c_{0:t}^{\mathcal{T}}$ (what we refer to as c) denotes the set of accumulated contextons. This set contains contextons sampled from $t = 0$, where the agent is initialised along the task-dependent observation distribution $p_0(o|\cdot)^{\mathcal{T}}$, to the contexton collected at time-step t, which is rolled out by the conditioned policy $\pi_\theta(a|o, z)$.

Demonstrations significantly reduce the search space in exploration whilst providing a natural means of communicating the goal of the task. Thus, access to expert trajectories provides information-rich context which can be exploited to pre-condition the policy. Furthermore, online adaptation aims to disambiguate the RL problem further. In our approach we leverage dual meta-learning objectives to perform hybrid adaptation to an unseen task conditioned on both demonstration and exploratory contexts:

Primal Inference: The agent observes a set of $k = 1, ..., K$ demonstrations from an expert set $D_{demo} := \{d_k\}$ to form a context prior $c_{demo} = \{d_k\}_k^K$.

Exploratory Adaptation: The agent explores with initialised context $c \leftarrow c_{demo}$ and adapts with sampled trajectories τ such that at time-step t, a hybrid context is formed $c \leftarrow c_{demo} \cap \tau_{0:t}$.

where we define a demonstration as a trajectory of T observations and actions $d = \{(o_t, a_t)\}_{t=0}^T$, and an adaptation episode τ_e as a trajectory of N steps $\tau_e = \{(o_t, a_t, r_t, z_t)\}_{t=0}^N$. Inspired by hippocampal memory retention, primal task inference conditioned on demonstrations from a new task provides long-term inference on a task belief. Using posterior sampling, exploratory adaptation provides a framework for short-term memory: as the context is updated with recent transitions, the long-term belief of the task becomes more redundant, allowing structured exploration to adjust to imperfect demonstrations or under-confident task beliefs $z \sim p(z|c)$ by reasoning about the uncertainty of z, where $c := \{c_{demo} \cap (\tau_e^T, ..., \tau_e^{\infty T})\}$.

One of the caveats of RL which make it impractical to real conditions is that, in many cases, designing a reward function not only requires iterative laborious work, but can also result in unnatural behaviours [30]. Most importantly, in complex tasks as those observed for dexterous manipulation, learning from sparse or even hand-tuned dense reward functions can lead to prohibitive sample inefficiencies and eventually failure to learn the task [23]. Finally, reward functions are not commonly available during adaptation to new tasks. To that end, PERIL aims to use the synergies of both meta-RL and meta-IL approaches to perform efficient and generalisable adaptation to unseen tasks in sparse reward settings.

4.2 PERIL

4.2.1 Latent Context Inference

Probabilistic meta-RL aims to produce latent descriptors of a task via a probabilistic context variable z, which subsequently conditions the otherwise partially observable MDP. Meta-training consequently consists in finding optimal parameters which can adapt latent variables z from a recent history of collected and demonstrated experiences, *i.e.* the context c, in an arbitrary task $\mathcal{T} \sim p(\mathcal{T})$ to produce a policy $\pi_\theta(a|o,z)$ which maximises the expected sum of discounted rewards.

As we do not have access to the posterior $p(z|c)$, we leverage variational inference methods to produce an approximation to the true posterior $q_\phi(z|c)$. Through generative processes, we can sample $z \sim q_\phi(z|c)$ and optimise the parameters ϕ by maximising a meta-objective conditioned on z, allowing agents to explore similar tasks in a structured manner.

As outlined in section 2.2.2, Variational Encoders (VEs) are capable of producing latent distributions which optimise for an arbitrary task-dependent goal $\mathcal{G}(\mathcal{T}|\cdot)$. $\mathcal{G}$ will be defined in this chapter in an incremental manner. For the time being, consider it as a proxy for the global probabilistic meta-RL objective of maximising expected rewards (2.26). In our approach, we average the expectancy over a set of tasks drawn from $p(\mathcal{T})$ to formulate the meta-objective 4.1, where β controls the IB constraint for mutual information between the inferred variable $\mathcal{Z}$ and context c.

$$\phi^* = arg\max_\phi \mathbb{E}_{\mathcal{T}\in p(\mathcal{T})}\big[\mathbb{E}_{z\sim q_\phi(z|c^\mathcal{T})}\big[\mathcal{G}(\mathcal{T}|z) + \beta D_{KL}\big[q_\phi(z|c^\mathcal{T})||p(z)\big]\big] \tag{4.1}$$

In theory, the prior distribution over the latent context variable $p(z)$ could correspond to any arbitrary distribution. In practice, we use a unit Gaussian prior $p(z) := \mathcal{N}(0, \mathbb{I})$ as commonly suggested in VE setups for deep learning [72, 73]. In essence, the IB constrains the distributional shift from the information-less prior, acting as a regularisation technique to filter down redundant task-related information and compress the latent context space into a generalisable form.

Inference Modelling

Traditional meta-RL methods leverage RNN-based inference systems with the aim of inferring latent features from a history of recently collected transitions. The problem with this approach is that learning from entire trajectories leads to massive variances which hinder the learning process [38, 52]. Moreover, as we use inference to produce a task belief, we do not benefit from recurrent systems as empirically shown in [51]. Although counter-intuitive, this must be true, especially in off-policy RL, since any MDP can be strictly defined by a set of sequentially invariant transitions stored in a replay buffer. In contrast, Rakelly et al. [51] suggest inference modelling for $q_\phi(z|c_{1:N})$ using a product of Gaussian Factors:

$$q_\phi(z|c_{1:N}) \propto \prod_{n=1}^{N} \psi_\phi(z|c_n)$$

where each factor $\psi_\phi(z|c_n)$ parametrised by ϕ is independently computed:

$$\psi_\phi(z|c_n) = \mathcal{N}(f_\psi^\mu(c_n), f_\psi^\sigma(c_n))$$

The problem we identify with this approach is that it assumes that each transition collected from the environment is independent from the other, hence there is no message passing from one contexton to another (Figure 4.1).

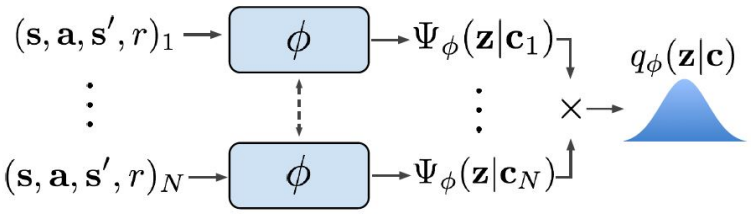

Figure 4.1: Inference network architecture for PEARL. Prior experiences are all conditioned on the same parameters ϕ which eliminates the possibility to reason amongst transitions. Extracted from [51].

Unfortunately, this assumption is not generalisable as observing a set of arbitrary transitions can already discard the possibility of belonging to certain task families. For instance, if an agent is exploring and suddenly encounters a wall, the transition which stores this contact is going to suggest that the task belongs to a contact-rich family, whilst the previous transitions infer navigation in free space. This example also brings light to the fact that using factorised distributions may not be ideal since not all transitions are equally important in recovering a task belief z. This is especially applicable to our case considering we leverage sparse rewards to train the encoder. For that matter, particular emphasis should be given to goal-completing contextons. We identify several requirements for powerful task inference:

1. Input-size agnostic: Task inference should be unbiased to varying context sizes in order to allow continuous adaptation.
2. Permutation invariance: The order of the contextons stored in c should not affect task inference.
3. Interstitial-awareness: Each contexton must influence the extraction of information on other conexTons.
4. Learnt Emphasis: Allow self-attention mechanisms to pinpoint task-related relations.

With these objectives in mind, we draw our attention to Latent Graph Neural Networks (LatentGNNs) [74] for inference modelling as proposed in [63]. The latter model provides a flexible graph affinity approach to learn non-local relations between contextons via weighted-sum aggregation and latent spaces.

The task encoder is a latent bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodal space $\mathcal{V} = \mathcal{V}^i \cap \mathcal{V}^h$ can be divided into $\mathcal{V}^i$ input nodes and $\mathcal{V}^h$ latent nodes. We define n and d as the number of input and latent nodes $|\mathcal{V}^i|, |\mathcal{V}^h|$ such that $|\mathcal{V}| = n + d$. The number of input nodes n can be any arbitrary number (input-size agnostic objective), whilst d is fixed. Nodal features $X \in \mathbb{R}^{n \times c}$ are assigned to $\mathcal{V}^i$, where $x_\nu \in \mathbb{R}^c$ represents the feature vector of $\nu \in \mathcal{V}^i$. Similarly we introduce a set of latent node features $H \in \mathbb{R}^{d \times c}$

assigned to $\mathcal{V}^h$, where $h_\nu \in \mathbb{R}^c$ represents a c-channel feature vector of $\nu \in \mathcal{V}^h$, and the number of latent features $d << n$. Note that the dimensions of c match the dimension of the contexton $\{o, a, r, o\prime\}$ such that X represents the context c used in task inference.

In $\mathcal{G}$, we also define bipartite edges $\mathcal{E} = \mathcal{E}^i \cap \mathcal{E}^h$ where $\mathcal{E}^i$ denotes the connection between $\mathcal{V}^i$ and $\mathcal{V}^h$ (weighted aggregation), whilst $\mathcal{E}^h$ denotes the graph edges within $\mathcal{V}^h$ (self-attention). The computational steps in LatentGNN are threefold: (i) Context-to-latent propagation; (ii) Latent-to-latent propagation; (iii) Latent-to-belief propagation. Figure 4.2 illustrates the computational graph. Note that we propose a LatentGNN inspired by that in [63]. However, the key difference is that in our approach we leverage MLPs to provide non-linear affinity functions in both context-to-latent and latent-to-belief stages.

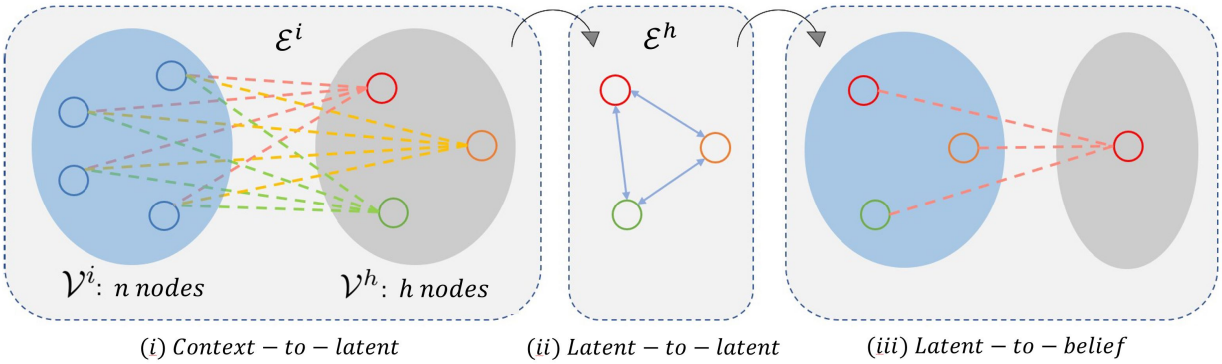

Figure 4.2: Computational graph of the LatentGNN task encoder $q_\phi(z|c)$. Adapted from [63].

In the first computation of $\mathcal{G}$ we perform context-to-latent message passing via weighted sum aggregation (4.2):

$$h_k = \sum_{j=1}^{n} \psi(x_j, \phi_k) x_j, \quad 0 \leq k \leq d \tag{4.2}$$

where $\psi(x_j, \phi_k)$, parameterised by ϕ_k, encodes the affinity between node x_j and latent node h_k. We use a mapped version of the normalised dot-product as a form of affinity function, resulting in computation (4.3). Note that Φ is modelled as an MLP which maps $\Phi(X) \rightarrow d \times n$.

$$H = softmax(\Phi(X))X \tag{4.3}$$

Latent-to-latent propagation is subsequently computed via a self-attention layer (4.4), where $\overline{h}_k$ denotes the attention product of latent node h_k.

$$\overline{h}_k = \sum_{i=1}^{d} f(h_i, h_k) h_k, \quad 0 \leq k \leq d \tag{4.4}$$

More specifically, we leverage scaled dot-product attention to produce the latent space $\overline{H}$ of $\mathcal{G}$, where $\mathcal{J}(\cdot)$ is a standard batch normalisation kernel.

$$\overline{H} = sotmax(\mathcal{J}(H)\mathcal{J}(H)^T)H \tag{4.5}$$

Finally, we perform a second weighted sum aggregation over the latent space $\overline{H}$ (4.6), where Ψ is modelled as an MLP similar to that in the first aggregation layer.

$$z_L = softmax(\Psi(\overline{H}))\overline{H} \tag{4.6}$$

The output $z_L \in \mathbb{R}^c$ is the latent task belief which is ultimately generated from context c. In order to translate LatentGNN into a variational inference model, we linearly map Z_L using a set of two independent MLPs to produce vectors z_μ, z_σ of dimensions $\mathbb{R}^f$. These parameters define the distribution of $q_\phi(z|c)$. Note f defines the vector space of z and can be any arbitrary number (hyper-parameter). The computation of $\mathcal{G}$ can be viewed as a multi-stage encoder which (i) processes inputs into a set of summaries; (ii) finds self-similarities from within these summaries; (iii) ultimately produces a final concise representation z of the context c.

4.2.2 Probabilistic Embeddings for Meta-Reinforcement Learning

PERIL is built on top of the formulation for maximum entropy RL (MaxEnt RL), where a proxy for the policy $\pi(a|s)$ is defined by including the task belief as part of the observational space $\pi_\theta(a|o, z)$. The main reasons for investing MaxEnt RL instead of pure RL reward maximisation are twofold: (i) Rewarding increased state space coverage through $\mathcal{H}(\cdot|s)$ aims to reduce distributional shift at test time by allowing the policy to be robust to otherwise unknown observations; (ii) Rewarding exploration rather than exploitation is crucial in sparse or under-defined reward settings, where credit assignment problems are inherent. We adapt a MaxEnt RL formulation (sec. 2.1.1) to define a meta-objective for task $\mathcal{T} \in p(\mathcal{T})$ (4.7).

$$\pi^* = arg\max_\pi \sum_{\mathcal{T}\in p(\mathcal{T})} \mathbb{E}_{c\sim\mathcal{T}}\left[\sum_{t\in T} \mathbb{E}_{z\sim p(z|c),(o_t,a_t)\sim\pi^{\mathcal{T}}}\left[r(o_t, a_t, z) + \alpha\mathcal{H}(\pi(\cdot|o_t, z)\right]\right] \tag{4.7}$$

By keeping the task belief z constant along each trajectory trajectory roll-out of length T we allow agents to develop structured exploration, akin to posterior sampling. Since we are training off-policy agents, we leverage the soft actor critic (SAC) formulation, a method to develop off-policy actor-critics in the MaxEnt RL setting [35]. SAC has demonstrated state-of-the-art performance in various RL benchmarks for both stability and sample efficiency. Moreover, as suggested in PEARL, due to its probabilistic background, SAC can be folded adequately with probablistic latent contexts [51].

Optimising the objective defined in (4.7) is possible by using the SAC soft policy iteration approach. The difference to standard SAC is that in our case, the observation is augmented by z resulting in task-conditioned critic (4.8) and actor (4.9) losses for the the Q-function $Q_\theta(\cdot)$ and the policy $\pi_\theta(\cdot)$ respectively. You may seek further details on the derivation of actor and critic losses in Appendix B of [35].

$$\mathcal{L}^{\mathcal{T}}_{critic} = \mathbb{E}_{c\sim\mathcal{T},z\sim q_\phi(z|c),(o,a,r,o\prime)\sim\mathcal{B}^{\mathcal{T}}}\left[Q_\theta(o, a, z) - (r + \overline{V}(o\prime, \overline{z}))\right]^2 \tag{4.8}$$

Where $\overline{V}$ corresponds to the frozen target value function (no gradients stored in the forward pass).

Notice we use the term $\mathcal{B}^{\mathcal{T}}$ to represent the distribution of transitions in task $\mathcal{T}$, which is modelled by a replay buffer. Moreover, the over-line operator in $\overline{z}$ denotes that gradients are detached.

$$\mathcal{L}^{\mathcal{T}}_{actor} = \mathbb{E}_{c\sim\mathcal{T}, z\sim q_\phi(z|c), o\sim\mathcal{B}^{\mathcal{T}}, a\sim\pi_\theta(a|o,z)}\left[D_{KL}\left(\pi_\theta(a|o,\overline{z}) \middle|\middle| \frac{\exp(Q_\theta(o,a,\overline{z}))}{\mathcal{Z}_\theta(o)} \right) \right] \tag{4.9}$$

Where $\mathcal{Z}_\theta(o)$ is the normalising partition function, which is intractable yet has no effect on the computation of the gradients. In the computation of $\mathcal{L}^{\mathcal{T}}_{actor}$, the gradients for $\overline{z}$ are detached, allowing the policy loss to exploit the representation of the MDP passed on by the critic without conditioning the inference network $q_\phi(z|c)$.

Observe in 4.8 we allow gradients from the inference process $q : C \xrightarrow{\phi} Z$ to pass onto the computation of $Q_\theta(o,a,z)$. As VEs are a form of generative processes, task-dependent goals $\mathcal{G}(\mathcal{T}|\cdot)$ can be defined with the aim of generating a model of the reward function or the transition dynamics [52]. Instead, we optimise $q_\phi(z|c)$ in a model-free manner as proposed in [51]. Here, parameters ϕ can be optimised to maximise expected discounted rewards under a policy $\pi_\theta(\cdot|z)$, or rather recover the state-action value function $Q_\theta(\cdot,z)$. With the aim of stabilising meta-training, we choose the latter option. By doing so, the critic loss can be added to the computational graph which controls parameters ϕ. In essence, we fold $\mathcal{L}^{\mathcal{T}}_{critic}$ as a proxy for the task-dependent goal $\mathcal{G}(\mathcal{T}|z)$ as suggested in PEARL:

$$\mathcal{G}(\mathcal{T}|z) \leftarrow \mathcal{L}^{\mathcal{T}}_{critic}(z) \tag{4.10}$$

Auxiliary Module

An inherent problem in the two-player game (encoder, actor-critic) is that the algorithm generates predictions $Q_\theta(o,a,z)$ based on unsupervised estimates $z \sim q_\phi(z|c^{\mathcal{T}})$ which lead to disambiguation difficulties. Whilst this is theoretically generalisable to any MDP, it brings large variances and instabilities during training as the agent must simultaneously figure out (i) how to distinguish a task from another, and (ii) how to use task beliefs to solve that task. In order to mitigate these instabilities, we exploit privileged information during meta-training to allow the encoder $q_\phi(z|c^{\mathcal{T}})$ to produce task beliefs z which provide succinct descriptions of $\mathcal{T}$. Privileged information must be provided by the designer and can consist in a brief task descriptor such as the position and orientation of a door handle.

In essence we model a ground truth task descriptor $\hat{b}$ for task $\mathcal{T} \in p(\mathcal{T})$ as a vector $\hat{b} \in \mathbb{R}^v$ where v is the dimension of the task descriptor. During training we wish to condition the encoder to produce latent spaces $z \sim q_\phi(z|c^{\mathcal{T}})$ which, when mapped into the dimensions of $\hat{b}$, can produce approximations of the task descriptor b. We use an MLP $d_\lambda(b|z)$ parametrised by λ as our auxiliary module, which produces vectors $b_\mu, b_\sigma \in \mathbb{R}^v$. As we output a distribution, we set log-likelihood maximisation as the objective (4.11).

$$\mathcal{L}^{\mathcal{T}}_{aux} = -\mathbb{E}_{(\hat{b},c)\sim\mathcal{T}, z\sim q_\phi(z|c)}\big[\log d_\lambda(b = \hat{b}|z) \big] \tag{4.11}$$

Notice that task descriptor inference $b \sim d_\lambda(b|z)$ from the trajectories in $c^\mathcal{T}$ is decoupled from RL (gradients do not pass through the actor or the critic). This stabilises the encoder by having a fixed supervised target, and it also allows training in off-policy settings. In theory, a perfect task descriptor is such that it provides sufficient information to define the MDP [52]. However, in PERIL we use simple descriptors such that these solely condition the encoder to converge towards a suggested direction. For that matter, we extend the proxy for the task-dependent goal $\mathcal{G}(\mathcal{T}|z)$ with the auxiliary loss:

$$\mathcal{G}(\mathcal{T}|z) \leftarrow \mathcal{L}^{\mathcal{T}}_{critic}(z) + \mathcal{L}^{\mathcal{T}}_{aux}(z) \tag{4.12}$$

Leveraging auxiliary objectives for meta-RL was coined by Humpling et al., where the authors condition task inference solely on $\mathcal{L}^{\mathcal{T}}_{aux}$ to train RNN-based policies [52]. However, our approach is novel as we exploit auxiliary losses as a supplementary aspect to our meta-objective, with the unique aim of stabilising meta-training. Moreover, by keeping the critic loss $\mathcal{L}^{\mathcal{T}}_{critic}$ in $\mathcal{G}(\mathcal{T}|z)$, we give freedom for the encoder to perform unsupervised learning such that it can separate the latent representations of tasks with different dynamics and sparse rewards. Note that if we were to solely base task inference on $\mathcal{L}^{\mathcal{T}}_{aux}$, a task such as screwing a bolt would output the same task belief as a task of inserting a peg, even though the tasks are clearly different. In light of this observation, we speculate that by following the approach in [52], PERIL would not generalise well to unseen tasks.

