Università degli Studi di Salerno

Dipartimento di ingegneria dell'Informazione ed Elettrica e Matematica applicata

Ph.D. Program in Information Engineering

First year report

Author:
dott. Francesco Rosa

Supervisor:
Ch.ma. Mario Vento

XXXVII cycle - Academic year 2021/2022

Overview

The report represents the final document of the first year of the doctorate carried out by the undersigned. The document conforms to the latest version of the document *"Requirements for the PhD degree"* approved on 14/12/2020 by the professors' college and available on the university platform in the area reserved for doctoral students. As requested, the document contains a report on the activities carried out by the student during the first year.

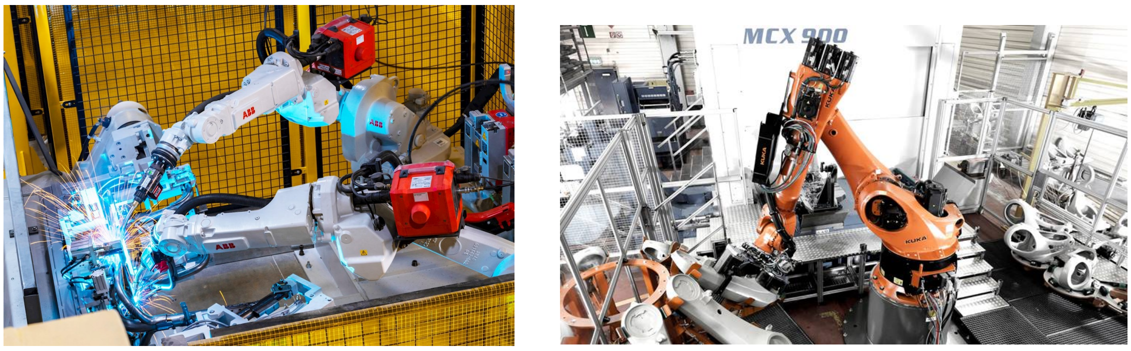
The document is divided, as suggested in the aforementioned document, into 4 chapters:
The first chapter ...
The second chapter ...
The third chapter contains ...
The last chapter ...

Chapter 1

Background

1.1 Introduction

Robotic technology is one of the pillars of modern society. Thanks to advances in information, electronic, and mechanical fields, we can build and program machines to solve various tasks in different scenarios, e.g. industrial, surgery, space missions etc.
In manufacturing, given their ability to reproduce the same movement over time with an high level of accuracy and precision, robots are massively used to solve repetitive and dangerous activities such as assembly, welding (Figure 1.1a) and material handling (Figure 1.1b).



(a) Robots involved in arc welding operation

(b) Robot involved in loading operation

Figure 1.1: Industrial Robots: example of applications

While in the early day, robotic systems were constrained in isolated and known environments. Over the past few decades, robots have been asked to solve tasks in dynamic and unknown/partially-known environments, where they must **coexist** and **cooperate** with humans, while solving different **dynamic** tasks (e.g. pick a requested object, whose position is not known a priori).

The reason for the increasing demand for such applications may be found in factors such as:

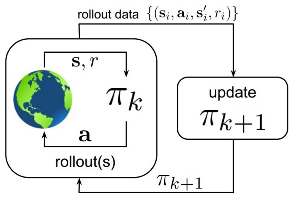
(a) The affirmation of the *Industry 4.0* paradigm, in which the various phases of a factory are linked to each other and therefore the robots must be in communication with each other and be able to adapt to changes in the production phases.

(b) The spreading of *social robots* (e.g. Pepper [1]), that find their appeal in highly unstructured applications such as household or shop-recommendation systems.

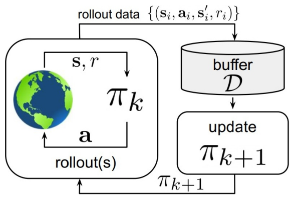
(c) The increasing availability of the so-called *cobots* (e.g. Franka Emika Panda robot [2]), which are low-cost robots that allow the automation of light load processes (e.g. drilling, palletizing, polishing) while sharing the workspace with humans or other robots. Thanks to cobots, also small factories, that were left out from this automation process because of the high-cost of previous solutions, can now start to automate their production phases, contributing to increase the demand.

In this scenario, the desired characteristics of such robotic systems are:**(a)** The capability to exhibit *"intelligent" behaviors* during the task execution in response to dynamic changes in the environmental condition (e.g. pre-grasping manipulation [3], motor failure and environmental uncertainties [4]). **(b)** The ability to easily *adapt* to new tasks [5].
These requirements can be difficult to obtain with typical robotic programming techniques based on hand-written policy, and control techniques which require a careful analysis of the process dynamics, the building of an analytic model, and finally, the derivation of a control law that meets certain design criteria [6]. This design process is tedious, and time consuming, especially when **high-level perception systems** (e.g. camera, microphones, motion sensors etc.) are used to infer the state of the environment (e.g. the unknown position of the desired object to pick with respect to the end-effector) and/or the intention of the human operator.
In contrast, very relevant results have been obtained by leveraging *Learning Techniques*, which exploit **agent experience** or **expert demonstration**. Generally, the former is referred as *Reinforcement Learning* (RL) [7], while the latter is referred as *Imitation Learning* (IL) or *Learning from Demonstration* (LfD) [8]. Both the approaches aim to obtain an artificial agent, that able to perform correctly a task or a set of tasks by following a learned policy, however the way how this goal is achieved is quite different. The main differences between RL and IL are related to presence/absence of a reward function, and the presence/absence of expert demonstrations.
Indeed, in RL formalism [9], the learning procedure is based on a *hand-designed reward function*, i.e. a measure of the quality of the performed action with respect to the task to be executed (e.g. the distance between the current position and the target one in a reaching task [**?**]), which drives the agent to produce a sequence of actions which maximize the reward. While in IL formalism [10], the learning

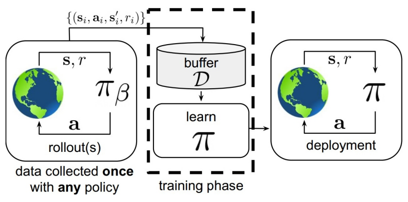
procedure is based on the minimization of a *loss-function* (e.g. Mean-Squared Error [11], Hinge-loss [12], Kullback-Leibler divergence [13]), which guides the agent to reproduce/mimic the same actions of the expert.
Regarding the possibility to exploit past experience, in RL literature, three approaches can be found [14]: • ***On-policy*** Reinforcement Learning (Figure 1.2a), the trained policy is the same policy used to collect data, so, the only used past experience is related to the last policy rollout. These methods require a considerable amount of interactions to be trained [?]. • ***Off-policy*** Reinforcement Learning (Figure 1.2b), there are two policies, the behavior policy used to get experience, and the target policy, the policy to evaluate and improve. The target policy can exploit past experience during training. These methods are known to be more data-efficient [?], although they still require interaction with the environment. • ***Offline*** Reinforcement Learning (Figure 1.2c), this setting is similar to the classic *supervised learning approach*, where the target policy is trained via a *static dataset* D, collected by an unknown policy π_β.