4.2.3 Conditioning on Demonstrations

Learning from demonstrations typically requires a vast amount of expert trajectories and can result in over-fitted behaviour, where unexpected events or slight changes in the initial state distribution results in collapsing policies. With meta-learning, we suggest that the agent can learn how to exploit the overlapping statistical information from within heterogeneous demonstrations belonging to different tasks, with the aim of inferring task embeddings z which can help improve exploration during test-time adaptation.

In order to link demonstrations into the probabilistic meta-RL framework we formulate an objective based on mutual information between the demonstration trajectories τ_E and the latent space distribution $p(z)$ (4.13). This approach is similar to that used in [70] with the exception that we adapt this to off-policy RL via importance sampling, and do not use generative processes to generate reward distributions. Recall that mutual information $\mathcal{I}(z;\tau)$ (otherwise referred to as the information gain) gives a measure of the dependency from one variable to another. Our main motivation of using $\mathcal{I}(z;\tau)$ instead of simply conditioning the policy on BC is that, as we want the agent to infer task beliefs from demonstrations, we must provide a means to link the distribution of z to that of the expert demonstrations. In the contrary, BC could simply choose to ignore variable z.

$$I(z;\tau) = \mathbb{E}_{z\sim p(z),\tau\sim p_\theta(\tau|z)}\big[\log p_\theta(z|\tau) - \log p(z)\big] \tag{4.13}$$

Where the conditional distribution $p_\theta(\tau|z)$ is approximated by the policy $\pi_\theta(\tau|z)$. As direct access to the posterior $p_\theta(z|\tau)$ is not available, $I(z;\tau)$ is intractable. However, in order to adjust for this, we

leverage a variational approximation to $p_\theta(z|\tau)$ using our task belief encoder $q_\phi(z|\tau)$. We assume access to a distribution of expert demonstrations $p_{\pi_E}(\tau|z)$, also defined as $p_{\pi_E}(\tau)^{\mathcal{T}}$ for each task $\mathcal{T} \in p(\mathcal{T})$. Additionally, we include further desiderata over the mutual information objective based on distributional matching:

1| Learning from demonstrations. Matching generated trajectories to those sampled from expert trajectories: $\min_\theta \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)]$. This secondary objective can be considered as trying to match the distribution of trajectories generated by the agent's policy conditioned on z, to those from the expert policy (similar to BC). Note that because they share the same marginal distribution $p(z)$, matching these distributions also encourages matching of the conditional distributions $p_{\pi_E}(z|\tau)$ and $p_\theta(z|\tau)$.

2| Linking variational posterior. Matching posterior distributions: $\min_\theta \mathbb{E}_{p_\theta(\tau)}[D_{KL}(p_\theta(z|\tau)||q_\phi(z|\tau)]$. This objective encourages the encoder $q_\phi(z|\tau)$ to be a good approximation to the real posterior distribution $q_\theta(z|\tau)$ such that given a new demonstration, the encoder properly infers the task. In essence we believe this objective acts as a regulator as it effectively reduces over-fitting by reducing variance (optimal distributional mismatch D_{KL} is reached when the encoder outputs a constant, information-less z).

The distributional matching objective can therefore be defined as:

$$\begin{aligned}
&\min_{\theta,\phi} \quad -I_{p_\theta}(z;\tau) + \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)] + \mathbb{E}_{p_\theta(\tau)}[D_{KL}(p_\theta(z|\tau)||q_\phi(z|\tau)] \\
&= \min_{\theta,\phi} \quad \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)] + \mathbb{E}_{p_\theta(m,\tau)}\left[\log \frac{p(z)}{p_\theta(z|\tau)} + \log \frac{p_\theta(z|\tau)}{q_\phi(z|\tau)}\right] \qquad (4.14) \\
&= \min_{\theta,\phi} \quad \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)] + \mathbb{E}_{z\sim p(z),\tau\sim p_\theta(\tau|z)}\Big[\log q_\phi(z|\tau)\Big]
\end{aligned}$$

In order to optimise the two terms in (4.14), we define $\mathcal{L}_{BC}(\theta)$ and $\mathcal{L}_{info}(\theta,\phi,z)$ as the leftmost and rightmost distribution expectancies respectively. In this definition, the expectancies are not differentiable and requires parametric manipulation. The first term $\mathcal{L}_{bc}(\theta)$ can easily be adapted to the meta-RL formulation by substituting $p_\theta(\tau|z)$ for the policy $p_\theta(\tau|z)$ and deconstructing the D_{KL} (novel). Naturally, this results in a loss similar to that of BC (we derive this in see section 9.1.1 of the appendix).

$$\mathcal{L}_{bc}^{\mathcal{T}} = -\mathbb{E}_{\tau\sim p_{\pi_E}^{\mathcal{T}}(\tau),z\sim q_\phi(z|\tau)}[\log \pi_\theta(\tau|z)] \qquad (4.15)$$

By a similar set of substitutions, the second term $\mathcal{L}_{info}(\theta,\phi,z)$ can be evaluated as follows:

$$\mathcal{L}_{info}^{\mathcal{T}} = -\mathbb{E}_{\tau\sim p_{\pi_E}^{\mathcal{T}},z\sim q_\phi(z|\tau),\tau_\theta\sim\pi_\theta(\tau|z)}\Big[\log q_\phi(z|\tau_\theta)\Big] \qquad (4.16)$$

Notice how in $\mathcal{L}_{info}^{\mathcal{T}}$ we generate a set of trajectories τ_θ from a task belief sampled from expert demonstration trajectories. From this we aim to maximise the likelihood of matching the posteriors on z given τ_θ. In essence, this can be viewed as a two-player game: we want to match the z generated by demonstrations to the z generated by a policy which is also conditioned on the demonstrations.

Even though this loss function can be directly computed, it is incompatible with off-policy RL: notice that the policy rollout τ_θ is sampled from the current policy π_θ. In order to accommodate off-policy RL we propose truncated importance sampling (TRIS) by sampling from the distribution which collected τ_θ (novel). In practice, we use a behavioural policy $\pi_b(\tau|z)$ which is updated every episode as a frozen (no gradients) deep copy of π_θ. Transitions in τ_b are sampled off-policy at the start of every episode.

$$\mathcal{L}^{\mathcal{T}}_{info} = -\mathbb{E}_{\tau\sim p^{\mathcal{T}}_{\pi_E}, z\sim q_\phi(z|\tau), \tau_b\sim\pi_b(\tau|z)}\left[\min\left(\frac{\log\pi_\theta(\tau_b|z)}{\log\pi_b(\tau_b|z)}, 1\right)\log q_\phi(z|\tau_b)\right] \tag{4.17}$$

As $\mathcal{L}^{\mathcal{T}}_{info}$ conditions the encoder, we aggregate it to the task-dependent goal $\mathcal{G}(\mathcal{T}|z)$ to produce the ultimate meta-objective of our proposed method:

$$\mathcal{G}(\mathcal{T}|z) \leftarrow \mathcal{L}^{\mathcal{T}}_{critic}(z) + \mathcal{L}^{\mathcal{T}}_{aux}(z) + \mathcal{L}^{\mathcal{T}}_{info}(z) \tag{4.18}$$

An overall illustration of how each loss defined in this chapter contributes to the computational graph which defines PERIL is presented in Figure 4.3.

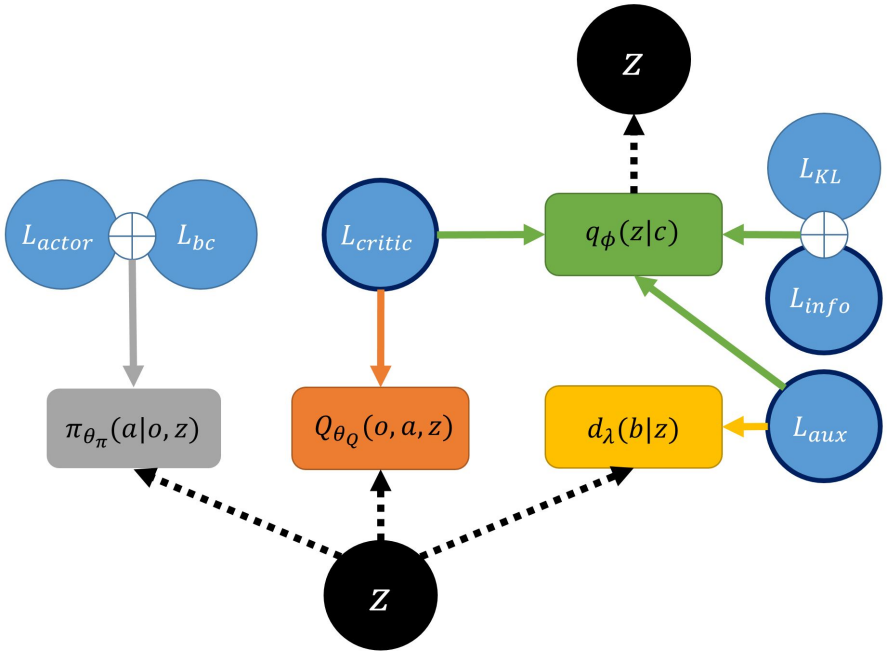

Figure 4.3: Illustration of the computational graph of PERIL. Loss gradients (defined by blue circles) propagate to their corresponding models. These include the task encoder q_ϕ, actor π_{θ_π}, critic Q_{θ_Q}, and auxiliary module d_λ. Black circles denote placeholders where latent task beliefs z can be updated (arrow head pointing inwards) or evaluated (arrow head pointing outwards). The operator $\bigoplus$ denotes summation of the loss gradients. The colour of the arrows pointing outwards from the losses indicates the model dependency for that loss. Notice critic and auxiliary losses are shared within two models. Last, the black outlines denote the loss gradients which contribute to the meta-objective $\mathcal{G}(\mathcal{T}|z)$.

Chapter 5: Implementation

In this chapter we describe the implementation details of PERIL by presenting an overview of the system followed by a detailed analysis on the training and testing algorithms behind it. Furthermore, we introduce the set of environments and benchmarks created to evaluate our method.

5.1 Framework

5.1.1 Meta-Training

During meta-training we assume full access of the environment such that we can recover privileged information on the task descriptors. Note that this is something which is only plausible in quasi-perfect conditions (training in simulation or using online tracking systems which may identify the task descriptor). In our case, we perform meta-training in simulated environments. Rather than hand-tuning task-dependent reward functions, we use naive reward functions which are not conditioned on task dynamics. These reward functions are only used to train the SAC agent, whilst we feed sparse rewards to the task encoder with the attempt of inciting real-life conditions.

Meta-training for probabilistic RL requires sampling from a set of transitions $(o_t, a_t, r_t, o_{t+1})_{t=0}^{T}$ corresponding to a distribution of tasks $\mathcal{T} \in p(\mathcal{T})$ and optimising the computational graph which computes the meta-objective from the latter transitions. Since our method is compatible with off-policy RL, we use task-specific replay buffers $\mathcal{B}^{\mathcal{T}}$ to store collected transitions from which we can randomly sample during training. As motivated in the previous chapter, given a recent set of sparsified transitions, or otherwise referred to as contextons, PERIL aims to infer a relevant task descriptor z. Following the primal inference objective outline in section 4.1, task inference is initially conditioned via contextons retrieved from expert demonstrations. Consequently, we store a set of k demonstrations generated by an expert policy on task-conditioned demonstration buffers $\mathcal{D}^{\mathcal{T}}$. Details on how we obtain expert demonstrations will be presented in section 5.2.4.

On the other hand, following the continuous task adaptation objective, the agent must also exploit recently collected samples to update its task belief. For that matter, we define a proxy for posterior sampling in our meta-training algorithm and use exploratory contextons to update the task belief. In light of this, we set additional task-dependent encoder buffers $\mathcal{X}^{\mathcal{T}}$ which store recently collected contextons.

The issue with sampling context from outdated transitions is that this presents with a distributional shift between off-policy training and on-policy adaptation. Inherently, if we were to sample exploratory context

from the standard RL replay buffer $\mathcal{B}^{\mathcal{T}}$, the contextons collected would belong to different of policies π_θ, resulting in little structure and consequently high task uncertainty. Not only does this hamper structured exploration, but it is also too dissimilar to what the agent would use to adapt to an unseen task. For that matter, task encoder buffers $\mathcal{X}^{\mathcal{T}}$ are emptied before collection and so they contain quasi-on-policy contextons.

An illustration of the overall training process is presented in Figure 5.1. Notice the hourglass operator on the orange arrow which points from the trajectories generated by policy π_{θ_π} conditioned on demonstrations, to the task-conditioned demo buffer $\mathcal{D}^{\mathcal{T}}$. This operator is a filter which processes the quality of the trajectory generated by the policy and evaluates if the trajectory should be categorised as "expert" or not. As training proceeds and the encoder starts producing meaningful task beliefs, policy rollouts during posterior sampling may lead to superior discounted rewards than those available. In this case, the filter categorises the trajectory as "expert" and concatenates it to $\mathcal{D}^{\mathcal{T}}$. We refer to this method as "demo augmentation".

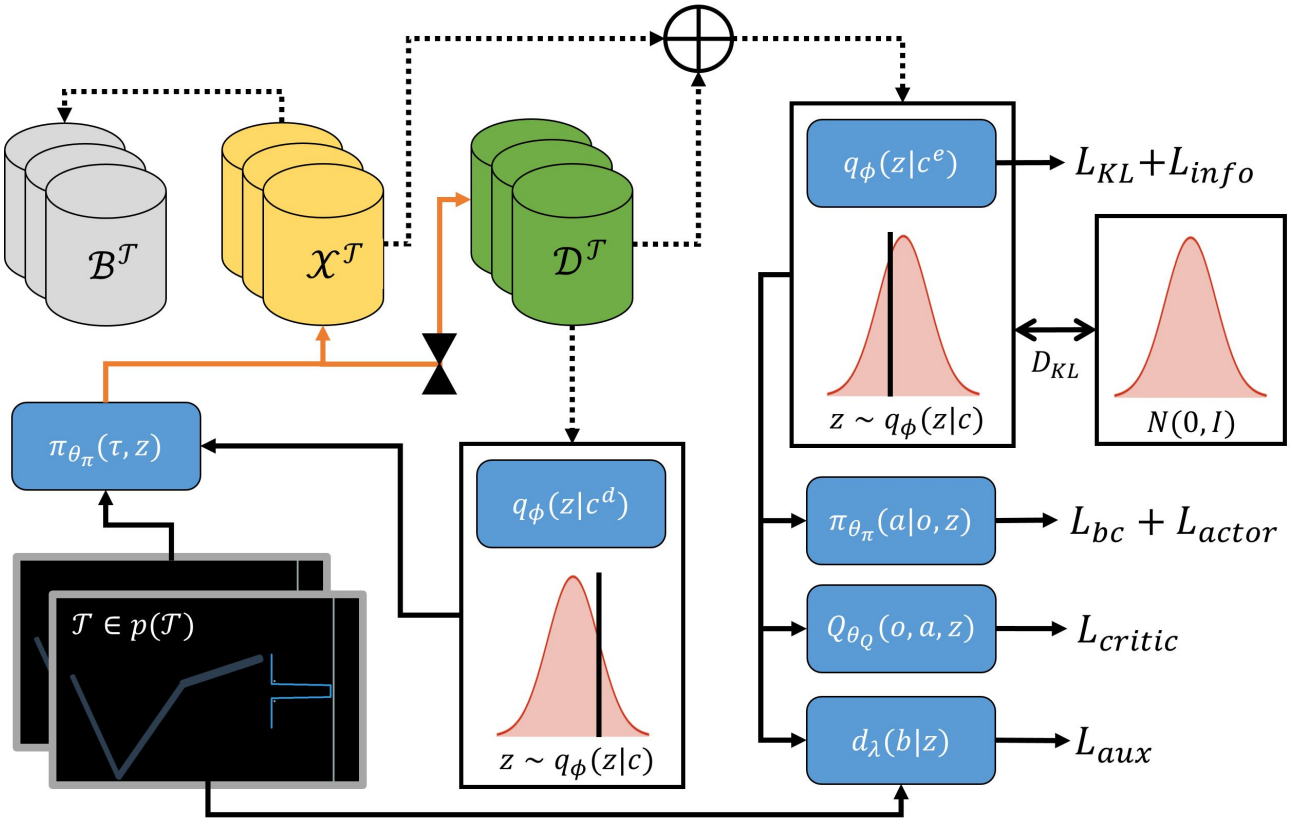

Figure 5.1: Illustration of the general processes behind the meta-training framework for PERIL. The environment contains a set of tasks $\mathcal{T} \sim p(\mathcal{T})$, each with its own ground truth task descriptor b and initial state distribution $p_0(o|z)$. After being conditioned by the task-belief generated by the demonstrations, samples generated by the policy overwrite the encoder buffer $\mathcal{X}^{\mathcal{T}}$ for the current task $\mathcal{T}$. All samples are added to the replay buffer $\mathcal{B}^{\mathcal{T}}$. The operator $\bigoplus$ concatenates context from the demonstration and the task buffer to produce a mixed task belief z used for the forward computation of the losses.

Demo augmentation is introduced post analysis of the first evaluation of PERIL as a means of (i) reducing the inherent brittleness of performing BC with a limited set of demonstrations, and (ii) introducing

learning from imperfect demonstrations. Since adaptation in the real world is imperfect, we need PERIL to condition its exploration strategy in a way that imperfect task beliefs can be readily corrected during posterior sampling.

In essence, the task encoder must generalise task beliefs such that these solely provide a high-level representation of the unseen task, and to that end, we seek an encoder which can reason about the inherent uncertainty of a demonstration. To account for this need, we speculate that learning to reason about the uncertainty of a task from a limited set of k demonstrations presents too much of a distributional shift with the demonstrations which may be provided during adaptation. For that matter we augment the demonstration buffers with "novel" trajectories (Figure 5.2).

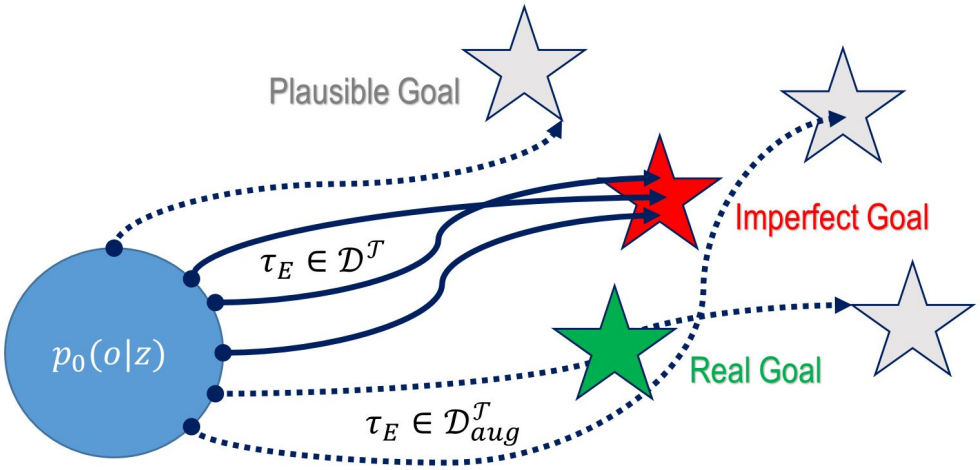

Figure 5.2: Illustration of the distribution of trajectories for goal-reaching tasks in the imperfect demonstration setting. A set of $k = 3$ demonstrations could be provided in real time. Yet, due to the inherent uncertainty of both the user and the robot, these demonstrations may be imperfect (reaching red star). By augmenting the demonstration buffer with trajectories generated in imperfect settings, we provide a means to help the task encoder reasoning about the natural uncertainty of a task given a demonstration.

In order to generate trajectories which cover the distribution of plausible tasks (see Figure 5.2) we introduce randomised alterations to each task before collection (more on this in section 5.2.1). The suitability of modelling each task as a POMDP is reflected in this approach: random alterations of a task would otherwise violate a task defined by an MDP.