(a) On-policy RL



(b) Off-policy RL



(c) Offline RL

Figure 1.2: Graphical Representation of Reinforcement Learning Methods [14]

The need for a task-specific reward function, and the need for expensive and time-consuming training procedures, are the main difficulties in applying such methods for real-world application [15].
As opposed to RL, in IL, the starting point for all the methods (Figure 1.3), is a dataset D containing expert demonstrations (e.g. trajectories of teleoperated robot [?], video of human pouring a liquid [?]). Based on the approach, the information contained in the dataset is used in different way: • ***Behavioral Cloning*** (BC), the information contained in the dataset is used as *ground truth*, i.e. the final learned policy mimics the same actions in the dataset [?]. • ***Inverse Reinforcement Learning*** (IRL), the demonstrations are used to learn a *cost function*, that is subsequently used by an RL algorithm [?], • ***Generative Adversarial Imitation Learning*** (GAIL), combination of GAN [?] and IL, in this setting the agent, which acts as the Generator in GAIL, must produce a sequence of state transitions, such that Discriminator is not able to distinguish between generated transitions and demonstrated ones [?], • ***Learning from Observation*** (LfO), inspired by the fact that humans and animals are able to learn by just watching a demonstration, without knowing the underlaying performed actions (e.g. the joint position), then the aim of the artificial agent is to reproduce the observed behavior, starting from state-only demonstrations [16].

From Figure 1.3, it can be noted as IL and RL are strictly related, indeed, RL algorithms are used in the training procedure of some methods, such as IRL, GAIL,

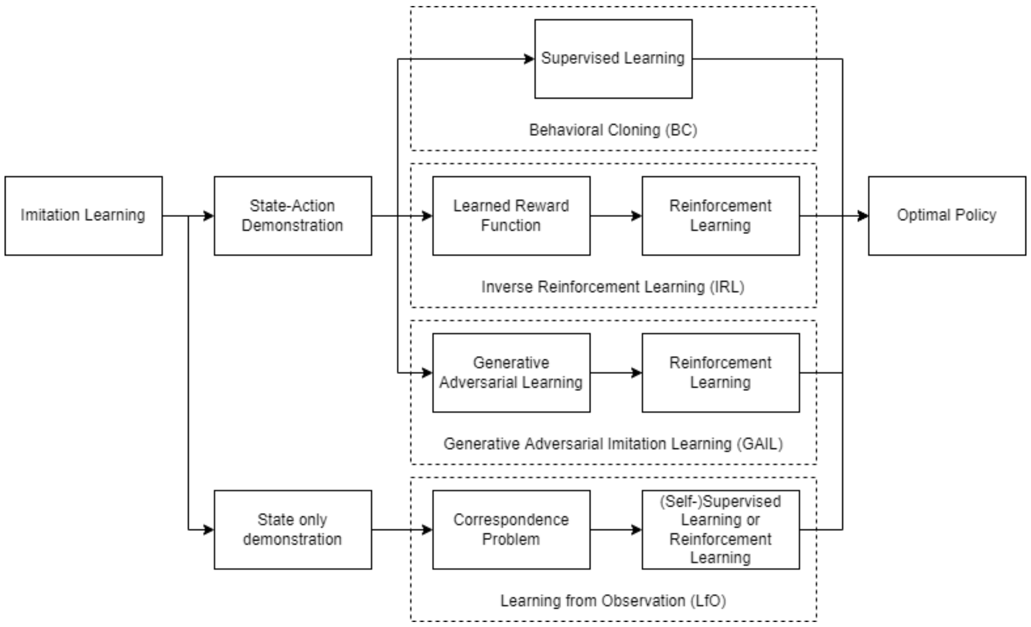

Imitation Learning
State-Action Demonstration
State only demonstration
Supervised Learning
Behavioral Cloning (BC)
Learned Reward Function
Reinforcement Learning
Inverse Reinforcement Learning (IRL)
Generative Adversarial Learning
Reinforcement Learning
Generative Adversarial Imitation Learning (GAIL)
Correspondence Problem
(Self-)Supervised Learning or Reinforcement Learning
Learning from Observation (LfO)
Optimal Policy

Figure 1.3: Imitation Learning: Taxonomy and main components

and LfO. However, with respect to classic RL, IL allows to: **(1)** bypass, or at least attenuate, the time-consuming exploration that would be required in a RL setting, **(2)** communicate, through demonstrations, a "preference" for how a task should be performed. **(3)** describe concepts that may be difficult to define formally or programmatically.. As will be explained in detail in Section 1.2, BC methods suffers of the *compounding error* problem, since the i.i.d. assumption that lays behind Supervised Learning is violated [**?**]. IRL was proposed to combine both the data-efficiency of IL, to learn a reward function given the samples, and the exploration capability of RL, to attenuate the *compounding error* problem. However such methods have proved difficult to train. GAIL was proposed to reduce the training complexity related to IRL, by framing the IRL problem as an Adversarial Learning problem, while the methods improved both the performance and the efficiency of w.r.t. IRL methods, they showed poor performance when applied in high-dimensional data, mainly due to the *casual-confusion* problem [**?**]. LfO [**?**] was mainly proposed with the aim to find a way to exploit state-only demonstrations, that are easy to obtain, and to collect. However, since there is not any information about the true action,a way to measure the quality of prediction is needed. Moreover, the *correspondence problem*, i.e. how to map demonstration in human-space into the corresponding action in the robot space, must be solved.
Despite all the open-challenges presented, IL was successfully applied for solving complex tasks, such as driving a toy car [17], pick-and-place [18], and collaborative toolbox assembly [19], highlighting the high potential of such methods, which may justify an effort in an attempt to resolve the gaps presented. A detailed description of the mentioned approaches will be given in *Section* 1.2, with an exhaustive comparison and description of the most recent and relevant methods.

1.2 State-of-the-Art (last page 27)

The chapter provides a review of the State-of-the-Art related to the *Learning from Demonstration* problem. The chapter will start with a general problem formulation (Section 1.2.1). Then it will go trough a brief presentation of the different methods used to collect data (Section 1.2.2). Next, the proposed methods will be explained and compared (Section 1.2.3). In the end, a summary will be presented, where a cross-approach comparison will be performed and the current challenges will be emphasized (Section 1.2.4).

1.2.1 Problem Definition

Imitation Learning aims to obtain an agent that can replicate the behavior demonstrated by an expert agent. The behavior is described by a **policy**, where π^L defines the learned policy, while π^E defines the expert policy.
The policy π^L is obtained starting from a dataset $\mathcal{D} = \{(\boldsymbol{\tau}_i, \boldsymbol{c}_i)\}_i^N$, where:

- $\boldsymbol{\tau}_i$ is the i^{th} demonstrated trajectory, which can be described as:
  - A state-action sequence, i.e. $\boldsymbol{\tau}_i = [s_0, a_0, \dots, s_T, a_T]$.
  - A state sequence, i.e. $\boldsymbol{\tau}_i = [s_0, \dots, s_T]$.
- $\boldsymbol{c}_i$ is the *context-vector*, it contains task-related information, e.g. the initial state of the system s_0, or the position of the target position.

The policy π^L can be defined with respect to different abstraction levels [20, 10]: (**a**) *Symbolic Characterization*, used in [?], the policy maps states, and context to a sequence of options, i.e. $\pi : \mathbf{s}_t, \mathbf{c} \rightarrow [o_1, \dots, o_T]$, where each option is a sequence of actions. With this representation, complex tasks can be decomposed into a sequence of simple movements. However, it is hard to achieve an accurate task segmentation and motion ordering. (**b**) *Trajectory Characterization*, used in [?], the policy maps contexts to trajectory, i.e. $\pi : \mathbf{c} \rightarrow \boldsymbol{\tau}$. Since it allows to map initial state to a complete sequence of actions, this representation can be used to obtain the options in the Symbolic Representation. However, they need as many dynamic features as possible, that can be difficult to obtain. (**c**) *State-Action Characterization*, used in [?], the policy maps states(-context) to actions, i.e. $\pi : \mathbf{s}_t, \mathbf{c} \rightarrow a_t$. This representation allows a direct mapping between current state and the corresponding action. However, it is easy to accumulate errors in long-term process. The agent, by means of its policy π, acts on a system, which is modeled as a *Markov Decision Process* (MDP) [21]. A MDP is defined as a tuple (S, A, R, T, γ), where:

- $S \subseteq \mathbb{R}^n$, is a set of states (e.g. joints position [?] and/or image [?]).
- $A \subseteq \mathbb{R}^n$, is a set of actions, (e.g. desired end-effector position [?], desired joints torque [?]).
- $R(s, a, s')$, is a *reward function*, which expresses the immediate reward for executing action a in state s, and transitioning to state s'.

- $T(s'|s,a)$ is a *transition function*, which defines the probability to reach the state s', after the execution of action a in state s. This distribution, which describe the *system dynamic*, that can be given a priori or learned (Model-Based [?]), or do not taken into account (Model-Free [?]).
- $\gamma \in [0,1]$, is a discount factor expressing the agent's preference for immediate over future rewards.

With respect to the MDP definition, in BC methods the reward function is not implicitly used, but a surrogate loss-function is used. In IRL the reward function is learned, assuming that the expert acts (near-)optimal w.r.t. some unknown reward function that has to be learned. In GAIL and LfO, the rule played by the reward function is different based on the specific method, as will be explained in Section 1.2.3.

1.2.2 Source of Demonstration [MAX 5]

Regarding the way how the expert demonstrations can be obtained, according to [20], two categories can be identified: **(a)** Direct Demonstration. **(b)** Indirect Demonstration.

Direct Demonstration
It allows to obtain samples directly from the robot, either with kinesthetic teaching (Figure 1.5a) or teleoperation teaching (Figure 1.5b).In kinesthetic teaching the human operator contacts and guides the robot, that collects data by itself, while in teleoperation teaching the human operator remotely guides the robot with joystick, tactile sensor, control panel, and wearable device. The former is characterized by the fact that there is no need to consider difference in kinematic between human and robot, as consequence, data has less noise. However, the robot must be passively controllable and require direct contact, introducing safety problems, and unintuitive demonstrations for robot with multiple degrees of freedom. The latter is characterized by a higher safety, since there is no direct contact, and a wide application range. A very popular example of teleoperation system is *Roboturk* [22], it is a cloud-based teleoperation framework that enables the collection of high-scale demonstration dataset [23, 24], for both simulated and real-world robots, using a mobile-phone as controller (Figure 1.4). While this framework is interesting because it allows to collect data from a wide range of demonstrator with different demonstrations quality, it lacks of tactile information.

Indirect Demonstration
It allows samples to be acquired in a way that is completely disconnected from the robotic platform. In this case the human motion is recorded by vision system [?] or wearable device [?]. In last years, the ability to extrapolate a policy starting from video of human performing a task has become a very important topic in the current literature [20, 16]. While Indirect Demonstration based on visual system allows to collect samples in the most intuitive and scalable way possible, potentially any video

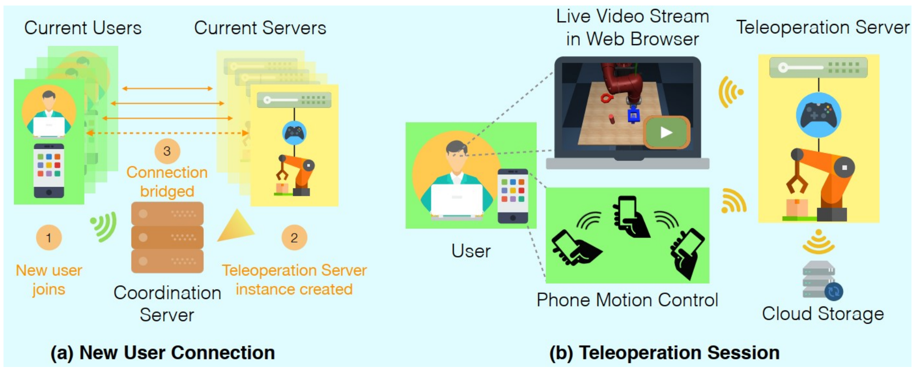
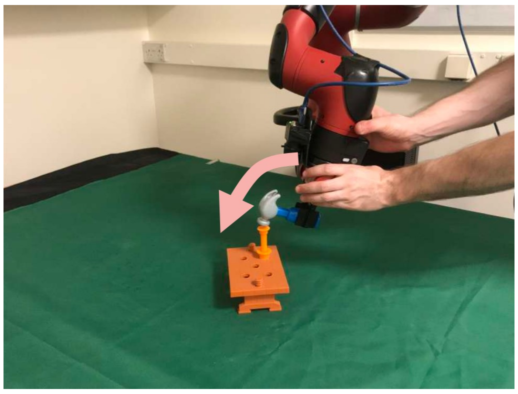
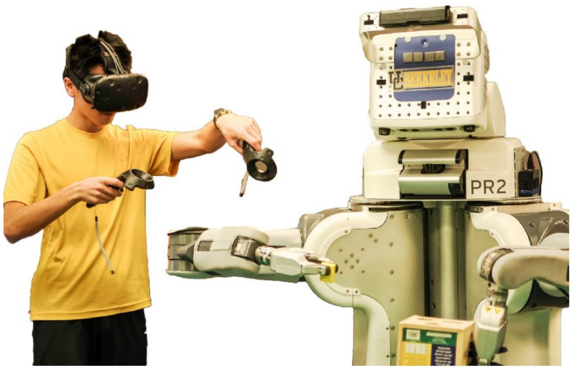