Having access to privileged information $\hat{b}$ during meta-training allows pre-conditioning of the networks in PERIL. Note that this is an additional implementation which is novel. Since the auxiliary objective is decoupled from the RL framework, we leverage expectancy maximisation (EM) meta-pre-training to initialise the networks. In this setting, we aim to reproduce task descriptors from the context sampled from the demonstration buffers (E-step). We then sample task beliefs $z \sim q_\phi(z|c_{demo})$ to perform heavily regularised BC. We use regularised BC via L2 regularisation to ensure pre-training does not induce any overfitting and solely provides a good prior for meta-training.

Having provided a detailed deconstruction of the key functions of PERIL, we present this in detailed format in Algorithms 1 and 2. A description of the set of default hyperparameters used in our method is presented in section 9.2.1 of the appendix. To run the aforementioned training algorithms we use `Python` as the main programming language. More specifically, we use `PyTorch` backend to construct and optimise the computational graph which defines PERIL. Additionally, we base our data handlers and samplers on the open source `RLKit` library. Due to lack of direct access to the computers at the Robots Learning Lab, we used remote access to Google-Cloud services in order to accelerate training via GPUs (NVIDIA Tesla-T4).

Algorithm 1 PERIL

Require: batch of T training tasks $\{\mathcal{T}_i\}_{i=1,..,T}, \mathcal{T}_i \in p(\mathcal{T})$

Initialise π_θ, Q_θ, q_ϕ, b_λ and their learning rates α_π, α_Q, α_ϕ, α_b. Set L2 regularisation parameter κ.

Initialise replay buffers $\mathcal{B}_i^{\mathcal{T}}$, encoder replay buffers $\mathcal{X}_i^{\mathcal{T}}$ and demo replay buffers $\mathcal{D}_i^{\mathcal{T}}$ for $\mathcal{T} \in \{\mathcal{T}_i\}_{i=1,..,T}$

Meta-Pre-Training ▷ Pre-conditioner

`MetaPreTrain`$(\pi_\theta, q_\phi, b_\lambda, \mathcal{D}_i^{\mathcal{T}}, p(\mathcal{T}), \kappa)$

Meta-Training

while not done **do**

Collect Data

for each $\mathcal{T}_i, i \in \{1, ..., T\}$ **do**

Initialise the context by sampling demonstrations $c^i = \{\tau_E\}, \tau_E \sim \mathcal{D}_i$

Produce randomised task alteration on task $\mathcal{T}_i$ ▷ Imperfect demonstration case

$\mathcal{X}_i \leftarrow \emptyset$ ▷ Clear task encoder to reduce on-policy mismatch

for k = 1,...,K_{prior} **do** ▷ Demonstration-conditioned trajectories

Sample $z \sim q_\psi(z|c^i)$

Collect trajectories $\tau \sim \pi_\theta(\tau|z)$ and add to $\mathcal{B}_i$ and $\mathcal{X}_i$

end for

for k = 1,...,K_{post} **do** ▷ Exploration-conditioned trajectories via posterior sampling

Sample $z \sim q_\psi(z|c^i)$

Collect trajectories $\tau \sim \pi_\theta(\tau|z)$ and add to $\mathcal{B}_i$

$\mathcal{D}_i \leftarrow \{\mathcal{D}_i \cup \tau\}$ **if** $\mathbb{E}_R[\tau] > \mathbb{E}_R[\tau_E]$ ▷ Demo augmentation

Sample recent trajectories $x^i \in \mathcal{X}_i$ and update context $c^i \leftarrow \{c^i \cup x^i\}$

end for

end for

Meta-Optimisation ▷ N_t meta-training steps

for $i = 1, ..., N_t$ **do**

Sample R random training tasks from $p(\mathcal{T})$

Clone behavioural policy $\pi_b \leftarrow \pi_\theta$ ▷ To apply TRIS

for each $\mathcal{T}_i, i \in R$ **do**

Sample recent trajectories $x^i \sim \mathcal{X}_i$, $\tau_E \sim \mathcal{D}_i$ to form mixed context $c^i = \{\tau_E \cup x^i\}$

Sample RL batch $b_i \sim \mathcal{B}^i$. Sample $z \sim q_\phi(z|c^i)$

Compute $\mathcal{L}^i_{mi}(\tau_E, z), \mathcal{L}^i_{bc}(b^i, \tau_E, z), \mathcal{L}^i_{actor}(b^i, z), \mathcal{L}^i_{critic}(b^i, z), \mathcal{L}^i_{D_{KL}}(z), \mathcal{L}^i_{aux}(z)$

end for

$\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \sum_{i \in R}(\mathcal{L}^i_{critic} + \mathcal{L}^i_{D_{KL}} + \mathcal{L}^i_{mi} + \mathcal{L}^i_{aux})$

$\theta_Q \leftarrow \theta_\pi - \alpha_\pi \nabla \pi \sum_{i \in R} \mathcal{L}^i_{critic}$

$\theta_\pi \leftarrow \theta_Q - \alpha_Q \nabla_Q \sum_{i \in R}(\mathcal{L}^i_{actor} + \mathcal{L}^i_{bc})$

$\lambda \leftarrow \lambda - \alpha_\lambda \nabla_\lambda \sum_{i \in R} \mathcal{L}^i_{aux}$

end for

end while

In practice, every loss component defined in PERIL is weighted by a scalar w (specifics on section 9.2.1 of the appendix) which is used to increase or decrease their individual effect on the overall meta-objective.

Algorithm 2 Meta-Pre-Training

Inputs: Networks $\pi_\theta, q_\phi, b_\lambda$, task distribution $p(\mathcal{T})$, demonstration buffers $\mathcal{D}_i^{\mathcal{T}}$, L2 scalar weight κ.
function MetaPreTrain($\pi_\theta, q_\phi, b_\lambda, \mathcal{D}_i^{\mathcal{T}}, p(\mathcal{T}), \kappa$)
  for $t = 1, ..., (E + M)$ **do**
    Randomly select C tasks from $p(\mathcal{T})$ and sample the demonstrations $\{\tau_E^i\}_{i=1}^C$ from $\mathcal{D}_i^{\mathcal{T}}, i \in C$
    Sample latent space $\{z^i\}_{i=1}^C \sim q_\psi(z|\{\tau_E^i\}_{i=1}^C)$
    if $i \mod (E + M) < E$ **then** ▷ E-step
      Compute $\mathcal{L}_{aux}^i$.
      $(\phi, \lambda) \leftarrow (\phi, \lambda) - \alpha_{(\phi,\lambda)} \nabla_{(\phi,\lambda)} \sum_{i \in C} \mathcal{L}_{aux}^i(z^i)$
    else ▷ M-step
      Compute regularised BC loss: $\mathcal{L}_{bc}^i + \kappa ||\theta_\pi||_2^2$.
      $\theta_\pi \leftarrow \theta_\pi - \alpha_\pi \nabla_{\theta_\pi} \sum_{i \in C} \mathcal{L}_{bc}^i + \kappa ||\theta_\pi||_2^2$
    end if
  end for
end function

5.1.2 Online Adaptation

During online adaptation we wish to formulate a task belief z given a set of k demonstration of an unseen task $\mathcal{T} \sim p(\mathcal{T})$ and reason about the uncertainty of the demonstration to explore accordingly. In order to exploit short-term adaptation, a proxy for posterior sampling is desired such that it can use recently collected experiences to improve its belief upon the given task.

During the first steps of adaptation, since the agent has theoretically meta-learnt to reason about the uncertainty given from the demonstrations and adapt its policy accordingly, we use the optimised stochasticity of the soft actor and the inherent stochasticity present in the IB of the task encoder to drive exploration. Adaptation performance can be evaluated by the number of trajectories required to perform optimally in an unseen tasks.

A detailed description of the processes behind meta-testing is presented in Algorithm 3.

Algorithm 3 Meta-Test adapation

Require: Networks π_θ, q_ϕ, test task $\mathcal{T} \sim p(\mathcal{T})$, demonstration buffer $\mathcal{D}^{\mathcal{T}}$,
Initialise context by sampling k demonstrations $c = \{\tau_E\}_1^k \sim \mathcal{D}^{\mathcal{T}}$
for r = 1, ..., R **do**
  Sample $z \sim q_\phi(z|c)$
  Roll out policy $\pi_\theta(a|o, z)$ for N steps and collect trajectory $\tau_{post} = \{(o_t, a_t, r_t, o_{t+1})_{t=1}^N\}$
  Update context $c \leftarrow \{c \cap \tau_{post}\}$
end for

As is common in ML, distributional shift between training and testing datum presents a limitation to the extent of how much a task at test time can vary from the task distribution observed during meta-training. Indeed, to allow adaptation to novel tasks, there must be a minimal regime of overlapping

statistical information. In essence, the task encoder must be capable of interpolating from within the POMDPs encountered during meta-training. For instance, if we meta-train in tasks relating grabbing objects of different geometries, we cannot expect the agent to adapt to a demonstration of a task such as screwing a bolt. In the next section we will introduce the set of carefully devised task families used to evaluate PERIL.

5.2 Environments

Applications including peg insertion and lock & key manipulation are amongst other types of contact-rich task families envisioned for this project. Real robots particularly struggle in situations where models of the environment are particularly difficult to define. Effectively, many robotic tasks may involve multiple discrete boundaries which impart solving complex dynamical model. Furthermore, DNNs, amongst other statistical models, are particularly susceptible to abrupt discontinuities. Thus, it is of particular interest to leverage meta-training in situations where identification of the transition dynamics is key. For that matter, we aim to produce tasks which involve contact-rich interactions and discontinuities, factors which are considered to require a higher level of understanding and perception of the the environment [75].

In order to test and iterate our proposed method, we develop a set of environments compatible with the OpenAI framework. We build these environments using `Pymunk` backend, a `Python` ready 2D physics simulator, and give visualisation support through `Pyglet`, a third party graphics library.

5.2.1 Task Families

A repertoire of distinct task families is devised with the goal of analysing adaptation to different conditions. Each task family shares a similar robotic agent defined as a multi-armed 2D manipulator. Since we want full control of the position of end effector, we use 3 joints to control the 3 DoF in 2D space. Depending on the task family, the geometry of the end effector may vary from one form to another.

Note that although differences from one task family to another are clearly distinguishable, differences from within a single task family are not so evident. In a nutshell, each task within a task family differs from one another in the following: (i) the position of the goal (G_x, G_y); (ii) the geometries of the interactive site (depth d, width w, insertion angles θ); (iii) simulated physical conditions (coefficient of friction, elasticity, damping). Recall that we impart demonstration imperfection via task alterations. Whenever we perform task alterations, we randomly re-initialise a specific task configuration with randomised perturbations on parameters (i) and (ii). Since `Pymunk` does not offer compatibility with the metric system, we define all spaces based on "pymunk units" (PyU).

Peg2D

Peg2D is a simple task family, where the objective is to insert an end effector inside a clearance hole (Figure 5.3). The task is completed once the end effector is fully inserted. To get a grasp of the PyU metric, note that the visual space of the following covers 1280 and 720 PyU in x and y respectively.

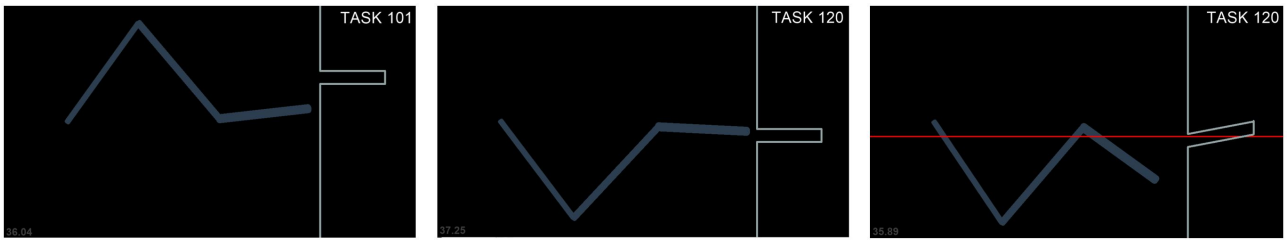

Figure 5.3: Snapshots of the Peg2D task family. Left and centre images represent different tasks within the task family. Rightmost image is a snapshot of the altered version of the task in the centre image, where the red line denotes the position and orientation of the unaltered task.

Stick2D

In Stick2D the end deflector must encapsulate an extrusion (Figure 5.4).

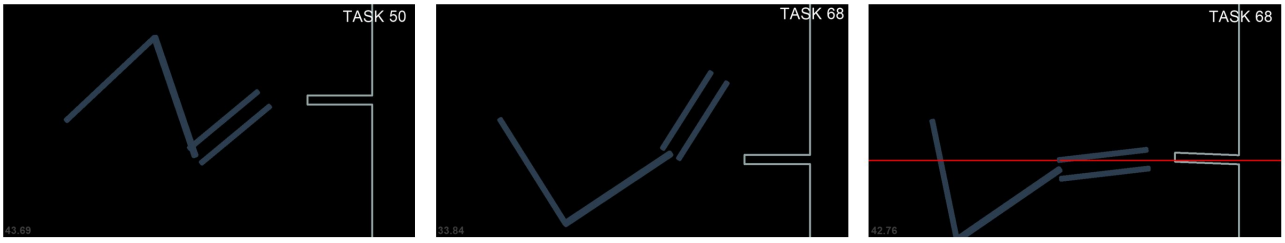

Figure 5.4: Snapshots of the Stick2D task family. Left and centre images represent different tasks within the task family. Rightmost image is a snapshot of the altered version of the task in the centre image, where the red line denotes the position and orientation of the unaltered task.

Key2D

Key2D is a slightly more complicated task family. Here, the end deflector (key) must first enter a static clearance hole (handle). Once the key reaches the whole depth of the handle, the handle unlocks and becomes dynamic. At this point the handle must rotate until it touches the wall (Figure 5.4).

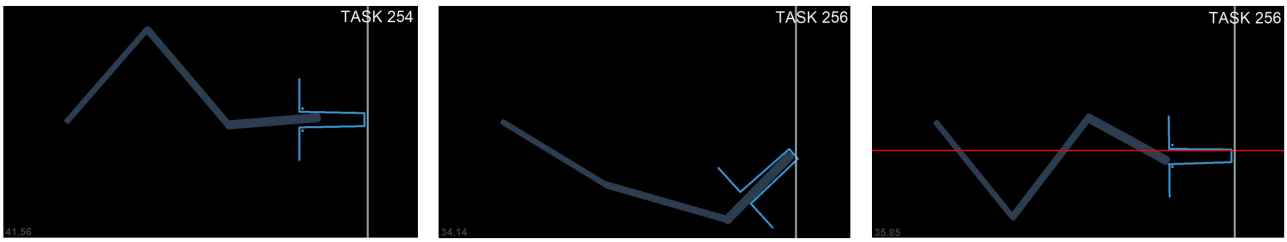

Figure 5.5: Snapshots of the Key2D task family. Left and centre images represent different tasks within the task family. Notice in the centre image we have unlocked the handle and it is thus free to rotate. Rightmost image is a snapshot of the altered version of the task in the centre image, where the red line denotes the position and orientation of the unaltered task.

Reach2D

Reach2D has the simplest dynamics. Nonetheless, its observational space is unconstrained in contrast to the other tasks, where a wall typically limits exploration in one dimension. Since we adapt through sparse rewards, this can prove to be challenging for exploration. Reach2D's larger observational space can create interesting difficulties during meta-training with other task families.

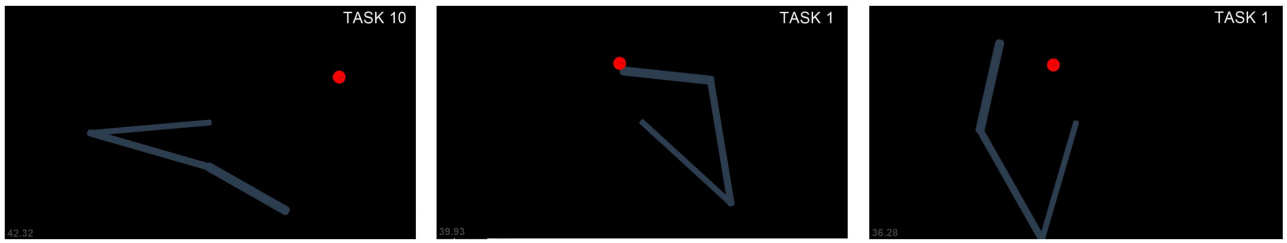

Figure 5.6: Snapshots of the Reach2D task family. Left and centre images represent different tasks within the task family. Rightmost image is a snapshot of the altered version of the task in the centre image.

5.2.2 Implementation Details

Alterations

When assuming imperfect demonstrations, each task $\mathcal{T} \in p(\mathcal{T})$ is altered such that the demonstration provided for the unaltered task is subsequently imperfect. Every task in MetaSim contains a set of uniform distributions $\mathcal{U}(\cdot)$ which define all the possible alterations we wish to impart on that task. These distributions could be task-specific but we reduce any chance of imparting biases by defining a global definition for the former (Table 5.1).

Table 5.1: Alterations over some elements which describe a task's condition. Some alterations are not compatible with specific task families.

Element	Compatibility	Alteration
(G_x, G_y) (PyU)	all	$(G_x, G_y) \leftarrow (G_x, G_y) + \mathcal{U}(-25, 25)$
θ (radians)	not Reach2D	$\theta \leftarrow \theta + \mathcal{U}(-\pi/6, \pi/6)$
w (PyU)	not Reach2D	$w \leftarrow w + \mathcal{U}(-5, 5)$
d (PyU)	not Reach2D	$d \leftarrow d + \mathcal{U}(-50, 50)$

Initialisation

Each task $\mathcal{T} \in p(\mathcal{T})$ also contains an initial observational distribution $p_0(o)^{\mathcal{T}}$. This distribution defines how the robot is initialised in the environment before collecting data. Since in this project we want to focus on meta-learning the dynamics of contact-rich tasks, we provide an initial distribution $p_0(o)^{\mathcal{T}}$ which allows the agent to start relatively close to the interaction (see the left snapshots in Figures 5.3 and 5.4). We summarise $p_0(o)^{\mathcal{T}}$ in Table 5.2, where P_0 and θ_0 represent the initial effector position (PyU) and angle (radians). Note that via inverse kinematics, the other links of the robot are automatically set.

Table 5.2: Initial observational distribution for all task families, where E denotes the point of interaction and is specific to each task family. In Peg2D and Key2D, E is defined as the clearence hole entry point, where in Stick2D, E is defined by the extrusion's tip.

Task Family	Distribution
Peg2D, Stick2D, Key2D	$100 < \lvert P_0 - E \rvert < 250$; $\lvert\theta_0\rvert < \pi/4$
Reach2D	$250 < \lvert P_0 - E \rvert < 500$; $\lvert\theta_0\rvert < \pi$

Spaces

Each task has an accessible ground truth task descriptor $\hat{b} \in \mathbb{R}^2$ which describes the exact position in space of its goal G. In all tasks, the base (origin) of the robotic manipulator is kept constant. The first component of the observational space $\mathcal{O}$ is defined by the normalised relative position from the tip of the end effector ($P \in \mathbb{R}^2$) to the origin Ω of the robot (5.1), where ω is the vector which defines the width and height in PyU of the visualisation domain (1280, 720).

$$o_{rel} = \mathcal{O}(P) = \frac{P - \Omega}{\omega} \tag{5.1}$$

The second component of the observational space corresponds to the sine and cosines of the angle of the end effector θ. The action space $\mathcal{A}$ gives complete freedom to the effector and it includes its cartesian velocity (v_x, v_y) (PyU/s) and angular velocity ω_θ (rad/s). Through inverse kinematics, the latter are translated back to the other links. Observation and action spaces are summarised in Table 5.3.

Table 5.3: State-action spaces for all task families.

Feature	Description
$\mathcal{O}$	$\{o^x_{rel}, o^y_{rel}, \cos\theta, \sin\theta\} \in \mathbb{R}^4$
$\mathcal{A}$	$\{v_x, v_y, \omega_\theta\} \in \mathbb{R}^3$

By definition of the observational space, the agent has no information about the position or condition of the goal. Hence, during adaptation the robot must seek to find the goal and identify the task dynamics using its own interactions as well as the information given by the demonstration. Note that in real world conditions, observation and action spaces could be augemented with kinematic components such as forces and torques at the end effector. However, in this simulated environment, we will show that deciphering the dynamics can be performed by using the aforementioned observational and action spaces.

Reward Functions

Naive dense reward functions are provided during meta-training of the critic. These reward functions are unshaped and give no information about the dynamics. In essence the reward function $r(\cdot)$ is defined as the absolute distance from the effector tip P to the goal G. The only exception is the Key2D task family, where reward functions are augmented with additional incentives to twist the handle to a desired angle α. Table 5.4 summarises the reward functions used in each task family.