Figure 1.4: System diagram of Roboturk [22]

of performed task can be used, it lacks of relevant information such as the tactile one, moreover, the correspondence problem must be solved, i.e. the system must be able to map motion captured in human space into the corresponding motion of the robot. In Section 1.2.3 the different way how this problem has been solved in the current literature will be explained in detail.


(a) Example of kinesthetic teaching [25]



(b) Example of teleoperation [26]

Figure 1.5: Examples of Direct Demonstration

1.2.3 Methods

Behavioral Cloning (BC). [MAX 5]
Behavioral Cloning is one of the first approaches used to learn a task from a set of demonstrations. It is framed as a Supervised Learning problem, solved through Algorithm 1. In this setting there are at least to questions to answer: **(1)** Choose whether to optimize in trajectory space or action space. **(2)** What kind of representation to use for the policy. Generally speaking, Algorithm 1 defines the procedure used to solve the learning task, given the dataset $\mathcal{D}^E$, a parameterized learner policy π^L_θ, and a loss-function $\mathcal{L}$, the goal is to find the policy parameter that minimizes the loss-function, in other terms, $\theta^* = \underset{\theta}{argmin}\ \mathbb{E}_{(\boldsymbol{\tau},\mathbf{c})\sim\mathcal{D}^E}\ [\mathcal{L}((\boldsymbol{\tau},\mathbf{c}),\ \pi^L_\theta)]$.
According to [10, 27], BC methods can be categorized as Model-Free and Model-Based, and based on the fact that the policy is optimized either in the trajectory space or in the action space.

```
Algorithm 1 Abstract Algorithm for Behavioral Cloning
Require: A set of expert demonstrations $\mathcal{D}^E$, a parameterized policy $\pi^L_\theta$
Ensure: The optimal set of policy parameter $\theta^*$
  Optimize $\mathcal{L}$ w.r.t. policy parameter $\theta$ using $\mathcal{D}^E$
```

Model-Free methods learns a policy that reproduces the expert's behavior without learning/estimating system dynamics. Since they do not require neither to estimate the system dynamic nor to learn a reward function, they are simple to implement and do not necessary require system interactions.

Model-Free methods that derive policy in the space of trajectories were very popular in the context of Model-Free BC for robotic manipulation trajectory planning, given their ability to explicitly model constraints on the generated trajectory (e.g. a smooth convergence to the goal state). Indeed, in this setting very popular and studied methods are the *Dynamic Movement Primitives* (DMPs) [28, 29], and the *Probabilistic Movement Primitives* (ProMPs) [**?**]. [To continue...]

Regarding the methods that derive the policy in the action-space. One of the primal work is such setting was [30], which proposed *ALVINN*, an autonomous vehicle driving system based on a Neural Network, that infers the steering angle, given a camera image as input. The network was trained on pairs (image, steering-angle), the steering-angle was discretized over 45 units, and the training was defined as a supervised classification problem. This work immediately emphasized the problem of compounding-error, caused by covariate-shift phenomena. This issue occurs because an action a_t influences the next state s_{t+1}, violating the i.i.d assumption of Supervised Learning, and generating a test-data distribution, that may be different from the the training one. This phenomena has a relevant consequence on the expected performance of the system. Indeed, assuming to have a system that makes an error with probability ϵ, and a task with time-horizon T, then, due to compounding error, a supervised learner reaches a total cost of $O(\epsilon\ T^2)$, rather than $O(\epsilon\ T)$ [31, 32]. To attenuate this problem, interactive supervised learning algorithms have been proposed, such as the well-known *DAgger* [32]. Algorithm 2 describes the DAgger procedure. It is an aggregation strategy, based on the idea to train the policy π under the state-distribution induced by the policy itself, but with the correct action performed by the expert. the main problem with DAgger is that it requires the expert to interact with the system during the training, introducing both safety and data-efficiency problems, especially when the system does not provide the human expert with sufficient control authority during the sampling process [33].

Algorithm 2 DAgger Algorithm [32]

```
**Require:** Initial Dataset $\mathcal{D} \leftarrow \emptyset$, Initial policy $\pi_1^L$
**Ensure:** The best policy $\pi_i^L$
  **for** $i = 1, \dots N$ **do**
      Sample $T - step$ trajectories using $\pi_i^L$
      Let $\mathcal{D}_i = (s_t, \pi^E(s_t))$, state $s_t$ visited by policy $\pi_i^L$,
      and actions given by the expert
      Aggregate Dataset, $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$
      Train policy $\pi_i^L$ on $\mathcal{D}$
      Let $\pi_{i+1}^L = \beta_i \pi^E + (1 - \beta_i)\pi_i^L$
  **end for**
```

Human-Guided DAgger (HG-DAgger) [34] is an extension of the classic DAgger strategy, in which the human expert observes the rollout of the current policy, so if the agent has entered an unsafe region of the state space, the expert takes control and guides the system to a safe and stable region. In [5] was shown how HG-DAgger can be effectively used in the context of robotic manipulation. Indeed, starting from the same total number of episodes, a policy trained with only expert demonstration has a significantly lower success rate than a policy trained on a dataset with both expert demonstration and expert adjustments. In the context of Interactive Learning for Robot Manipulation, other works of interest include [35, 36]. In [36], a human expert provides both corrective and evaluative feedback. The former consists in the human that takes control of the robot to adjust the trajectory, the latter consists in a scalar weight q, set to 1 if the trajectory is satisfactory, 0 the trajectory is not satisfactory, α if the trajectory is adjusted by the expert, where α is the ration between non-corrected and corrected samples. Then a Neural Network was trained by minimizing a weighted version of the maximum-likelihood $\mathcal{L}(a_t, s_t) = -q\ log(\pi_\theta^L(a_t|s_t))$. Real-world experiments show that with a training time of **41 minutes**, including environmental reset, it is possible to have an agent capable of performing tasks such as picking up a cube or pulling a plug.

Despite the covariate-shift problem, [26] showed that very interesting performance can be obtained in the context of Robot Manipulation Task, by means of Behavioral Cloning and high quality demonstrations given by teleportation system. In this work, a CNN was trained to predict the desired linear-velocity and angular-velocity of the end-effector, with the binary gripper state (open/close), given in input the current RGB-D observation of the scene, and the position of three points of the end-effector, during the last 5 time-steps. The system was tested on 10 tasks, and the performance are reported in Table 1.1.

Table 1.1: Statistics of Training set, and Test Success rate [26]

task	reaching	grasping	pushing	plane	cube	nail	grasp-and-place	grasp-drop-push	grasp-place-x2	cloth
#demo	200	180	175	319	206	215	109	100	60	100
demo duration (min)	13.7	11.1	16.9	25.0	12.7	13.6	12.3	14.5	11.6	10.1
Test success rate (%)	91.6	97.2	98.9	87.5	85.7	87.5	96.0	83.3	80.0	97.4

One important aspect to note is that for each task the network was trained from scratch. This is not a desired property for a highly adaptable system, as stated in Section 1.1. For this reason, methods based on Meta-Learning algorithms have been proposed. The idea behind Meta-Learning is to train a model on a variety of tasks, in such a way that it can solve a new tasks, using only a small number of training samples [37]. Algorithm 3 describes the steps followed by the *Model-Agnostic Meta-Learning* (MAML) algorithm [37], that is the base for different methods which apply One-shot Imitation Learning in the context of Behavioral Cloning [38, 39, 40].

Algorithm 3 Model-Agnostic Meta-Learning (MAML) [37]

```
Require: Distribution over tasks $p(\mathcal{T})$
  Randomly initialize $\theta$
  while $i = 1, \dots N$ do
      Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
      for all $\mathcal{T}_i$ do
          Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ w.r.t. $K$ examples
          Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
      end for
      Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$
  end while
```

In [38], MAML algorithm was used to prove the effectiveness of Meta-Learning in the context of real robot manipulation, with visual observations, as opposite to [41]. A CNN was trained by following the Algorithm 3, using as loss-function the Mean Squared Error, computed between the predicted action and the ground truth one. For real-robot experiments a dataset of 1300 placing demonstrations (i.e. place an holded object in a target container), containing near to 100 different objects, was collected through teleportation. The trained system was tested by performing the adaptation step on one video demonstration, over 29 new objects, moreover, between the video demonstration and the actual execution, the objects configuration was changed. In this setting the system reached the **90**% of success rate, outperforming baseline methods based on LSTM [41], and contextual network (i.e. a CNN that takes in input the current observation and the image representing the target state).
In [39], the *Domain Adaptive Meta-Learning* algorithm (DAML) was proposed with the goal of learning to infer a policy from a single human demonstration. To achieve it, a two-step algorithm was proposed. In the first-step, called **Meta-Learning step**, given in input, for each task $\mathcal{T}$, a set of human demo $\mathcal{D}^h_\mathcal{T}$ and a set or robot

demo $\mathcal{D}^r_\mathcal{T}$ (Figure 1.6), the *initial policy parameters* θ and the *adaptive loss* parameters ψ are learned, solving the problem in Formula 1.1.

$$\min_{\theta,\psi} \sum_{\mathcal{T}\sim p(\mathcal{T})} \sum_{\mathbf{d^h}\sim D^r_\mathcal{T}} \sum_{\mathbf{d^r}\sim D^h_\mathcal{T}} \mathcal{L}_{BC}(\theta - \alpha\nabla_\theta \mathcal{L}_\psi(\theta, \mathbf{d^h}), \mathbf{d^r}) \tag{1.1}$$

Where the outer loss is $\mathcal{L}_{BC}(\phi, \mathbf{d^r}) = \sum_t log(\pi_\phi(a_t|s_t, o_t))$, and the inner-loss $\mathcal{L}_\psi$, is the learned adaptive loss, which is used during the **Meta-Test step**, where the policy parameters are adapted with gradient descent given in input a video of human demo of a new task $\mathcal{T}$, i.e. $\psi_\mathcal{T} = \theta - \alpha\nabla_\theta \mathcal{L}_\psi(\theta, \mathbf{d^r})$. Experimental evaluation on tasks such as placing, pushing, and pick-and-place, has shown that: **(a)** The system was able to generalize across both new objects and objects configuration starting from only a single human-demonstration. **(b)** A performance degradation was observed in large domain-shift experiments, such as novel backgrounds and different camera view-points. The work proposed in [40] is an extension of DAML, for multi-stage tasks, i.e. tasks that are composed of different motion primitives.

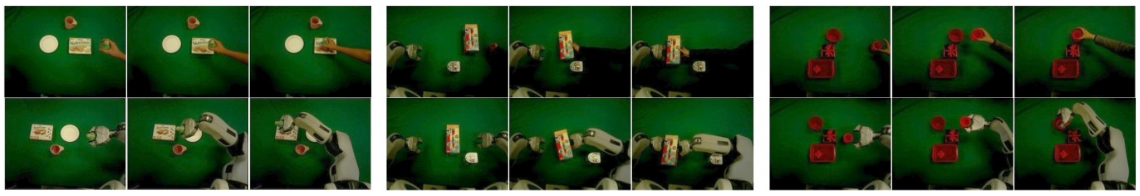


Figure 1.6: Tasks performed in [39]. (Top row) Human demonstration, (Bottom row) robot demonstration. (Left) Placing task, (Middle) pushing task, (Right) pick-and-place task.