Table 5.4: Dense reward function used for critic meta-training for all task families in MetaSim.

Task Family	Function
Peg2D, Reach2D, Stick2D	$\lvert G - P\rvert$
Key2D	$\lvert G - P\rvert + 0.5\lvert\theta - \alpha\rvert$

On the other hand, we only provide binary sparse rewards $r(\cdot) \in [0, 1]$ to the task encoder. These are 0 in all cases except when the effector has reached its goal, where we provide a reward of 1.

5.2.3 Benchmarks

In multi-task meta-learning the goal is to learn a transferable set of skills which can be used to learn new skills quickly. We wrap the aforementioned task families into a meta environment called MetaSim, with the inclusion of their orthogonal (vertical) equivalents (Figure 5.7). Note that each task family in MetaSim contains a set of 50 distinct tasks (40 for training, 10 for testing).

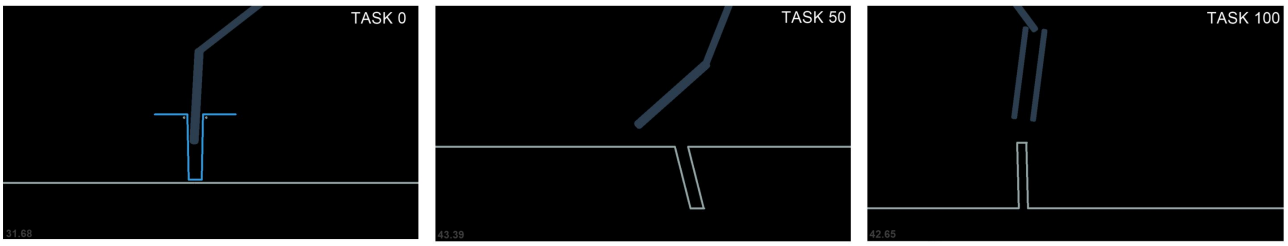

Figure 5.7: Vertical task family equivalents for Key2D, Peg2D and Stick2D (left to right).

To evaluate our method we devise 4 benchmarks from the set of 7 task families in MetaSim. With increased difficulties, different benchmarks require adaptation through a structured yet diverse set of transferable skills.

MLX

The multi-task meta-learning (MLX) setting involves meta-training on tasks from X task families where the conditions (goal location and geometries) of said task may vary. During test time we present with tasks from these task families which have not been seen during training. MLX particularly aims to evaluate reasoning and task identification, as well as multi-task generalisation.

- ML1 is based on adaptation of tasks within the Peg2D task family.
- ML4 is a more intricate meta-learning task, where we train on 4 different task families: Peg2D, Reach2D, Key2D and Stick2D.
- ML5 involves training on 3 different task families and 2 orthogonal equivalents: Peg2D, Stick2D, Reach2D, Peg2D (vetical), Stick2D (vertical).

MTL(X+Y)

Multi-task transfer meta-learning MTL(X+Y) aims to evaluate meta-learning via transfer learning. We train on tasks from X task families and test on tasks from Y different task families.

- In MTL(3+1) we train on Peg2D, Key2D, Reach2D and test on tasks from the Stick2D family.

5.2.4 Expert Demonstrations

Every task in MetaSim has access to expert demonstrations. The latter can be extracted from any source such as a human user or a hard-coded optimal path. For this project we leverage SAC experts $\pi_E^{\mathcal{T}}$ to generate demonstrations for each of the 7 task families (convergence details on section 9.3.1 of the appendix). The difference with respect to the agent used in PERIL is that we provide the expert

with ground truth task descriptors $\hat{b}$ and train an individual expert for each task family. As we assume imperfect demonstrations, for every task $\mathcal{T} \in p(\mathcal{T})$ we roll out a set of $k = 3$ trajectories, each with its own randomised task alteration, and store these in their respective demonstration buffers $\mathcal{D}^{\mathcal{T}}$. We provide the agent with no more than 3 demonstrations per task to motivate meta-imitation learning methods to exploit transferable structures from within demonstrations of different tasks. Moreover, as a practical consideration, a limited set of demonstrations may be available in real world conditions, especially when defined by an expert user.

5.3 Ethical and Legal Considerations

Considering the project has been concurred via in-house simulated environments, there has been no direct influence with external sources such as other people, animals, or developing countries. Moreover, there have been no manipulation of dangerous equipment or material which could have harmed us. Notwithstanding, machine learning training via Google Cloud services does stress energy consumption through high-performance GPU usage. Indeed the field of machine learning is growing exponentially and, as part of the research community as well as the general community, we must be considerate when running massive-scale projects.

In addition, there has been no use of data provided by a human user. Even when leveraging expert demonstrations, we use external algorithms to produce these. Noteworthy is that in this project we exploit open-source code under the MIT license. For that reason, we have no copyright issues.

Another factor which must be considered is misuse. In effect, algorithms such as that proposed in this project are becoming more powerful as research in the fields of AI advance. In effect, these are typically envisioned for applications which contribute to society. Nevertheless, these can also be used, and in the worst case misused, in applications involving military or civilian use. All things considered, we believe that our project does not directly provide a means to these ends, and for that reason we are not ethically restrained on these matters. Our reason to believe this is that PERIL cannot discriminate as it is not equipped with tools to do so. In the evolutionary scope of general AI, current algorithms are but a mere speck of what general intelligence may become in the future. Indeed these will require further ethical consideration, but for now, we confidently select "No" in every item of the ethics checklist.

Chapter 6: Results & Evaluation

In this chapter we report a detailed discussion for the performance of our proposed method. We begin by performing in-depth analysis on the ML1 benchmark, where we demonstrate the key behavioural components behind PERIL and its derivatives. We also examine how probabilistic methods for hybrid meta-reinforcement and imitation learning compares with other baselines in terms of adaptation performance and training efficiency. To that end, we propose the following baselines:

- PEARL [51]. Baseline for probablistic meta-RL conditioned on exploratory data.
- MetaIL [70]. Conditioning PEARL from demonstrations to pre-condition exploration.
- PERIL-P. Our proposed method without conditioning on imperfect demonstrations.

Note we do not use other baselines as PEARL derivatives already demonstrate 20-100x more sample efficiency and substantially superior asymptotic results with respect to other baselines such as RL2 [38] or MAESN [39]. Moreover, in order to perform a fair evaluation, we use the same set of hyperparameters in both PEARL and MetaIL baselines as those implemented in our method (see appendix 9.2.1). Most of the research performed in meta-RL involves adaptation to a single task family [51, 38, 70].

In practice we seek for multi-use robots capable of interpolating from a diverse set of dynamics. To test the representational power behind PERIL we evaluate training and adaptation performance in multi-task meta-learning (MLX) settings and compare it to the aforementioned basilens. Consequently we evaluate our method on the complex multi-task transfer learning (MTL(X+Y)) benchmark. In the latter we examine if our method can generalise such that it can exploit long-term dependencies over a meta-trained set of dynamics to adapt to a different task family. Finally, we endure ablation studies to reflect the importance of each of the components which define PERIL, as well as other practical features.

6.1 Primal Analysis

In this section we perform a deep analysis on the adaptation behaviours of our approach. We introduce results for our first iteration and detail how we improve upon several faults observed during evaluation. For this specific task of empowering PERIL, we select the ML1 benchmark due to its simplicity. Recall in this benchmark, the agent must use context from both demonstrations and exploration to infer conditions in an unsupervised manner which allow it to be robust to changes in the geometries of the hole as well as its location. Note that in the following experiments we use the default settings for meta-training (section 9.2.1).

6.1.1 First Iteration

In the first iteration of PERIL (PERIL-P) we assume every demonstration is perfect, such that when given a demonstration of an unseen task, the exact same task conditions are observed during meta-test adaptation. In light of this, we halt any task alterations prior collection during meta-training. To evaluate the capacity of our method, we compare it to the aforementioned baselines. Training and adaptation performances are presented in Figure 6.1. Recall that during test time we perform adaptation at the trajectory level: after every posterior trajectory τ_{post} sampled online, this trajectory is aggregated to the context $c \leftarrow c \cap \tau_{post}$ and a new task belief $z \sim q_\phi(z|c)$ is sampled before rolling out the next trajectory. This process is performed for a maximum of 10 trajectories.

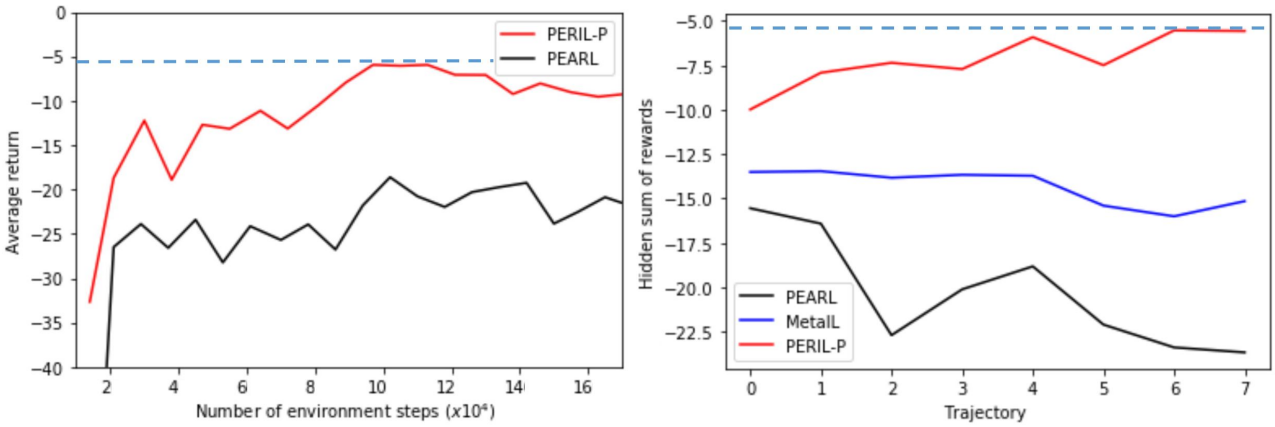

Figure 6.1: Test-time returns during meta-training (left) and test-time adaptation (right) via posterior sampling in ML1. Using a single demonstration, the agent can readily infer the task. We compare these results to adaptation using PEARL. Blue dashed line indicates the average return from the expert trajectories.

Training efficiencies on ML1 are substantially higher than PEARL. Taking into account that the latter is considered to be one of the most sample efficient meta-RL algorithms [51], these results line up with our motivation for providing hybrid learning abilities: by leveraging hybrid meta-IL on top of meta-RL we substantially improve sample efficiency. This is partly because behavioural cloning motivates the agent to select expert actions which typically lead to higher rewards. Another reason is that in PERIL-P, we pre-train the task encoder using expert demonstrations and auxiliary modules. Pre-conditioning the task encoder prior collection of data reduces the high initial variance on bootstrapped estimates of z: since task beliefs during the first episodes are noisy, inherent high-variance problems commonly observed during actor-critic training are exacerbated when these are conditioned by another layer of stochasticity.

Adaptation results demonstrate the powerful inference mechanism behind our method. Notice how PERIL adapts to unseen tasks without any additional exploration, whereas state-of-the-art methods such as PEARL fail to do so. Even when conditioned with the demonstrations, MetaIL struggles to extract sufficient information to discern the unseen task. In contrast, our method is meta-trained to do so via mutual information maximisation. Figure 6.2 illustrates sample trajectories generated by each one of the aforementioned baselines and a summary of the adaptation results is presented in Table 6.1, where

success rate is defined as the number of tasks which, after 10 trajectories, result in task completion.

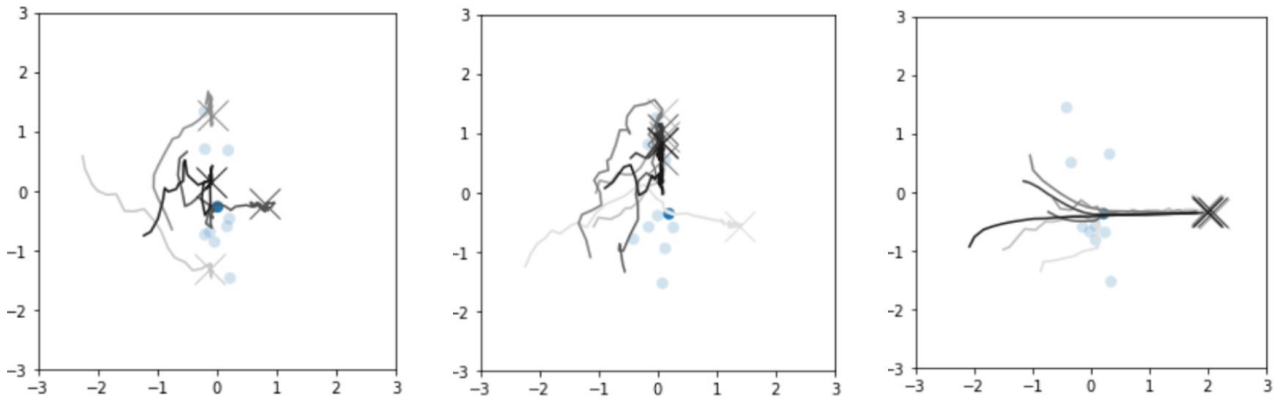

Figure 6.2: Sampled trajectories from MetaIL, PEARL and PERIL-P (left to right) for an unseen task. Horizontal and vertical axes represent the normalised observation space in directions x and y respectively. Trajectory shading (light grey to black) darkens as the trajectory index increases. Light blue circles represent the entry point of all the unseen tasks evaluated in ML1, whilst the dark blue circle represents the clearance hole entry point for that specific task.

Table 6.1: Evaluation performance metrics. In the ML1 setting, PERIL-P shows superior adaptation to unseen tasks both in adaptation speed and performance and is capable of zero-shot learning.

Model	Success Rate (%)	Min K-Shot	Mean K-Shot	Max K-Shot
PEARL	10	5	8.3	10
MetaIL	30	3	5.6	10
PERIL-P	**100**	**0**	**1.7**	**5**

The first iteration of our proposed method presents itself as a robust alternative to state-of-the-art methods. In all metrics PERIL-P surpasses other baselines, reaching one-shot and in some cases, zero-shot learning with 100% success rate under 5 adaptation trajectories. Notwithstanding, crucial to the robustness of this approach, we must evaluate its online adaptation performance in tasks which may otherwise be ill-defined from a set of demonstrations. To evaluate this we provide random alterations to the task after being given a demonstration. An illustration of the trajectories generated in this setup is presented in Figure 6.3.

During the first adaptation trajectories, where the agent acts stochastically, the agent relies too heavily on the demonstration. Consequently, this impedes it from exploring in a structured manner and thus disambiguating the task. Looking deeper into this issue, we study the distribution of beliefs sampled from the ground truth auxiliary module $b \sim b_\lambda(b|z), z \sim q_\phi(z|c)$ (Figure 6.4). Since the uncertainty in z directly affects the uncertainty in b, we use this as a proxy to z as a means to conceptualise how PERIL-P reasons about the new task.

As foreseeable by the trajectories generated, the task belief deviates from its ground truth in cases where alterations are too extreme. Instead of using posterior experiences to mitigate this, we postulate that

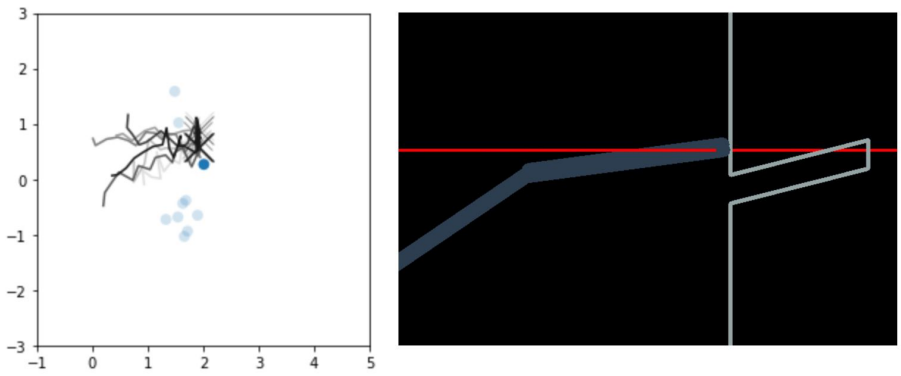

Figure 6.3: Trajectory distribution (left) and task setup (right) of a PERIL-P agent in a randomised altered ML1 task. The red horizontal line marks the position of the goal from the demonstration. In this example, the alteration is too large for PERIL-P to adapt to the imperfect demonstration.

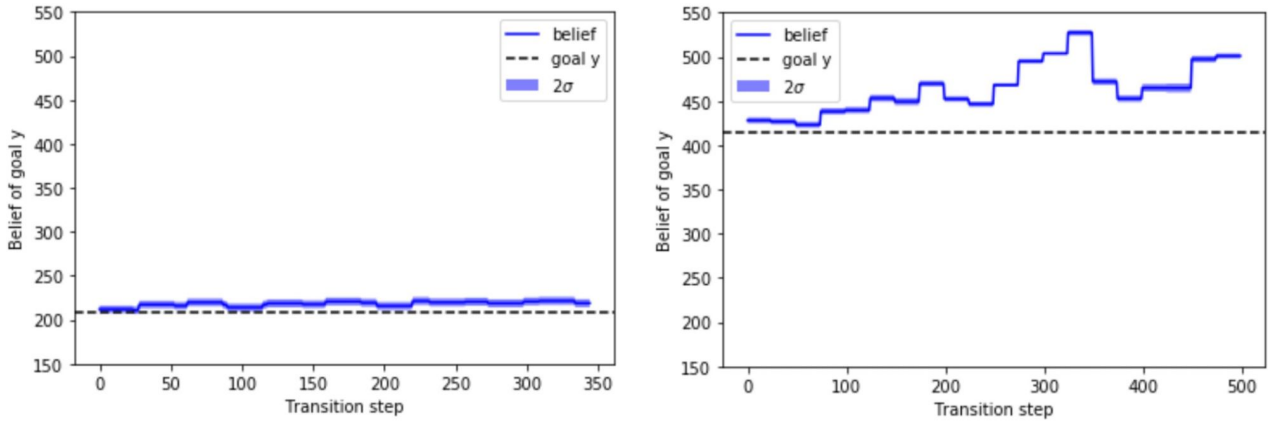

Figure 6.4: Belief for the task descriptor b_y during adaptation steps for the case of: (left) an unaltered task; (right) a heavily altered task.

PERIL-P has learnt to solely depend on demonstrations. In effect, this is exactly what it has been trained to do: if provided with a perfect demonstration and the goal is to maximise rewards, the optimal inference strategy is to produce deterministic task beliefs before rolling out an exploration policy. The reason for this is that posterior exploration is not necessary once the agent learns how to extract the right information from a demonstration. In light of these observations we propose several improvements to PERIL-P, with the aim of inciting the agent to (i) reason about the inherent uncertainty of a demonstration; (ii) balance task inference such that context collected during posterior sampling conditions exploration to a greater extent. Note that the latter objectives are only strictly required in adaptation environments where uncertainty is naturally present.

6.1.2 A Robust Alternative

By adding imperfect demonstrations in the meta-training loop we force the agent to reason about the potential uncertainty behind a demonstration. In essence, we hope that the agent learns to extract task descriptors of a demonstration such that it can rely further on the context collected during posterior sampling. With this, we propose a method which is highly robust to imperfect unseen tasks. Training and test-time adaptation performances in this setup are presented in Figure 6.5.

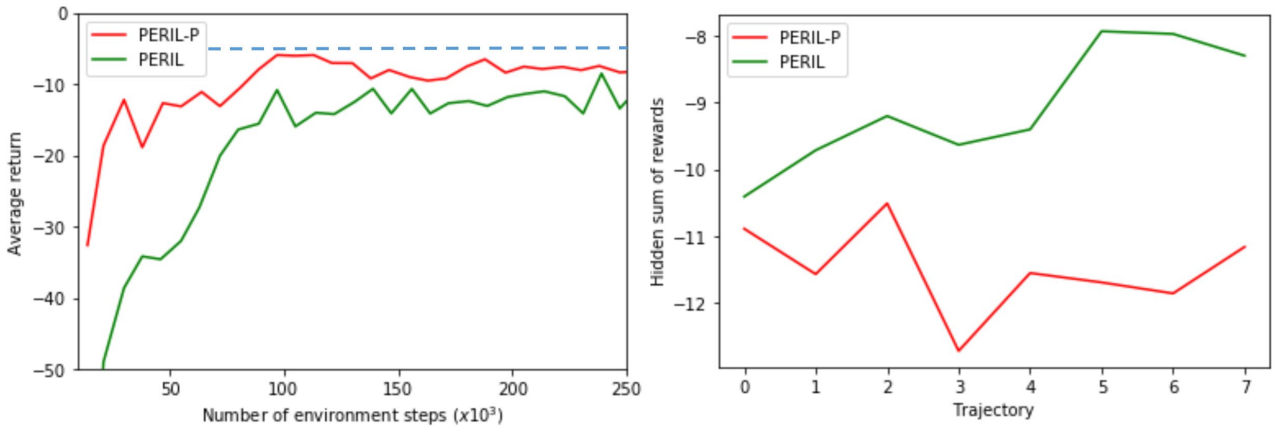

Figure 6.5: Test-time returns during meta-training (left) and test-time adaptation (right) via posterior sampling in ML1. PERIL demonstrates significantly superior adaptation performance to altered tasks. Blue dashed line indicates the average return from the expert trajectories.