Generally speaking, all the works described up to now, consider the task distribution $p(\mathcal{T})$ as composed of one-single task, but with many different variations [38] or different, but related tasks [39]. Very recent methods try to generalize BC to a huge variety of tasks [5, 42], In [5], a large-scale dataset containing **100** diverse manipulation tasks was collected. The demonstrations were collected through expert teleoperation and shared autonomy process (DAgger [32]). The demonstrated tasks were related to pick-and-place, grasp, pick-and-drag, pick-and-wipe, and push skills. The dataset was used to train the network in Figure 1.7. As it can be noted the samples were composed by current robot observation, and a conditioning represented by either a vocal command or a video demo. The idea was that training a conditioned policy over the current observation o_t, and a task representation c_t, $\pi^L(a_t|o_t, c_t)$, it would allow the policy to generalize over new tasks in a zero-shot manner (i.e. without any fine-tuining). Experimental results shown that, over 28 held-out tasks, containing both completely new object, and known objects but in different tasks, an average success rate of **38%** was reached in the easiest setting, with only one distractor and with language conditioning. The success rate dropped to **4%** in the hardest setting with 4 distractors and video conditioning. These bad performance can be justified by the classic training procedure followed, while different advantages may be gained from Meta-Learning algorithms, and the way in which the human video embedding was obtained, through a CNN on a 4x5 matrix of images, with a completely loss of temporal information.

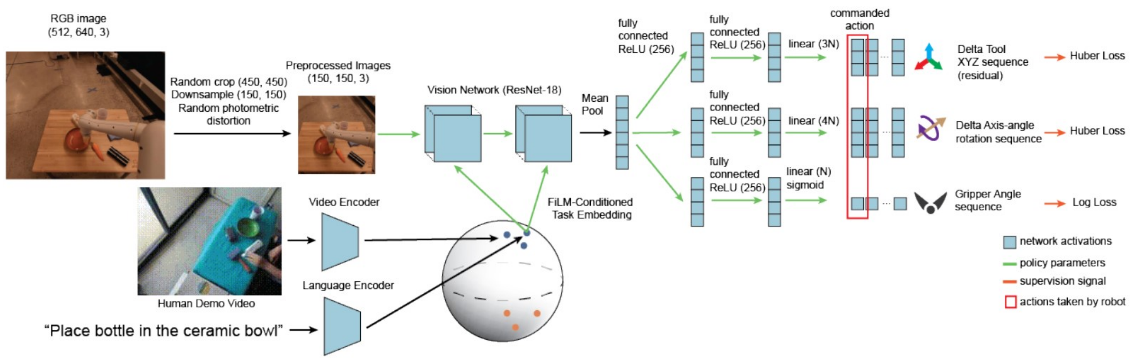


Figure 1.7: Architecture proposed in [5]

Model-Based methods,

Inverse Reinforcement Learning (IRL) [MAX 2]
Inverse Reinforcement Learning or Inverse Optimal Control is a Learning from Demonstration approach that was introduced in [43]. In IRL, starting from a set of expert-demonstrated trajectories, the main goal is to obtain the *Reward function* R, which explains the expert behavior. Once R is found, an RL algorithm is run to obtain the policy π_θ^L (Algorithm 4). An important design choice is related to how

Algorithm 4 Classic feature matching IRL algorithm

```
Require: Dataset of expert trajectories $\mathcal{D} = \{\boldsymbol{\tau}_i^E\}_{i=1}^N$
Require: Reward function $R_\omega$, policy $\pi_\theta^L$
  while policy improves do
      Evaluate the state-action visitation frequency $\mu$ of the current policy $\pi_\theta^L$
      Evaluate the objective function $\mathcal{L}$, w.r.t. $\mu$ and the dataset $\mathcal{D}$
      Update the reward-function parameters $\omega$ based on $\mathcal{L}$
      Update the policy $\pi_\theta^L$ through an RL algorithm, using the updated $R_\omega$
  end while
```

parametrize the reward-function, i.e. a **linear** function of the features [44, 45] or a **non-linear** function of the features [46, 47, 48]. Generally, the IRL algorithms are characterized by the following problems: **(1)** The learning process could be time-consuming and impractical for high-dimensional problems, because of the double-nested optimization procedure. **(2)** The IRL problem is *ill-posed*, since there are different reward-functions that could lead to the same action.
To solve the problem of multiple solutions some constraints to the optimization procedure have to be added, based on the constraint type, the two main IRL procedures can be identified: **(a)** The *Maximum Margin Prediction* (MMP) approach [44, 46]. **(b)** *Maximum Entropy* (Max-Ent) approach [45, 47, 48]. The MMP methods assume that the demonstrated trajectories are optimal and work in a deterministic setting. The aim is to find the cost-function where the reward of the demonstrated trajectories $R(\boldsymbol{\tau}^E)$ is greater than the reward of all the alternative trajectories $R_\omega(\boldsymbol{\tau})$, by a certain margin m, i.e. $\max\limits_{\omega,m} m \;\; s.t. \; R_\omega(\boldsymbol{\tau}^E) \geq \max(R_\omega(\boldsymbol{\tau})) + m$. The main problem with this formulation is that it does not handle the case in which the expert behavior is sub-optimal, leading to an ambiguous notion of margin. To handle this problem the Max-Ent approach has been proposed in [45]. In Max-Ent the goal is to find the reward-function parameters ψ that drives the policy to maximize the entropy, subject to the feature expectation matching, i.e $\max\limits_{\psi} \mathcal{H}(\pi^{r_\psi}) \;\; s.t. \;\; \mathbb{E}_{\pi^{r_\psi}}[\mathbf{f}] = \mathbb{E}_{\pi^*}[\mathbf{f}]$.
The setting based on the Maximum Entropy is the most popular setting in the IRL field since it removes the ambiguous aspects of the previous formulation. In the original work, the reward-function was a linear combination of the features, i.e. $r_\psi = \psi^T\mathbf{f}$. This reward formulation is not suited for high-dimensional feature space, which can require the capability to model non-linear reward structures. In [47], a deep-network was used to model the reward-function. In this work, the NN maps the feature vector $\mathbf{f}$ into the reward value, and trained according to the Maximum-Entropy setting. Experimental results have shown that the NN ability

to approximate highly nonlinear reward functions is essentially to successfully solve tasks in high-dimensional discrete state spaces. A generalization to continuous state-space was proposed in [48]. In particular, [48] faced the problem of learning a cost function in a high-dimensional continuous state space, with **unknown dynamics**. Indeed, starting from the exponential trajectory distribution $p(\tau) = \frac{1}{Z}\, exp(-c_\theta(\tau))$, the main difficulty is the estimation of the partition function Z, needed to compute the negative log-likelihood loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_{\tau_i^E \in D_{demo}} c_\theta(\tau_i^E) + \log(Z)$. Since, the dynamic is unknown the idea was to estimate the partition function through the trajectories obtained from the current policy rollouts, with the idea that during the learning of the cost function the current policy drives the distribution towards regions where samples are more useful. Starting from these consid-