Given the added stochasticity in the demonstrations which require additional reasoning, it is natural that PERIL is more sample inefficient than PERIL-P, taking around 2-3$\times$ more samples to reach average returns above -10. Moreover, PERIL shows a droop in meta-training evaluation returns. This observation is directly linked with the fact that it must explore the task further given the incomplete task belief z recovered from the imperfect demonstration. At the expense of increased sampled inefficiency, adaptation performance in imperfect conditions is improved significantly. We present metrics for the evaluation of both PERIL and PERIL-P in Table 6.2.

Table 6.2: Evaluation performance metrics. In the ML1 setting, PERIL-P shows superior adaptation to unseen tasks both in adaptation speed and performance.

Model	Alterations	Success Rate (%)	Min K-Shot	Mean K-Shot	Max K-Shot
PERIL-P	No	**100**	**0**	**1.7**	**5**
PERIL	No	90	0	2.4	6
PERIL-P	Yes	80	0	4.4	∞
PERIL	Yes	**100**	**0**	**3.1**	**6**

To discern the quality of exploration behind PERIL and PERIL-P, we use kernel density plots to extract observational space coverage during the first set of trajectories. As illustrated in Figure 6.6, trajectories generated by PERIL result in significantly narrower and more concentrated distributions in the observational space.

In cases where the alteration is low, PERIL-P consequently provides faster adaptation. However, in situations where task alterations are high, overconfidence of the task belief leads to poor exploration. On the other hand, PERIL balances exploration and exploitation by reasoning about the uncertainty of the task. In conclusion, increased entropy in the collected context aids the task encoder to become more certain about the task. We further support this argument by monitoring the progression of the task belief uncertainty during adaptation (Figure 6.7).

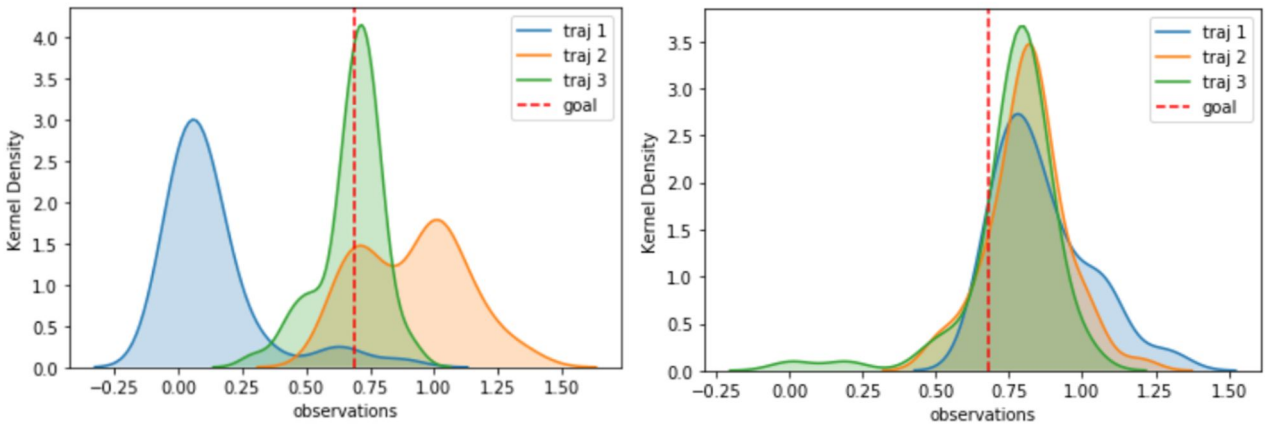

Figure 6.6: Meta-training (left) and test-time adaptation (right) via posterior sampling in ML1. Through adaptation in the imperfect setting, PERIL demonstrates higher adaptation performances in contrast to PERIL-P.

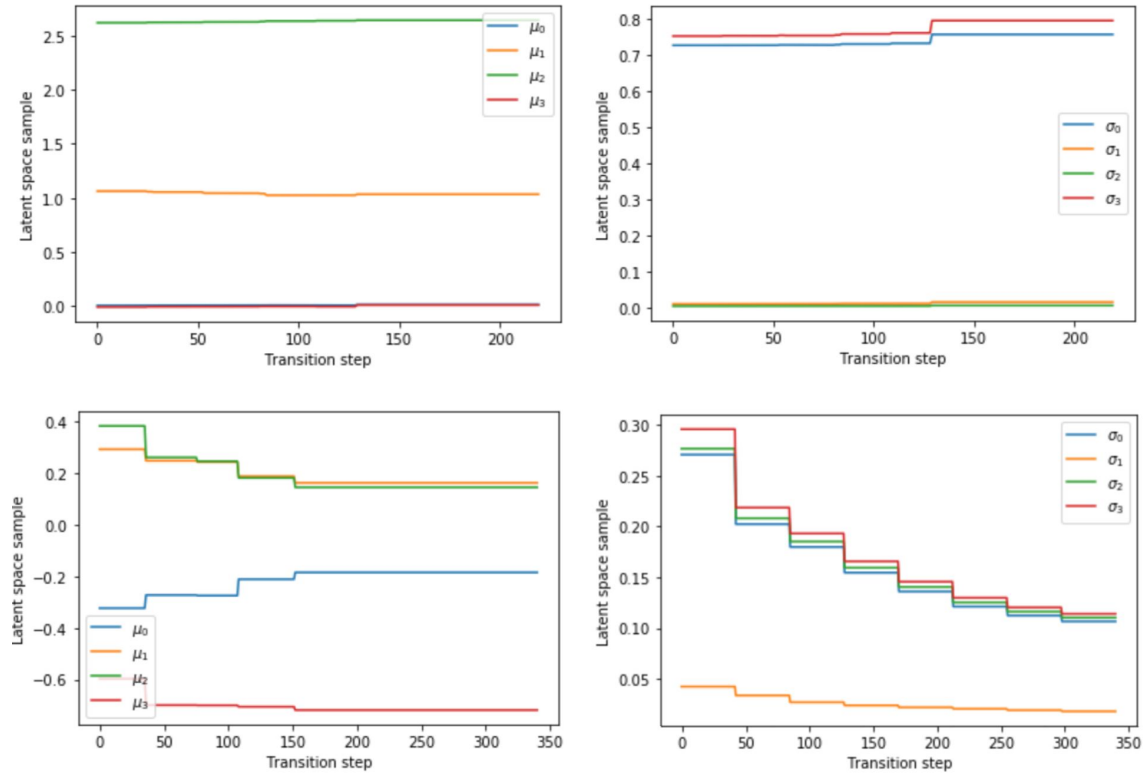

Figure 6.7: Mean μ (left) and standard deviation σ (right) components for 4 components of the latent context vector z during adaptation of an unseen task. PERIL-P (top) shows little to no dependency on the context collected during exploration. In contrast, PERIL (bottom) demonstrates greater dependency.

As the context is fed with additional contextons retrieved from the environment, PERIL adapts its belief z both by altering its mean distribution $z\mu$ and reducing its variation $z\sigma$. With these results we show how, by reducing dependency on demonstrations, the agent learns to reason about its uncertainty and exploit the context collected via posterior sampling in order to condition its exploration strategy. At the cost of reduced expected rewards in cases where the random alterations are small, this behaviour achieves maximum expected rewards in the long term.

6.2 Advanced Analysis

In this section we assess the performance of PERIL in complex multi-task and multi-task transfer meta-learning settings. We compare PERIL to PEARL and PERIL-P baselines during meta-training for all benchmarks but we only provide results for test-time adaptation for PEARL and MetaIL in the ML4 environment. The reason for this is that these methods have already demonstrated considerably inferior adaptation performances in unseen tasks. As we will show in section 6.2.1, these are also incapable of enduring robust adaptation to different dynamics in multi-task settings.

In the following studies we set a limit of approximately 500K collection samples. The reason behind this is that, although most training can be performed during simulation, data collection in real-time robotic systems is typically one of the biggest caveats of using reinforcement learning in the latter.

6.2.1 Multi-Tasking

Multi-tasking is a valuable skill which can be paramount in many machine learning systems. Specifically, in the application of robotics, multi-tasking can allow a single robot to perform multiple tasks without re-programming. This is particularly useful in situations where a robot may need to switch tasks frequently. We present results for two benchmarks: ML4 and ML5. Each benchmark has a different repertoire of task families which test different components of adaptation.

Note that during test time the agent is randomly provided with a demonstration of any of the task families belonging to that benchmark. The agent must not only discern the position and geometrical aspects of the goal as in ML1, but it must also realise to which task family it is trying to adapt. We leverage meta-training to interpolate from within (i) the distribution of demonstrations from different *and* identical task families; (ii) components of the dynamics between each task family which can be shared. By exploiting the latter, PERIL must be robust to unseen tasks from within each task family.

ML4

Training performances for each specific task family in ML4 are presented in Figure 6.8. We quickly observe that PERIL and PERIL-P demonstrate similar stable convergence behaviours in all 4 task families. In contrast, PEARL fails to provide any indications of meta-learning under the multi-task setting. These results are indicative that our method is capable of interpolating from within different sets of learnt dynamics and sparse reward feedback.

Averaged trajectory-based adaptation for all task families is reported in Figure 6.9. From these, an exhaustive summary of the performance metrics for all baselines in ML4 is presented in Table 6.3. Although mildly, we observe higher success rates in PERIL than in PERIL-P as well as lower adaptation upper bounds (max k-shot). On the other hand, PERIL-P seems to adapt to unseen tasks faster than PERIL. Overall, differences between one and the other are minimal, suggesting that PERIL-P has improved upon its ability to reason about the uncertainty of a demonstration under multi-task settings, even when leveraging perfect demonstrations. This phenomenon can be explained by the fact that, given

the diverse distribution of expert trajectories observed in each task family, task inference is intrinsically more convoluted. For that matter, PERIL-P cannot fully recover the ground truth descriptor from a single demonstration, thus, learns to leverage its belief on additional context collected from the environment.

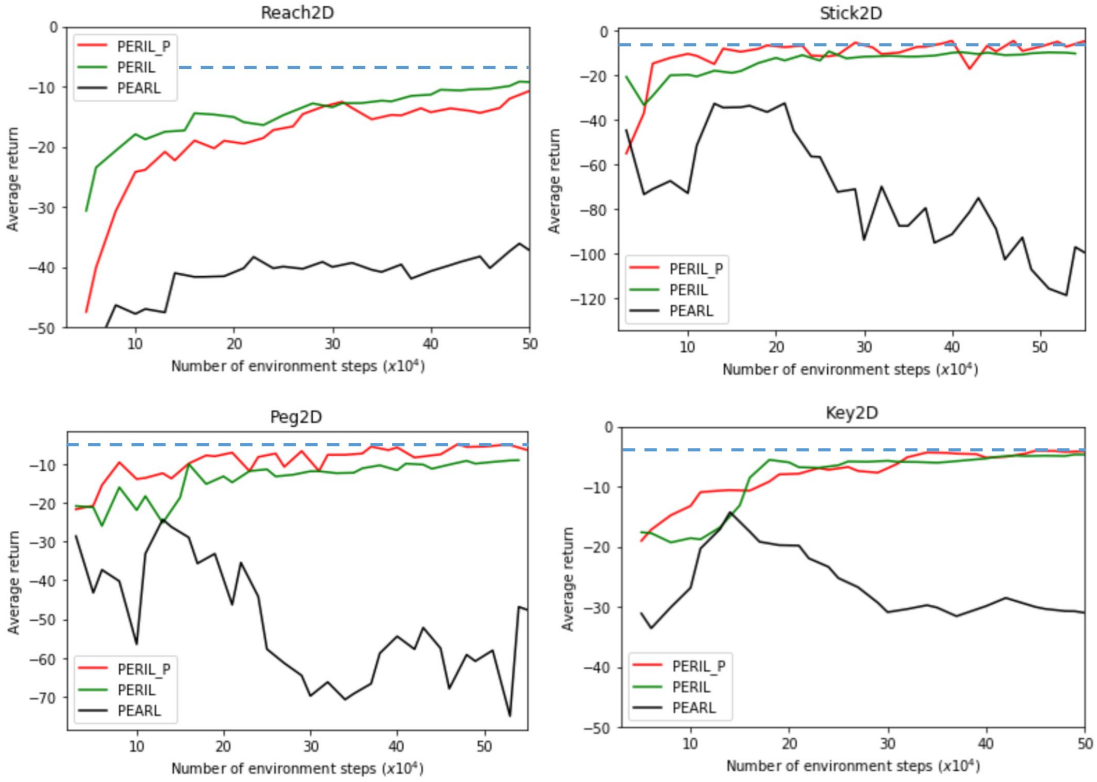

Figure 6.8: Test-task performance vs. number of samples collected during meta-training. PERIL and PERIL-P consistently show superior meta-learning capabilities with respect to PEARL in all 4 task families in ML4. Blue dashed line indicates the average return from the expert trajectories.

ML4 is specifically designed to test multi-tasking over task families which cover similar observational spaces. For that matter it is key to understand if the agent is capable of learning the differences from one task family to another or if it simply executing a policy which can solve all of these task families. Effectively, we wish to understand how the task belief distribution $p(z)$ differs when encountered with different task families. To do so we cluster the task belief generated from the context collected during each one of the 10 exploratory trajectories.

Formally, the task-specific context latent space $z^{\mathcal{T}_f}$ is sampled from a randomly selected task $\mathcal{T}_f \sim p(\mathcal{T})$ over each task family f and concatenated into a single vector $z_{agg} = [z^{\mathcal{T}_f} \sim q_\phi(z|c^{\mathcal{T}_f}_{1:10})], f \in [0, ..., 3]$. From z_{agg} we produce a T-Distributed Stochastic Neighbouring Entities (TSNE) plot (Figure 6.10) which performs clustering based on similar principles than that of principle component analysis (PCA). Results for the latter are also presented.

One distinct vector sub-space per task family is defined by each of the clusters in both TSNE and PCA plots. Notice how task families which are related to one another (Peg2D, Stick2D) produce task beliefs which are statistical similar. In contrast, uncorrelated task families (Key2D, Reach2D) produce task

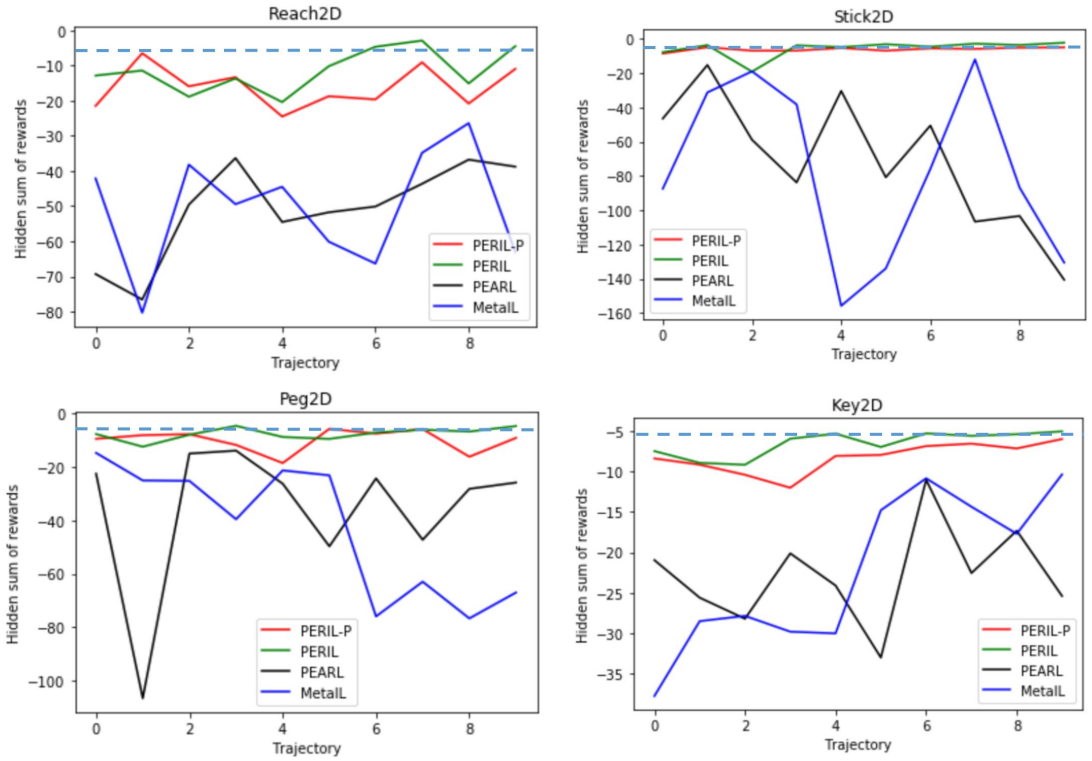

Figure 6.9: Average adaptation performance (along 10 unseen tasks per task family) for PERIL, PERIL-P, PEARL and MetaIL-based agents on the ML4 benchmark. Blue dashed line indicates the average return from the expert trajectories.

Table 6.3: Evaluation performance metrics. In the ML4 setting, PERIL and PERIL-P shows similar adaptation performances to unseen tasks. In contrast, PEARL and MetaIL fail to give indications of meta-learning.

Task Family	Model	Success Rate (%)	Min K-Shot	Mean K-Shot	Max K-Shot
Peg2D	PERIL	**100**	0	1.9	**5**
	PERIL-P	90	0	**1.5**	7
	PEARL	0	8	∞	∞
	MetaIL	10	7	∞	∞
Reach2D	PERIL	70	0	4.1	**6**
	PERIL-P	**80**	0	**3.6**	9
	PEARL	10	∞	∞	∞
	MetaIL	20	6	∞	∞
Stick2D	PERIL	70	0	**4.4**	**8**
	PERIL-P	70	0	4.7	9
	PEARL	0	∞	∞	∞
	MetaIL	0	∞	∞	∞
Key2D	PERIL	100	0	**2.9**	**5**
	PERIL-P	100	0	3.6	6
	PEARL	0	∞	∞	∞
	MetaIL	10	4	∞	∞

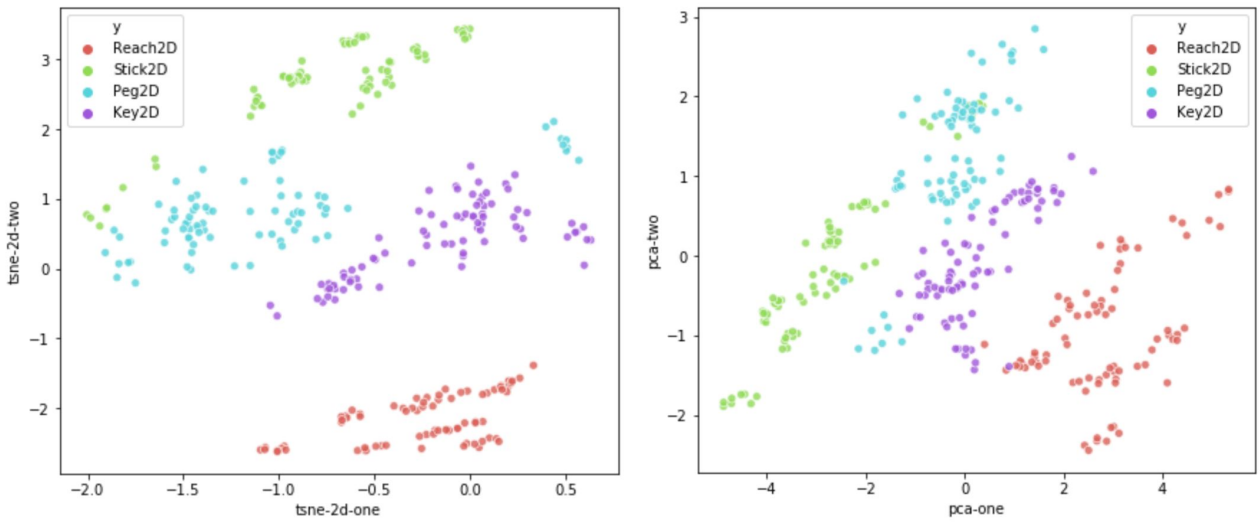

Figure 6.10: TSNE (left) and PCA (right) plots on the latent context variable z sampled for each of the 4 task families in ML4. TSNE results demonstrate that PERIL has meta-learnt to produce task descriptors which are clearly distinguishable from one task family to another. Additionally PCA shows smooth transition within the clustered vector spaces.

beliefs which are mapped to opposite sides of the spectrum. The relatively low inter-cluster distances (distance from one family cluster to its nearest neighbour's cluster) observed in the PCA plot suggests that distributional shift during adaptation is low: since the vector space for z is dense yet clearly defined a set of clusters, generated task beliefs are likely to condition the policy to solve for a unique task family.

The most important conclusion we can derive from these results is that by learning how to map the context collected from a set of task families, PERIL has constructed powerful inference mechanisms which are able to reason. This remark is paramount to resolving the nature of PERIL's task inference capabilities. Instead of learning a single latent context z which solves for all tasks, it learns how to adapt z depending on the conditions it observes such that it can solve that specific task. The latter is a far superior alternative as it suggests that PERIL can be used to interpolate from within any combination of task dynamics conditioned on the 4 vector sub-spaces.

Finally, by demonstrating that z clusters for each task family are distinguishable, we show that PERIL has not only relied on the auxiliary loss to optimise its task encoder, but rather in representing task descriptors which can help discern any task from any task family. In the contrary, Peg2D, Stick2D and Key2D clusters would lie in highly overlapping regimes of the vector space defined in PCA.

ML5

The ML5 benchmark is devised to test adaptation in situations where the agent must perform a task with similar dynamics in orthogonal observational spaces. Flexibility to changes in orientation may become a useful attribute in robotic applications derived from industrial lines, where the robot must adapt to perform a simple task such as spot welding. By giving it a single demonstration, the agent can subsequently adapt to any specific location in space.

Meta-training performances are reported in Figure 6.11, where the latter validate our method's ability to support multi-tasking along different observational spaces. In this set-up PERIL-P shows a clearer predominance with respect to PERIL than in ML4. In light of this, we postulate that the added uncertainty which involves incorporating imperfect demonstrations is not necessary in complex multi-task settings as reasoning between different task families reduces the amount of reliance on demonstrations. Hence, given the sampling limitations, we claim that PERIL-P is a more powerful and sample efficient algorithm in complex multi-task settings.

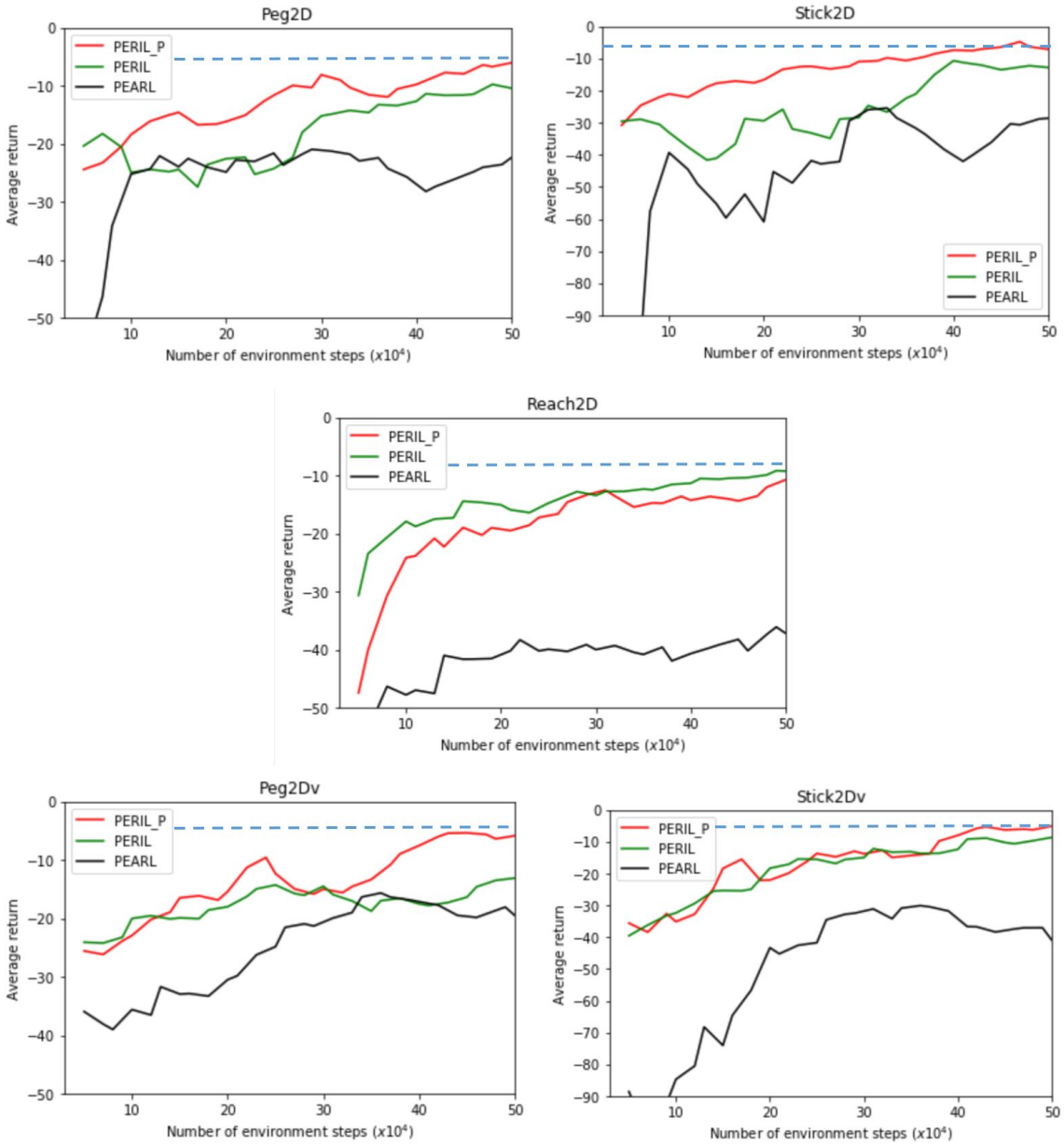

Figure 6.11: Test-task performance vs. number of samples collected during meta-training. PERIL-P shows superior overall performance in ML5. Blue dashed line indicates the average return from the expert trajectories.

As in ML4, we study the distribution of the latent space z collected adaptation to unseen tasks (Figure 6.12). In this case, we leverage PERIL-P to collect trajectories. There are 3 clear clusters which correspond to vertical (Stick2D, Peg2D), horizontal (Stick2D, Peg2D), and Reach2D task families.

These clusters provide evidence that the task encoder is capable of reasoning about the orientation of the macro policy. Furthermore, as illustrated in the PCA distribution, PERIL is also capable of discriminating between tasks which are governed by the dynamics of Stick2D or Peg2D by clearly separating their vector sub-spaces. Metrics for the per-trajectory adaptation performance of PERIL-P under the ML5 benchmark are presented in Table 6.4.

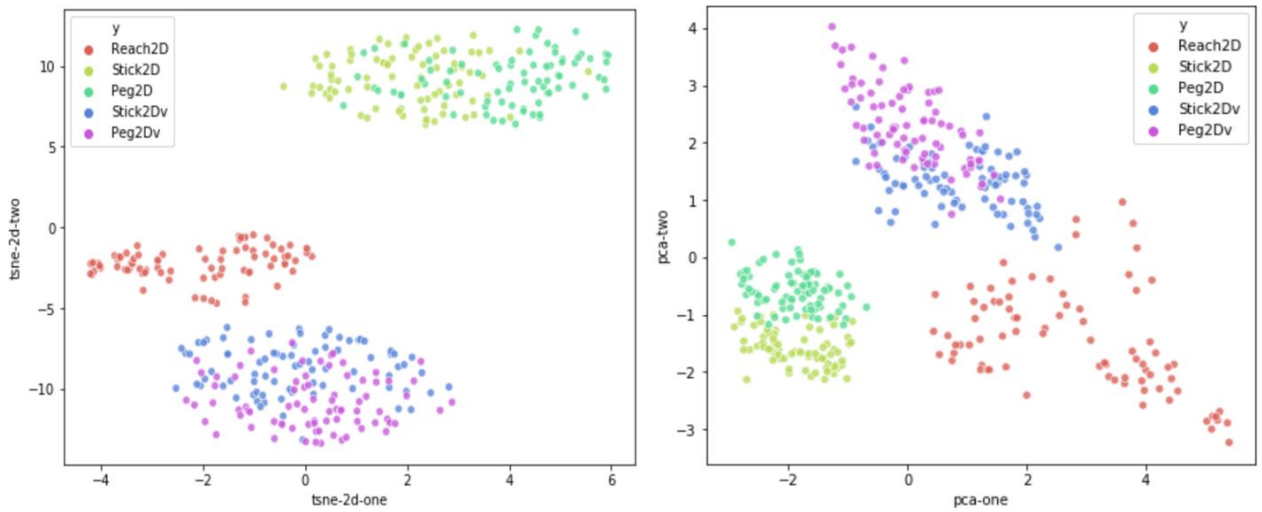

Figure 6.12: TSNE (left) and PCA (right) plots on the latent context variable z sampled for each of the 5 task families in ML5.

Table 6.4: Test-time adaptation performance metrics for PERIL-P in ML5.

Task Family	Success Rate (%)	Min K-Shot	Mean K-Shot	Max K-Shot
Peg2D	80	0	4.1	10+
vPeg2D	80	0	4.9	10+
Reach2D	70	0	5.0	10+
Stick2D	70	0	4.5	10+
vStick2D	60	0	6.2	10+

Adaptation performance is considerably lower than that observed in ML4, where success rates for vertical Stick2D drop down to 60%. However, the variation between one task family and the other is substantially greater in ML5 which accounts for this droop in performance. In practice, this could be mitigated by increasing the representational power of the task encoder as well as the actor-critic networks (adding hidden layers, increasing hidden layer populations, expanding latent space dimension, *etc.*), or simply by allowing more meta-training by extending the 500k sampling limit.

6.2.2 Transfer Meta-Learning

MTL benchmarks require the ability to leverage long-term memory over a different set of behaviours in order to perform a task of unknown dynamics. In essence, we expose the agent to a task from an unseen task family and evaluate if it is capable of providing task beliefs which are capable of conditioning the policy to successfully perform said task. In other words, this benchmark assesses the ability for learning to learn under a wider, more general, scope.

MTL(3+1)

Since we are specifically interest in PERIL's ability to reason about the unseen task, we begin the analysis by directly assessing the vector space defined by the principal components of the task belief generated during adaptation in MTL(3+1). As displayed in Figure 6.13, all tasks in training task families (Peg2D, Reach2D and Key2D) are easily distinguished by the task encoder.

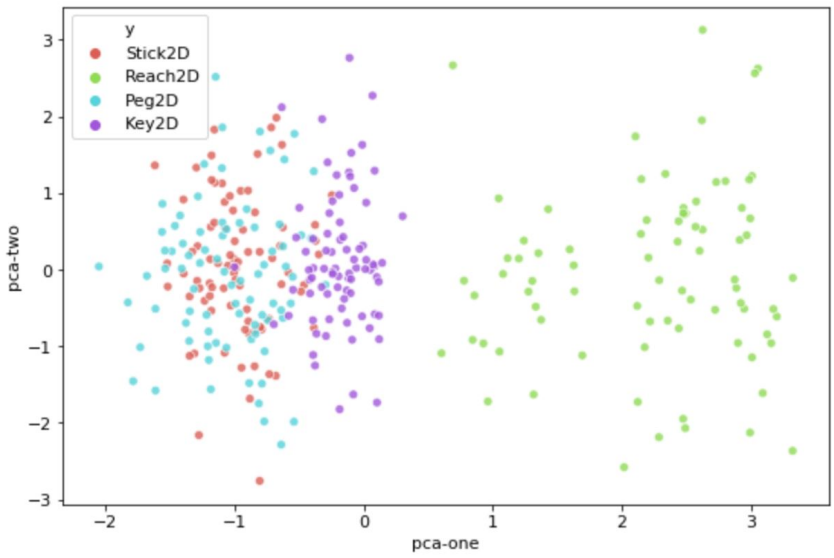

Figure 6.13: PCA on the principal two components of the latent context variable z distribution sampled for all task within each of the 3 training task families (X) in MTL(X + Y) as well as the unseen task family (Y) Stick2D.

A key phenomena observed during adaptation to the untrained task family Stick2D is that the agent interpolates between Key2D and Peg2D dynamics to produce a task belief, rather than inferring task beliefs outside its distribution. In terms of statistical robustness, this suggests that covariate shift during adaptation is controlled, something which is inherently difficult to achieve in the validation of ML systems outside their training data distribution.

Intuitively, optimal policies for Stick2D and Peg2D are somewhat similar. For that matter, by producing beliefs within the Peg2D cluster gives a prior indication that the agent is capable of adapting to Stick2D tasks. We further validate these claims by evaluating per-trajectory adaptation performances of PERIL under the MTL(3+1) benchmark (Figure 6.14). Notice how (i) Multi-task adaptation is highly stable in the ML3 setting; (ii) Out-of-task inference for Stick2D produces policies which can perfectly adapt via few-shot learning to tasks from an unseen task family.

The results are indicative that PERIL can become a powerful tool capable of using past experiences to solve a wide range of tasks. By conceptually demonstrating its ability to escape its training distribution we provide a framework which can generalise to different conditions.

Inspired by Quality Diversity [76], we suggest a potential approach which could scale PERIL to more complex and diverse multi-task settings. By setting up a hierarchical system, similar to the hierarchical behavioural repertoire (HBR) presented in [76], we can store the vector space which describes the different

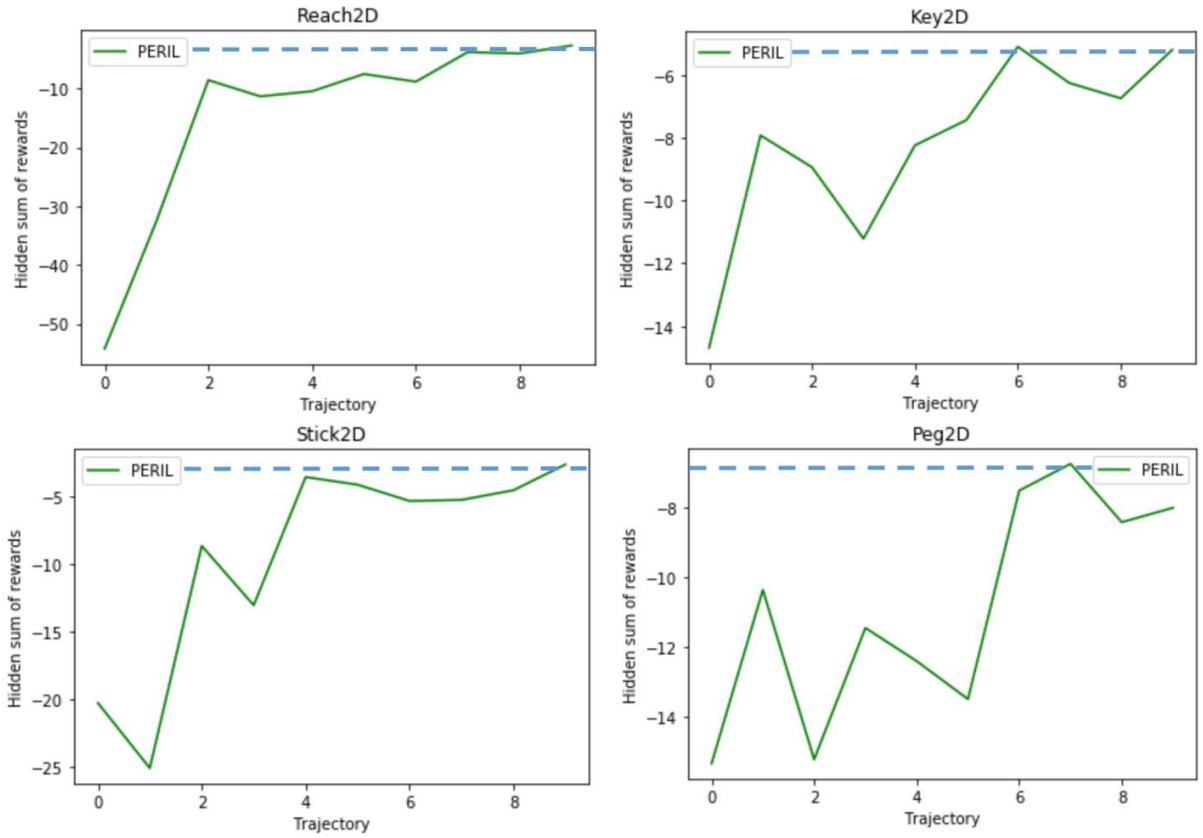

Figure 6.14: Average adaptation performance (along 10 unseen tasks per task family) for PERIL agents on the MTL(3+1) benchmark. Blue dashed line indicates the average return from the expert trajectories. PERIL demonstrates robust adaptation to multi-tasking (Peg2D, Reach2D, Key2D) as well as meta-transfer learning along tasks from the Stick2D task family.

clusters produced from the task encoder during meta-training. Consequently, a controller could be optimised, via evolutionary approaches or otherwise, to select the so-called behavioural descriptors from the z space such that it learns how to switch from one macro policy (behaviour) to another. Note that, as demonstrated in the trajectory adaptation performances, by altering z we condition the meta-trained policy to perform different tasks. In essence, this could result in a system which can rapidly switch from one behaviour to another.

6.2.3 Continuous Adaptation & Limitations

Continuous adaptation (infinite horizon setting) is an attractive alternative to trajectory-based adaptation (multiple trajectories of finite horizon). This is especially true in real world conditions, where having a robot which does not require episodic resets is substantially beneficial both in terms of sample efficiency and practical reasons (re-setting the robotic manipulator may involve external controllers and visual perception tools).

Notwithstanding, continuous adaptation presents with additional challenges. In particular, if the agent falls into a local minima during exploration, it may become particularly difficult to escape said minima without a re-start. In that case, the context would be continuously fed with similar contextons which do not help the task encoder recover the ground truth descriptor of the task. This results in an agent which

is incapable of altering its policy, leading to cyclical behaviours. Notice that this is not generally the case in trajectory-based adaptation as the agent can re-start from its initial state distribution.

Continuous adaptation is especially complex in our case since task inference is performed via sparse rewards. By visualising trajectories and their respective observational KDE plots, we find that we do observe the aforementioned local minimas in some cases during adaptation under the infinite horizon setting (Figure 6.15). The key point to take from these results is that the observational entropy collected during exploration is significantly lower via continuous adaptation.

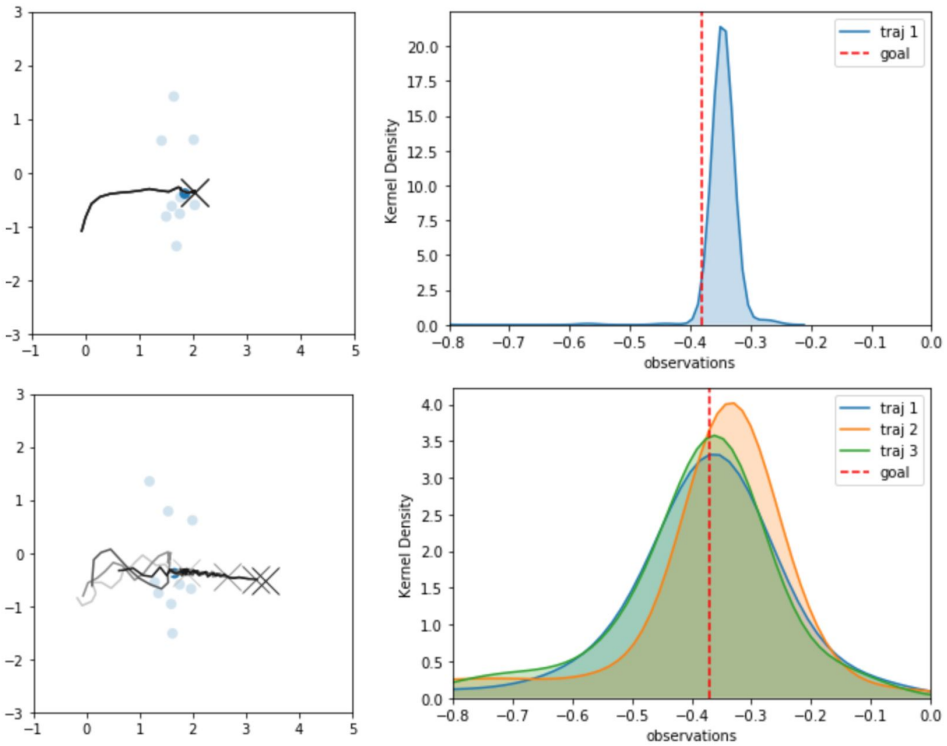

Figure 6.15: Trajectory adaptation in ML1. for continuous (top) and trajectory-based adaptation (bottom). In this specific unseen task, continuous adaptation forces the agent to get stuck at a local minima. Observational coverage (right) demonstrates how continuous adaptation collects context with lower entropy than in trajectory-based adaptation.