Algorithm 5 Guided-Cost-Learning Algorithm [48]

```
Require: Initial controller $q_k(\tau)$
  for i = 1, ..., N do
      Generate $D_{traj}$ from $q_k(\tau)$
      $\mathcal{D}_{samp} \leftarrow \mathcal{D}_{samp} \bigcup \mathcal{D}_{traj}$
      Update the reward-function parameters, using $\mathcal{D}_{samp}$ and $\nabla_\theta \mathcal{L}(\theta)$
      Update the LQR-FLM [49] controller $q_{k+1}(\tau) \leftarrow q_k(\tau)$, using $\mathcal{D}_{traj}$
  end for
```

erations, the Algorithm 5 was proposed. The controller was obtained as a result of a Linear-Quadratic Regulator (LQR) optimization procedure, which returns a *linear-gaussian policy*, $\pi(a_t|s_t) = \mathcal{N}(K_t s_t + k_t, S_t)$, which optimizes a *quadratic-cost function*, $c(s_t, a_t) = \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T C_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T c_t + cc_t$,under the assumption of *linear-gaussian dynamics* $s_{t+1} \sim P(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \Sigma_t)$. The cost-function was parameterized according to a Neural Network, and experiments performed on real-robot manipulation tasks such as dish placement and pouring proved the effectiveness of the proposed method, outperforming the classic Max-Ent method.
Recent method [50] tried to move IRL towards more complex learning scenario, such as learning a reward function from video-demonstrations. In [50] the architecture in Figure 1.8 was proposed. The goal was to obtain the parameters of a cost function $C_\Psi(\hat{\boldsymbol{\tau}}, z_{goal})$, such that it was possible to derive a sequence of actions by minimizing the cost-function itself, $\mathbf{a}_{new} = \mathbf{a} - \eta \nabla_a C_\Psi(\hat{\boldsymbol{\tau}}, z_{goal})$, Different cost functions have been proposed, but all with the same objective to reduce the distance between the current predicted keypoints and the goal keypoints configuration. The trajectory $\hat{\boldsymbol{\tau}}$ is a a sequence of predicted states $[\hat{s}_1, \ldots, \hat{s}_T]$, result of a learned dynamic model, $\hat{s}_{t+1} = f_{dyn}(\hat{s}_t, a_t)$. Experimental results have shown that it is possible to learn a cost-function starting from human/robot video demonstrations. However, the proposed setting was tested on a very simple reaching task, proving how a lot of work has to be made in order to prove the effectiveness or ineffectiveness of IRL in complex real-robot manipulation tasks, starting from video demonstrations.

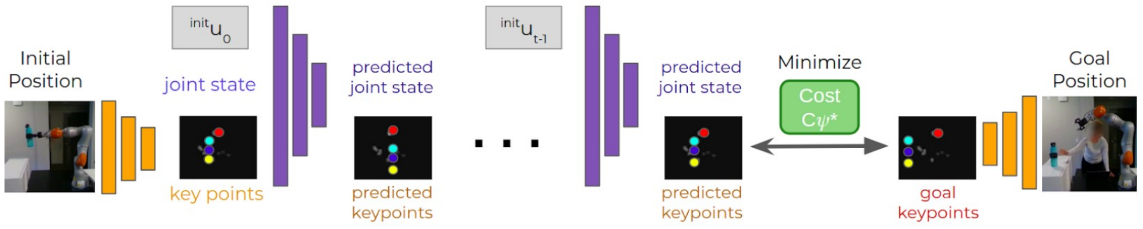


Figure 1.8: Architecture proposed in [50]

Generative Adversarial Imitation Learning (GAIL) [MAX 5]
Generative Adversarial Imitation Learning was proposed for the first time in [51], with the idea to improve the IRL setting, which is expensive to run, because of the double-nested optimization procedure. The authors in [51], starting from a general Max-Ent formulation (Formula 1.2), obtained a characterization of the learned policy (Formula 1.3), where $\psi(c)$ is a cost-regularizer, $\psi^*(c)$ is its conjugate, and ρ_π is the *occupancy measure*, i.e. the distribution of state-action pairs that the agent encounters when navigating the environment with policy π. The interpretation of Formula 1.3 is that the $\psi - regularized$ IRL finds a policy whose occupancy measure is similar to the expert's one, measured by ψ^*. The next-step was to choose an appropriate regularization function. In particular, by choosing the regularizer in Formula 1.4, the conjugate in Formula 1.5 can be obtained, which is the classic Adversarial-Learning Loss, where the current policy π^L plays the role of GAN generator, and D is the GAN discriminator, which has to distinguish between state-action pairs generated either by the expert-policy or by the current policy.

$$IRL_\psi(\pi^E) = \underset{c\in R^{S\times A}}{arg\ max} - \psi(c) + (\min_{\pi^L\in\Pi} - \mathcal{H}(\pi^L) + \mathbb{E}_{\pi^L}[c(s,a)]) - \mathbb{E}_{\pi^E}[c(s,a)] \tag{1.2}$$

$$RL \circ IRL_\psi(\pi^E) = \underset{\pi^L\in\Pi}{arg\ min} - \mathcal{H}(\pi^L) + \psi^*(\rho_{\pi^L} - \rho_{\pi^E}) \tag{1.3}$$

$$\psi_{GA}(c) = \begin{cases} \mathbb{E}_{\pi^E}[g(c(s,a))] & if\ c<0 \\ +\infty & otherwise \end{cases},\ g(x) = \begin{cases} -x - log(1-e^x) & if\ c<0 \\ +\infty & otherwise \end{cases} \tag{1.4}$$

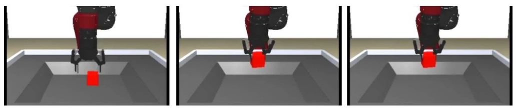
$$\psi^*_{GA}(\rho_{\pi^L} - \rho_{\pi^E}) = \underset{D\in(0,1)^{S\times A}}{max} \mathbb{E}_{\pi^L}[\log(D(s,a))] + \mathbb{E}_{\pi^E}[\log(1-D(s,a))] \tag{1.5}$$

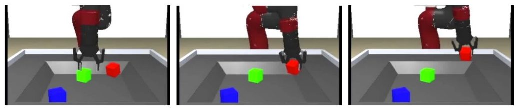
Based on these considerations the Algorithm 6 has been proposed. The algorithm comprises two fundamental steps, the first related to the Discriminator's parameter update and the second related to the policy's parameter update. Since GAIL has proven to be more effective than classic IRL algorithm [45], subsequent works have focused either on improving the sample efficiency, by replacing the model-free on-policy TRPO algorithm, with an off-policy RL algorithm, such as in [52], or by modifying the reward function in input to the RL algorithm [53, 54]. All