Exploration in PERIL is performed via the uncertainty in the task belief z as well as the stochasticity in the soft actor. When performing tasks, humans use short-term memory to help them reason about their exploration policy. For instance, when attempting the task of switching the lights whilst blindfolded, we remember where we have searched such that we do not search again. Inspired by this analogy, we propose that in order to successfully adapt continuously, we would require a third contribution to the stochasticity of PERIL. Said contribution would aim at exploiting recently collected transitions to condition exploration.

Meta-RL approaches conditioned on learning via memory-based architectures [37, 21] have been deprecated due to their instability. However, by decoupling exploration from exploitation as suggested in [63] we suggest a promising solution to continuous adaptation which leverages both long-term (meta-training behaviours stored in the actor and the task encoder) and short-term memory contributions. This hy-

pothetical solution consists in training an RNN-based exploration policy with the objective of collecting transitions which affect inference of the task belief. In other words, an RL agent which is rewarded on information gain between z and the currently collected transition. On the other hand, the exploitation policy could be trained as in PERIL. Although tentative, we leave this for future works.

6.3 Ablation Study

In this section we perform ablation studies on some of the features which define PERIL to understand how these affect our approach. For simplicity, we train each agent in this section on the ML1 benchmark.

6.3.1 Auxiliary Module

In PERIL we have exploited the availability of privileged information during meta-training. In essence, for each task we used a ground truth task descriptor $\hat{b}$ with the aim of providing stability during meta-training. In section 6.2.1 we prove that $\mathcal{L}_{aux}$ is not the only component which optimises the task encoder by demonstrating that the latent space representation z generated in unseen tasks from diverse task families are mapped into different clusters. Moreover, in 6.2.1 we demonstrate that, by projecting a task family into an orthogonal space, tasks of different dynamics cluster close (yet clearly separated) to one another based on their orientation. These results are indicative that PERIL conditions z on information which captures both the position of the goal and the dynamics of the task. In this study we ablate the auxiliary module $b_\lambda(b|z)$ from PERIL's framework to evaluate its contribution towards convergence in meta-training. The results of the latter are reported in Figure 6.16.

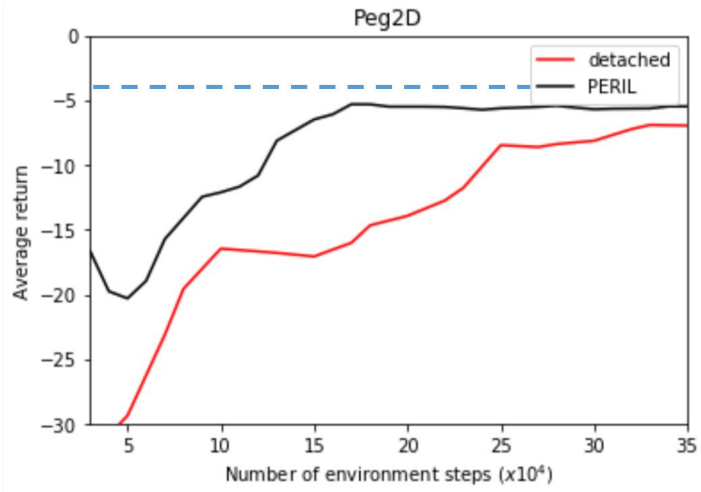

Figure 6.16: Test-task performance vs. number of samples collected during meta-training. By detaching the auxiliary module, sample efficiencies are considerably inferior ML1. Blue dashed line indicates the average return from the expert trajectories.

Removing the auxiliary module from the meta-objective reduces the sample efficiency of PERIL, but does not necessarily limit its capacity in the long term. This behaviour is expected considering that $\mathcal{L}_{aux}$ is a stable loss function which is decoupled from RL. In conclusion, the results suggest that incorporating auxiliary modules is not necessary. However, if privileged information is available during meta-training, these are highly recommended.

6.3.2 Mutual Information Dependency

In our approach we leverage mutual information $\mathcal{I}(z;\tau_E)$ between the trajectories generated by the demonstrations $\tau_E \sim \mathcal{D}$ and the latent context variable z, as means of linking z into PERIL's meta-IL learning component. As demonstrated in section 4.2.3, $\mathcal{I}(z;\tau_E)$ decomposes into a BC component $\mathcal{L}_{bc}$ and an information regularisation component $\mathcal{L}_{info}$. Although the effect of $\mathcal{L}_{bc}$ can be easily conceptualised, the contribution of $\mathcal{L}_{info}$ is somewhat more complex to pinpoint. Recall that in the derivation of the latter, we argue that it serves as a regulariser which aims to keep the latent space inferred from the collected trajectories similar to that extracted from the expert demonstrations.

In this study we show how this term can help improve adaptability at test time via regularisation. Figure 6.17 shows that by having a strong contribution of $\mathcal{L}_{info}$ (weighting factor of 1), PERIL struggles to converge. On the other hand of the spectrum, excluding $\mathcal{L}_{info}$ or including it as in PERIL (weighting factor of 0.2) leads to stable asymptotic convergence. These results support our theory that $\mathcal{L}_{info}$ acts as a regulariser. However, meta-training results do not indicate clear differences between including $\mathcal{L}_{info}$ or completely ablating it.

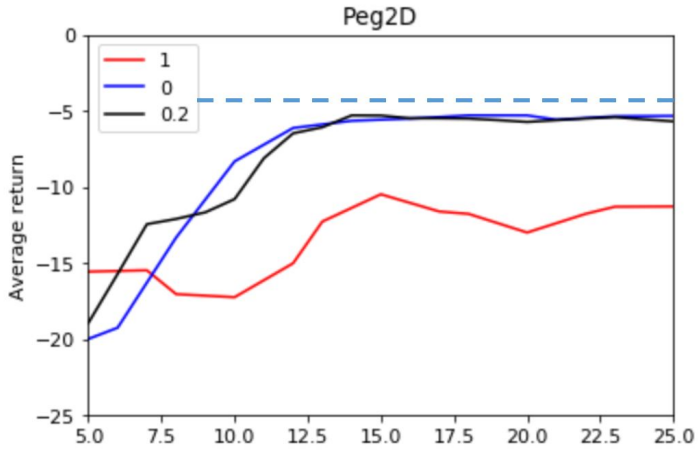

Figure 6.17: Test-task performance vs. number of samples collected during meta-training. By setting large weightings on $\mathcal{L}_{info}$, considerable performance droops are observed. Blue dashed line indicates the average return from the expert trajectories.

Assuming $\mathcal{L}_{info}$ does behave as a regulariser, it can be expected that including it does not improve training performances. Nevertheless, since $\mathcal{L}_{info}$ regularises inference over z, we expect the latter to be described by a simpler, well defined distribution during adaptation. We test this by performing PCA on the context latent spaces generated during adaptation to unseen tasks to in ML1. (Figure 6.18).

Indeed, including $\mathcal{L}_{info}$ regularisation induces additional structure on the latent space generated during adaptation. Naturally, this produces agents which are more robust to unexpected behaviours. Moreover, in order to impart generalisation and flexibility, having a well defined cluster of latent spaces per task family is paramount in multi-task and meta-transfer learning settings. In light of these observations, we claim that $\mathcal{L}_{info}$ is a highly recommendable regularisation component for meta-IL in multi-task and multi-task transfer meta-learning.

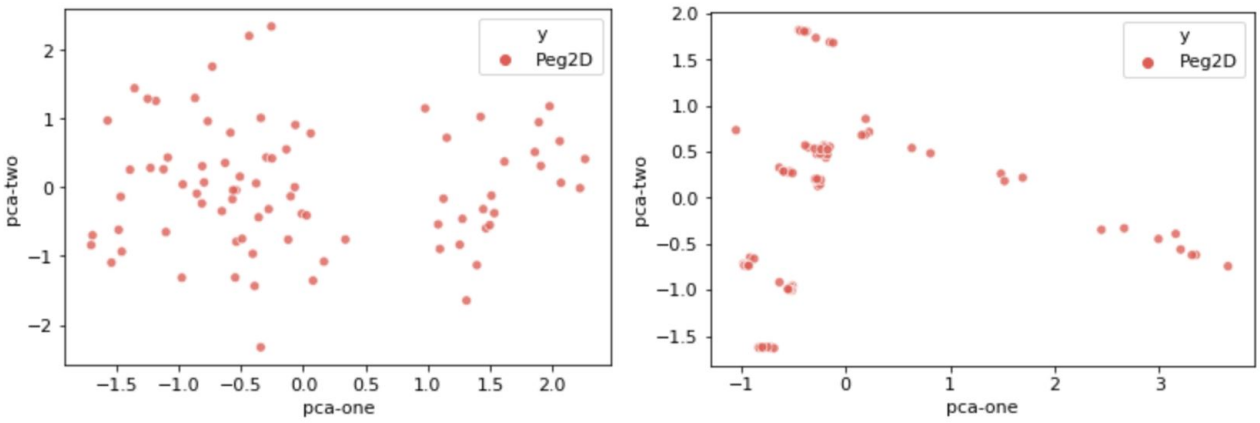

Figure 6.18: Principal components (PCA) of the distribution of latent context variable z sampled during adaptation to unseen ML1 tasks. On the left we use PERIL (weighting of 0.2) and on the right we ablate $\mathcal{L}_{info}$ (weighting of 0). We observe deficiencies in the structural integrity of the distributions of z in the latter condition.

6.3.3 Task encoder

In contrast to state-of-the-art methods for meta-RL such as PEARL [51], we leverage latent GNNs to parametrise our task encoder $q_\phi(z|c)$. A version of Latent GNNs [74] has been successfully implemented in probabilistic meta-RL [63]. However, as suggested in section 4.2.1, we propose slight modifications in the affinity functions. We compare the representational power of our preferred task encoder to the Gaussian factors encoder used in PEARL and report the differences in Figure 6.19.

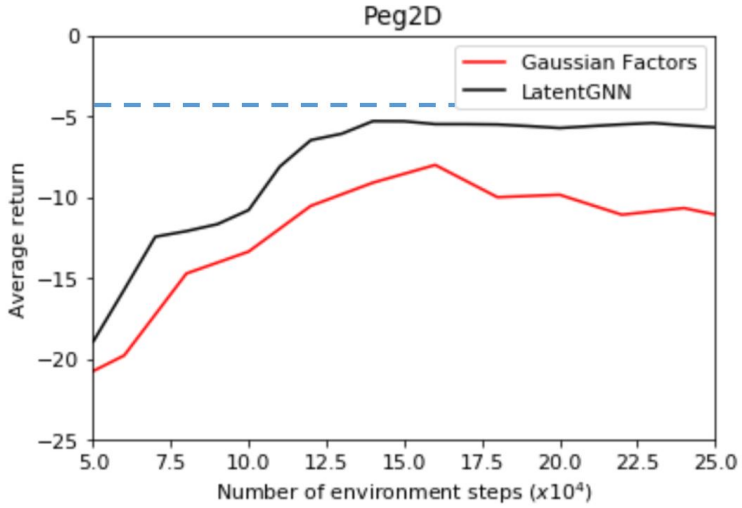

Figure 6.19: Test-task performance vs. number of samples collected during meta-training in ML1 when leveraging Latent GNN (PERIL) or Gaussian Factor encoders. Our approach showcases increased representational capacity. Blue dashed line indicates the average return from the expert trajectories.

Our Latent GNN seems to capture richer task descriptors in a more sample efficient and stable manner with respect to the task encoder used in PEARL. In light of this we support our claim that GNNs provide further representational power by: (i) Learning inter-dependencies between contextons generated from

the same task; (ii) Providing attention mechanisms which provide stability by reducing redundancies between contextons that provide little information gain.

6.3.4 Pure Meta-Imitation Learning

We study the effect of the meta-RL component in PERIL by ablating it and evaluating meta-training convergence. In essence, we are left with what we denote as a pure probabilistic meta-IL. By removing meta-RL, probabilistic meta-IL optimises the encoder based on the auxiliary information and mutual information only.

As in PERIL, we begin with 3 demonstrations for each task and use our "demo augmentation" method to augment this repertoire. In theory, with infinite expert trajectories and infinite sampling, this method would converge to produce policies which solve peg insertion in the ML1 setting. Nevertheless, as we demonstrate in Figure 6.20, given the 3 initial demonstrations, probabilistic meta-IL fails to converge at an efficient rate. Note that the average demonstration buffer size $|\mathcal{D}^{\mathcal{T}}|, \mathcal{T} \in p(\mathcal{T})$ at the point where meta-training is interrupted is augmented from 3 to 89 demonstrations per-task. Although this indicates that with sufficient sampling the algorithm could eventually converge, we can safely conclude that the meta-RL component in PERIL is crucial for stable, sample efficient learning.

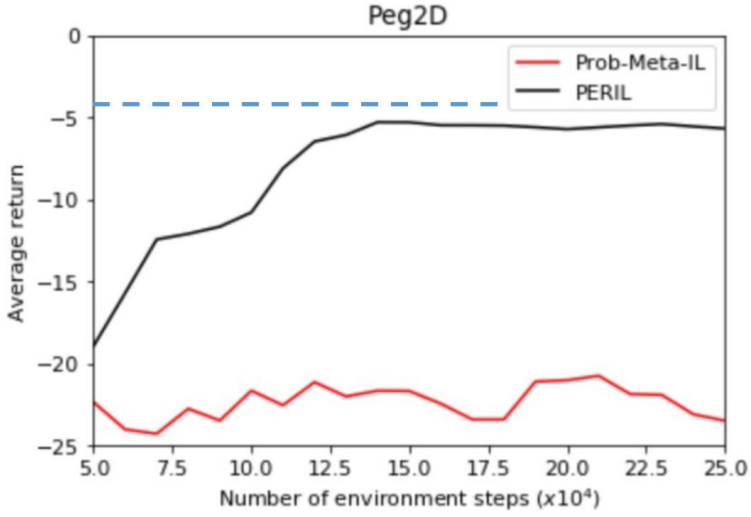

Figure 6.20: Test-task performance vs. number of samples collected during meta-training. By ablating the meta-RL component of PERIL, the algorithm fails to produce any indications of convergence in ML1. Blue dashed line indicates the average return from the expert trajectories.

6.3.5 Binary Rewards

In PERIL we leverage the fact that meta-training occurs during simulation such that we can feed the critic with naive dense rewards for each task. However, it is desirable to observe if PERIL can also exploit expert demonstrations to meta-learn under sparse RL settings such that real-world adaptation can also be supported. In this study we feed the same binary reward which is given to the task encoder to the critic. Under this setup we demonstrate that PERIL is capable of learning via sparse rewards (Figure 6.21). Nevertheless, as is expected (poor exploration via sparse rewards, large Q-function overestimates), learning is substantially more sample inefficient and unstable.

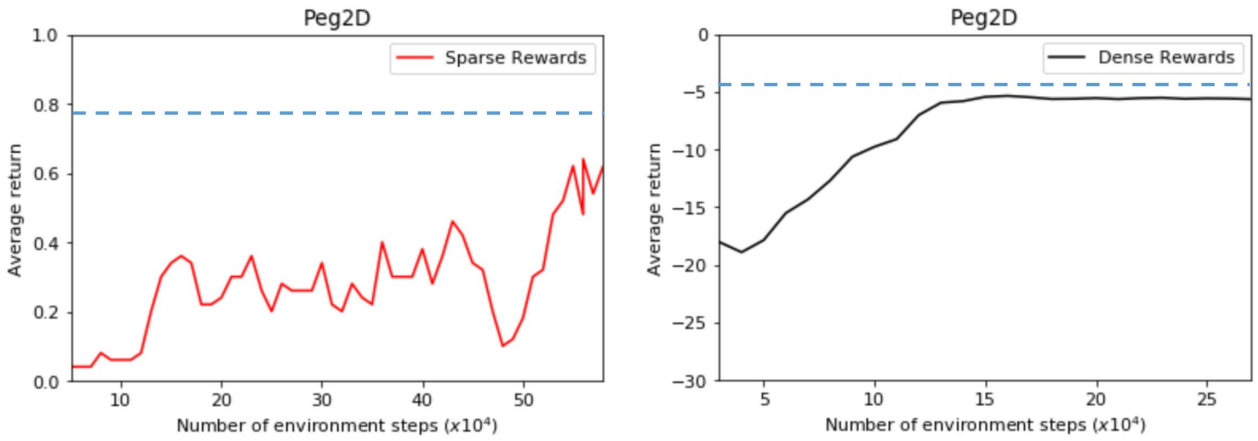

Figure 6.21: Test-task performance vs. number of samples collected during meta-training. By using sparse rewards, converge stability of PERIL under ML1 is substantially reduced. On the left plot the blue dashed line represents PERIL's converged sparse reward return. On the right, the blue dashed line represents the averaged expert return.

Chapter 7: Conclusion & Future Works

In this project we build on top of existing meta-RL methods to propose PERIL, a hybrid meta-reinforcement and meta-imitation learning approach to probabilistic task inference. Through extensive analysis we demonstrate that PERIL is capable of learning how to generate informative task representations from both demonstration and exploratory contexts to, in turn, condition a policy such that the latter can adapt unseen tasks.

To the best of our knowledge, PERIL is the first meta-RL algorithm capable of leveraging demonstrations to perform adaptation to different complex domains under continuous state-action spaces and sparse rewards. Moreover, our approach demonstrates substantially richer representational capacities, as well as sample efficiencies, with respect to state-of-the-art meta-RL algorithms. Via demonstration augmentation techniques, we leverage meta-IL to reason about the statistical structures within expert trajectories of different tasks such that we can provide a minimal set of demonstrations ($k = 3$) for meta-training. Furthermore, in contrast to other existing approaches, our method shows outstanding multi-tasking and meta-transfer learning abilities.

By learning how to learn, PERIL maps task beliefs of not only similar task dynamics but also different cardinalities into well defined vector spaces, showing that it is capable of task identification. In essence, the task encoder learns (in an unsupervised manner) how to create structures during meta-training such that it can produce a diverse set of task beliefs for each task family. This is a crucial attribute which can become particularly useful in robotic multi-tasking. In parallel works, we show how it can adapt efficiently to tasks of unseen dynamics by interpolating from within long-term memory clusters inherently imposed in the task encoder during meta-training. In effect, the latter observation provides indications that PERIL is capable of reasoning about an unseen task, inciting generalisation.

Evaluation in each one of the benchmarks devised for this project demonstrates outstanding adaptation performances under both perfect and imperfect settings. In contrast to existing methods, PERIL can perform zero-shot learning to unseen tasks. Even better, by giving a single demonstration, our method can readily adapt to perform any arbitrary task under the meta-training distribution, as shown in the ML and MTL benchmarks. Moreover, our method is capable of readjusting its task belief when given a single imperfect demonstration, by reasoning about its uncertainty. Last, via posterior sampling, PERIL can leverage online exploration to collect context from the environment so to improve its belief of the task. In light of this, we claim that our approach offers powerful yet robust adaptation abilities.

Ablation studies demonstrate the importance of the novel contributions we present in this project. By leveraging privileged information during meta-training, we prove that auxiliary modules can become particularly useful in meta-RL training by increasing sample efficiencies. Moreover, we validate our approach on linking demonstrations via mutual information maximisation by realising that the latter imparts regularisation on the task encoder. Furthermore, we support our decision of leveraging latent GNNs by proving that these are indeed equipped with increased representational power. In addition, we show that the meta-RL component of PERIL is crucial for stable learning. Since actor-critic training via sparse reward is the purest and most generalisable RL setup (also compatible with real-world robotic tasks), we also perform a study where we verify that PERIL is robust enough to support these adverse conditions.

7.1 Future Works

Given that we have demonstrated that PERIL is a powerful algorithm capable of enduring zero-shot learning in unseen tasks, there are several future works which we suggest to advance the state of the art. We begin by outlining future works which can further evaluate the robustness of our method in real conditions, followed by suggestions on how to improve the performance of our algorithm.

In this project we have used a 2D physics simulator (MetaSim) to study the intrinsic behaviours of our proposed method. Having verified different contact-rich tasks in this setup, we believe it would be particularly interesting to test our method in robotic frameworks such as MuJoCo [77] which support 3D control. In particular, we could evaluate multi-task and meta-transfer learning performances in MetaWorld [41], a set of benchmarks specifically devised for meta-learning in diverse robotic environments.

With regards to continuous control, we suggest the extension of the observational and action spaces (other than the additional degrees of freedom extended from 2D to 3D control) by including sensory information such as forces experienced by end effector (observational) or torque commands sent to the effector joints (action). In parallel or subsequently after performing the aforementioned studies, the ultimate step in evaluating our method would be to train real robotic manipulators under the PERIL framework. By meta-training over different dynamics, it would also be particularly interesting to evaluate if PERIL can readily transfer simulated behaviours to the real world by meta-learning over the "reality gap", as successfully performed via existing meta-RL methods in [6].

One caveat of our approach, as is also the case in most meta-RL methods, is that the policy does not support continuous short-term memory. Rather, it is solely conditioned on the task belief which is inferred from continuously collected contextons from the environment, without giving importance to the order in which these are collected. Instead of parametrising the task encoder with RNN-based models as studied in PEARL [51], we believe that exploration policies are those which should be equipped with short-term memory such that they can explore unseen tasks in a more efficient manner (remembering where they have been such that they are motivated to explore elsewhere). In light of this, we propose decoupled exploration and exploitation policy optimisation.

Regarding the imitation learning aspect of our method, in PERIL we use behavioural cloning (BC) to condition our agent on expert trajectories. During BC, every single transition carries the same weight in the computation of the log likelihood estimate for $\mathcal{L}_{bc}$. However, it is key to understand that not all transitions are equally important, hence cloning the behaviour of some redundant transitions can be counter-productive. In essence, more emphasis should be given to transitions which lead to higher expected rewards. For that matter using heuristic weighting functions (or meta-learning them) which focus on cloning information-rich transitions are amongst the future works envisioned.

Moreover, instead of manually defining a distribution of the alterations of each task, unsupervised curriculum learning could be employed to progressively increase the difficulty of each task with the aim of improving PERIL's convergence rates and robustness. Finally, given our method's highly structured task encoder, we motivate future works in the direction of exploiting the latent space clusters z to form behavioural repertoires. Form the latter, by exploiting PERIL's meta-policy conditioned on z, a separate hierarchical controller could be optimised via evolutionary algorithms or otherwise, to select different variables z in order to perform series of macro actions.

Chapter 8: Bibliography

[1] S. M. Hazarika and U. S. Dixit, "Robotics: History, Trends, and Future Directions," pp. 213–239, Springer, Cham, 2018.

[2] N. J. Nilsson and D. L. Nielson, "SHAKEY THE ROBOT," tech. rep., 1984.

[3] "Robots and the Workplace of the Future," tech. rep., International Federation of Robotics, Frankfurt, Germany, 3 2018.

[4] S. Rajana, "Robotics for the Supply Chain," no. April, 2018.

[5] H. R. Siebner, C. Limmer, A. Peinemann, A. Drzezga, B. R. Bloem, M. Schwaiger, and B. Conrad, "Introduction to Mechanical Engineering," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 22, no. 7, pp. 2816–2825, 2018.

[6] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks," 4 2020.

[7] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks,"

[8] J. A. Perez, F. Deligianni, D. Ravi, and G.-Z. Yang, "Artificial intelligence and Robotics," tech. rep., 6 2018.

[9] "Autonomous mobile robots grow beyond car manufacturing, get heavier loads - Xiaoxin Machines Pte Ltd."

[10] K. Fantian, C. Youping, X. Jingming, Z. Gang, and Z. Zude, "Mobile robot localization based on Extended Kalman Filter," in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, vol. 2, pp. 9242–9246, 2006.

[11] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. J. Leonard, and Leonard}, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, 2016.

[12] A. Rezaee, "Model predictive Controller for Mobile Robot," *Transactions on Environment and Electrical Engineering*, vol. 2, p. 18, 6 2017.

[13] M. Ben-Ari, F. Mondada, M. Ben-Ari, and F. Mondada, "Robots and Their Applications," in *Elements of Robotics*, pp. 1–20, Springer International Publishing, 2018.

[14] "Advantages and Disadvantages of Industrial Robots | Granta Automation."

[15] J. Falco, J. Marvel, and E. Messina, "Dexterous Manipulation for Manufacturing Applications Workshop," tech. rep., 2013.

[16] "Expert cooperative robots for highly skilled operations for the factory of the future Title : X-act Requirements analysis and specifications Reference : D1.2 Availability : Public," tech. rep., 2013.

[17] S. C. Diamantas, A. Oikonomidis, and R. M. Crowder, "Depth estimation for autonomous robot navigation: A comparative approach," in *2010 IEEE International Conference on Imaging Systems and Techniques, IST 2010 - Proceedings*, pp. 426–430, 2010.

[18] J. Josifovski, M. Kerzel, C. Pregizer, L. Posniak, and S. Wermter, "Object Detection and Pose Estimation Based on Convolutional Neural Networks Trained with Synthetic Data," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 6269–6276, Institute of Electrical and Electronics Engineers Inc., 12 2018.

[19] E. Galceran and M. Carreras, "A Survey on Coverage Path Planning for Robotics," tech. rep., 2013.

[20] P. F. Alcantarilla, L. M. Bergasa, and F. Dellaert, "Visual Odometry Priors for robust EKF-SLAM," tech. rep.

[21] H. Wang, "Adaptive Control of Robot Manipulators With Uncertain Kinematics and Dynamics," tech. rep., 2016.

[22] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep Dynamics Models for Learning Dexterous Manipulation," 9 2019.

[23] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," tech. rep.

[24] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to Trust Your Model: Model-Based Policy Optimization," tech. rep.

[25] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's Cube with a Robot Hand," 10 2019.

[26] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," 7 2019.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: Second Edition*, vol. 10. 2016.

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, pp. 3521–3526, 12 2016.

[29] T. Hester, T. Schaul, A. Sendonaris, M. Vecerik, B. Piot, I. Osband, O. Pietquin, D. Horgan, G. Dulac-Arnold, M. Lanctot, J. Quan, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 3223–3230, AAAI press, 4 2018.

[30] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning," 11 2015.

[31] B. Mehta, M. Diaz Mila, F. Golemo Mila, C. J. Pal Mila, P. Montréal, and C. Liam Paull, "Active Domain Randomization," tech. rep.

[32] M. Kaspar, J. David, M. Osorio, and J. Bock, "Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization," tech. rep.

[33] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," tech. rep., 2018.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, 9 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," tech. rep.

[36] J. Clune, "AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence," 5 2019.

[37] S. Ritter, J. X. Wang, Z. Kurth-Nelson, S. M. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick, "Been There, Done That: Meta-Learning with Episodic Recall," tech. rep., 2018.

[38] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Under review as a conference paper at ICLR 2017 RL 2 : FAST REINFORCEMENT LEARNING VIA SLOW REINFORCEMENT LEARNING," tech. rep.

[39] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-Reinforcement Learning of Structured Exploration Strategies," *Advances in Neural Information Processing Systems*, vol. 2018-December, pp. 5302–5311, 2 2018.

[40] V. C. Müller, "Ethics of Artificial Intelligence and Robotics," 2020.

[41] N. Blair, V. Chan, and A. Karnati, "Analyzing Policy Distillation on Multi-Task Learning and Meta-Reinforcement Learning in Meta-World," tech. rep.

[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 6 2016.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning,"

[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 7 2017.

[45] S. Tu and B. Recht, "The Gap Between Model-Based and Model-Free Methods on the Linear Quadratic Regulator: An Asymptotic Viewpoint," 12 2018.

[46] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 5 1992.

[47] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," 5 2015.

[48] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An Introductory Review of Deep Learning for Prediction Models With Big Data," *Frontiers in Artificial Intelligence*, vol. 3, p. 4, 2 2020.

[49] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 1 2019.

[50] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-Learning in Neural Networks: A Survey," 4 2020.

[51] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables," tech. rep.

[52] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess, "Meta reinforcement learning as task inference," 5 2019.

[53] I. Osband, D. Russo, and B. Van Roy, "(More) Efficient Reinforcement Learning via Posterior Sampling," *Advances in Neural Information Processing Systems*, 6 2013.

[54] R. Sheh, B. Hengst, and C. Sammut, "Behavioural cloning for driving robots over rough terrain," pp. 732–737, Institute of Electrical and Electronics Engineers (IEEE), 12 2011.

[55] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics," in *Industrial Robotics: Theory, Modelling and Control*, Pro Literatur Verlag, Germany / ARS, Austria, 12 2006.

[56] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, pp. 161–185, 1 2016.

[57] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, pp. 4397–4404, Institute of Electrical and Electronics Engineers Inc., 6 2015.

[58] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization," tech. rep.

[59] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3389–3396, 10 2016.

[60] 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," 7 2017.

[61] A. Pritzel, S. Srinivasan, A. Com, D. Hassabis, D. Wierstra, and C. Blundell, "Neural Episodic Control Adrì a Puigdomènech," tech. rep.

[62] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," tech. rep., 2017.

[63] H. Wang and J. Zhou, "Learning Context-aware Task Reasoning for Efficient Meta-reinforcement Learning KEYWORDS Multitask Learning; Deep Reinforcement Learning," tech. rep., 2020.

[64] S. Ross, G. J. Gordon, J. A. Bagnell, and M. Learning, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," tech. rep.

[65] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction," tech. rep.

[66] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *Advances in Neural Information Processing Systems*, pp. 4572–4580, 6 2016.

[67] P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, and D. Precup, "OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning," *AAAI2018*, pp. 3199–3206, 9 2017.

[68] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," tech. rep.

[69] A. Zhou, E. Jang Google Brain, D. Kappler, A. X. Herzog, M. Khansari, P. Wohlart, Y. Bai, M. X. Kalakrishnan, S. Levine, and C. Finn Google Brain, "WATCH, TRY, LEARN: META-LEARNING FROM DEMONSTRATIONS AND REWARDS," tech. rep.

[70] L. Yu, T. Yu, C. Finn, and S. Ermon, "Meta-Inverse Reinforcement Learning with Probabilistic Context Variables," tech. rep.

[71] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 10 2017.

[72] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "ON THE INFORMATION BOTTLENECK THEORY OF DEEP LEARNING," tech. rep.

[73] B. Dai, C. Zhu, and D. Wipf, "Compressing Neural Networks using the Variational Information Bottleneck," tech. rep.

[74] S. Zhang, S. Yan, and X. He, "LatentGNN: Learning Efficient Non-local Relations for Visual Recognition," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 12767–12776, 5 2019.

[75] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The Ingredients of Real-World Robotic Reinforcement Learning," 4 2020.

[76] A. Cully and Y. Demiris, "Hierarchical Behavioral Reper-toires with Unsupervised Descriptors," 2018.

[77] http://www.mujoco.org/, "MuJoCo."

Chapter 9: Appendix

9.1 Derivations

9.1.1 Conditional Behavioural Cloning

The derivation for the following BC equivalent is novel. We define $\mathcal{L}_{bc}$ as the conditional behavioural cloning loss:

$$\mathcal{L}_{bc} = \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)] \tag{9.1}$$

By manipulation of the Kullback-Liebler divergence we obtain the following expansion.

$$\min_\theta \mathbb{E}_{p(z)}[D_{KL}(p_{\pi_E}(\tau|z)||p_\theta(\tau|z)] = \min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\left[\frac{\log p_{\pi_E}(\tau|z)}{\log p_\theta(\tau|z)}\right]$$

$$\min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\left[\frac{\log p_{\pi_E}(\tau|z)}{\log p_\theta(\tau|z)}\right] = \min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\Big[\log p_{\pi_E}(\tau|z)\Big] - \min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\Big[\log p_\theta(\tau|z)\Big]$$

We then substitute policy π_θ as the conditional distribution $p_\theta(\tau|z)$ and our variatonal inference approximation for the posterior $q_\phi(z|\tau)$:

$$\min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\Big[\log p_{\pi_E}(\tau|z)\Big] - \min_\theta \mathbb{E}_{z\sim p_\theta(z|\tau)}\Big[\log p_\theta(\tau|z)\Big] = \min_\theta \mathcal{H}(p_{\pi_E}(\tau|z)) - \mathbb{E}_{z\sim q_\phi(z|\tau)}\Big[\log \pi_\theta(\tau|z)\Big]$$

Because the entropy term is not dependent on the parameters θ, we derive a lower bound objective which is identical to the maximum likelihood estimator for trajectory distributional matching in BC:

$$\mathcal{L}_{bc}^{\mathcal{T}} = -\mathbb{E}_{\tau\sim p_{\pi_E}^{\mathcal{T}}(\tau), z\sim q_\phi(z|\tau)}[\log \pi_\theta(\tau|z)] \tag{9.2}$$

Note that in the final representation (9.2), expert trajectory distributions $p_{\pi_E}^{\mathcal{T}}(\tau)$ are conditioned by task $\mathcal{T}$ as would be the case during meta-training.

9.2 Training

9.2.1 Hyper-parameters

Unless otherwise noted, the default hyperparameters for PERIL are as described below.

Training Loop

- Action space $|a|$: 3

- Observational space $|o|$: 4
- Latent space $|z|$: 8
- Maximum path length H: 60
- Prior collection steps K_{prior}: 240
- Posterior collection steps $K_{posterior}$: 240
- RL batch size: 256
- Encoder batch size: 64
- Number of expert demonstrations K: 3
- RL sparse rewards: False
- Task belief sparse rewards: True
- Maximum context size: 256
- Number of pre-training steps: 1000
- Number of training steps: 1000
- Number of tasks to average gradients in meta-training R: 16
- Number of tasks to collect data from each epoch: 32
- Number of Epochs: 100
- Discount factor γ: 0.99
- Entropy temperature: 0.1
- Soft target value = 0.005
- Imperfect demos: True
- Alter tasks: True
- Kullback-Liebler divergence weighting w_{KL}: 0.1
- Behavioural cloning weighting w_{bc}: 0.5
- Mutual information weighting w_{mi}: 0.2
- Auxilliary loss weighting w_{aux}: 4.0

Actor

- Hidden layer size h: 256
- Architecture: $[|o, a, z|,\ h,\ h,\ 1]$
- Learning rate α_π: 3×10^{-4}
- Output: Bounded action [-1,1] via tanh

Critic

- Hidden layer size h: 256
- Architecture: $[|a, z|,\ h,\ h,\ 1]$
- Learning rate: α_Q: 3×10^{-4}

Encoder

- Number of kernels: 2
- Latent size: d
- Input affinity architecture: $[|o, a, r, o|, 256, 256, d]$
- Output affinity architecture: $[d, 64, 64, |z|]$
- Learning rate: α_q: 3×10^{-4}

Belief Module

- Task descriptor dimension $|b|$: 2
- Architecture: $[|z|, 32, 16, |b|]$
- Learning rate: α_b: 3×10^{-4}

9.2.2 Additional Results

In this project we also defined an additional benchmark MTL(6+1) where we train on all task families (Peg2D, vertical Peg2D, Reach2D, Stick2D, vertical Stick2D, Key2D) except vertical Key2D (Key2Dv). At test time, the agent must adapt to tasks selected from the latter unseen task family. We explicitly choose not include results for MTL(6+1) as these did not provide items to further extend our discussion, other than demonstrating that meta-transfer learning is also applicable in more complex multi-task settings (Figure 9.1). Moreover, we demonstrate PERIL's reasoning over unseen tasks in the Key2Dv task family by observing the latent space mapping in Figure 9.2.

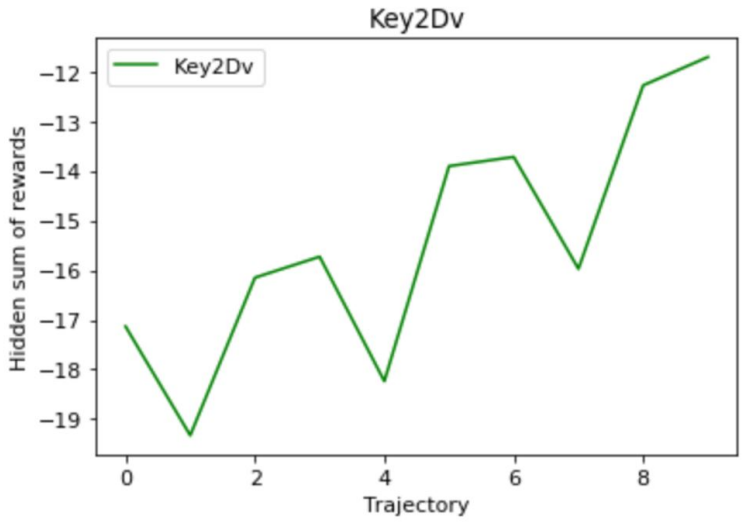

Figure 9.1: Average adaptation performance (along 10 unseen tasks per task family) for PERIL agents on the MTL(6+1) benchmark. PERIL demonstrates robust adaptation to meta-transfer learning along tasks from the unseen Key2Dv task family.

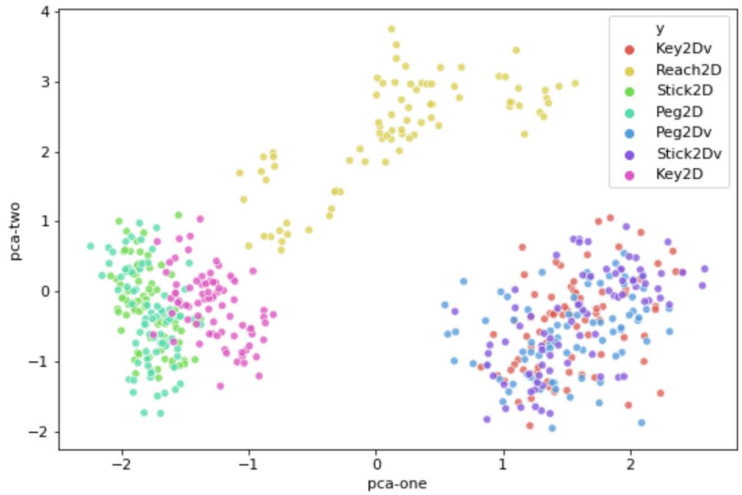

Figure 9.2: PCA on the principal two components of the latent context variable z distribution sampled for all task within each of the 6 training task families (X) in MTL(X + Y) as well as the unseen task family (Y) vertical Key2D (Key2Dv). Notice that during meta-training, PERIL distinguishes Key2D from Stick2D and Peg2D by a greater extent than during adaptation of the latter.

9.3 Environments

9.3.1 Experts

Experts for task families: Reach2D, Peg2D, Stick2D, Key2D and their vertical equivalents are trained via SAC. An example of the convergence behaviour for an expert is presented in Figure 9.3.

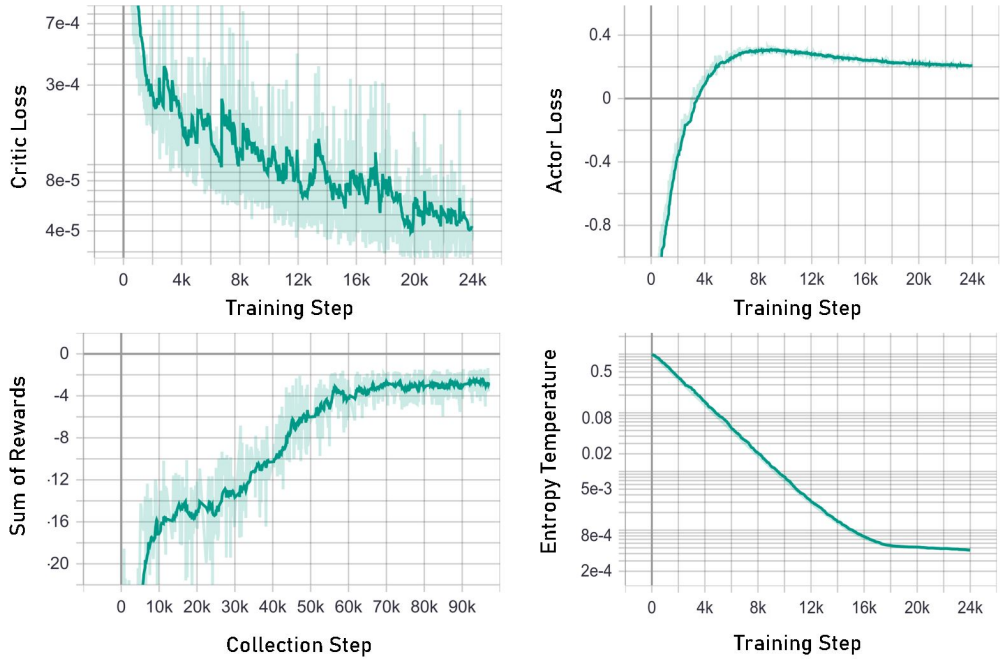

Figure 9.3: Convergence of the expert agent on the Peg2D task family.