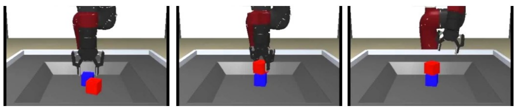
Algorithm 6 Generative Adversarial Imitation Learning Algorithm

```
Require: Expert Trajectories $\tau^E \sim \pi^E$, initial policy $\pi^L_\theta$, discriminator $D_\omega$
  for $i = 1, \dots, N$ do
      Sample trajectories, $\tau^L_i \sim \pi^L_\theta$
      Update Discriminator, $\hat{\mathbb{E}}_{\tau^L_i}\left[\nabla_\omega \log(D_\omega(s,a))\right] + \hat{\mathbb{E}}_{\tau^E}\left[\nabla_\omega \log(1 - D_\omega(s,a))\right]$
      Update Policy $\pi_\theta$, with TRPO [55], and cost-function $C(s,a) = \log(D_\omega(s,a))$
  end for
```

the cited methods have reported promising results on control tasks in a simulation environment [56], but they worked in a low-dimensional state-space, indeed when the GAIL method was adapted to work with a high-dimensional state-space, like in [57, 58, 59, 60], it shown very poor results. In particular, with respect to the Adversarial Imitation Learning setting, works of interest are [59, 60]. In [59], the authors focused on solving the **casual-confusion** problem. This problem occurs when the discriminator, during the learning process, focuses on task-irrelevant features between expert and policy generated transitions, this causes the rewards to become uninformative. To reduce the casual-confusion problem, in [59] two elements have been proposed: (**1**) A regularization term, with the aim to make the discriminator **unable** to distinguish between constraining sets I_E and I_A. The constraining sets are composed of expert and agent observations, such that a sample can belong either to I_E or I_A, based on spurious features. (**2**) An early-stopping policy, Actor Early-Stopping (AES), that restarts the episode if the discriminator score at the current step exceeds the median score of the episode so far for $T_{patience}$ consecutive steps.. The regularization term was defined as $accuracy(\mathcal{I}_E, \mathcal{I}_A) = \frac{1}{2}\,\mathbb{E}_{s \in \mathcal{I}_E}\left[\mathbf{1}_{D_\omega \geq \frac{1}{2}}\right] + \frac{1}{2}\,\mathbb{E}_{s \in \mathcal{I}_A}\left[\mathbf{1}_{D_\omega < \frac{1}{2}}\right]$. In the end, the discriminator parameters were the result of $\underset{\omega}{max}\ G_\omega(s_E, s_A) - \mathbf{1}_{accuracy(\hat{s}_E, \hat{s}_A) \geq \frac{1}{2}} G_\omega(\hat{s}_E, \hat{s}_A)$, where G_ω is the discriminator binary cross-entropy, $\hat{s}_E \in \mathcal{I}_E$, and $\hat{s}_A \in \mathcal{I}_A$. The proposed system was tested on 4 tasks (Figure 1.9). An agent was trained on each single task, according to the Distributed Distributional Deterministic Policy Gradients (D4PG) [61] RL algorithm. Experimental results have shown how the proposed system overcomes the GAIL [51] baseline, both in setting with spurious features and without spurious features (Figure 1.10).


(a) Block lifting


(b) Block lifting with distractors


(c) Block stacking

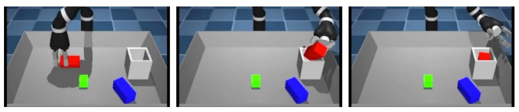
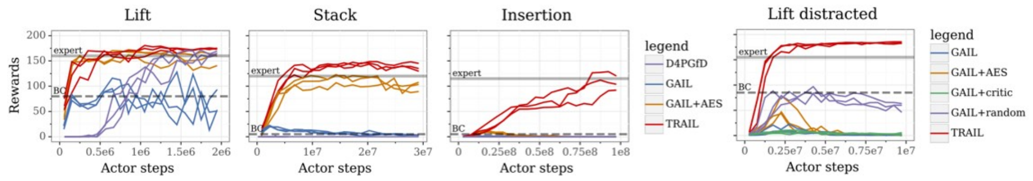
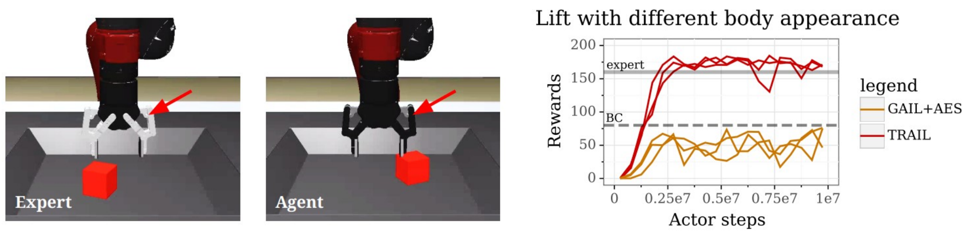

(d) Block insertion with distractors

Figure 1.9: Tasks solved in [59]



(a) Experimental results on tasks without spurious features



(b) Experimental results on task with spurious features

Figure 1.10: Experimental results in [59]

The authors of [60] made a step towards a more data efficient Adversarial Imitation Learning method. Indeed, they leveraged the idea of using a model-based approach in the context of high-dimensional state space. Still, instead of generating a dynamic model in the image space, the proposed method is based on encoding the observations defined in the image space into a corresponding latent space characterized by vectors of small dimensions. Then subsequently go on to learn a dynamic model in that space. In context the learning procedure is based on three main steps: **(1)** Learn the *Latent Dynamic Model*, $(\hat{\mathcal{U}}_\beta, \hat{\mathcal{T}}_\beta, q_{beta})$, by maximizing the Expected Lower Bound (Formula 1.6), where $\hat{\mathcal{U}}_\beta$ is the decoder, q_{beta} is the encoder, and $\hat{\mathcal{T}}_\beta$ is the transition model. **(2)** Train a *discriminator*, D_θ, by minimizing the Adversarial Loss function (Formula 1.7). **(3)** Train a *policy* π^L_θ, by maximizing the Value function (Formula 1.8).

$$\underset{\beta}{max}\ \mathbb{E}_{q_\beta}\left[\sum_t \log(\hat{\mathcal{U}}_\beta(s_t|z_t)) + \mathbb{D}_{KL}(q_\beta(z_t|s_t, z_{t-1}, a_{t-1})||\hat{\mathcal{T}}_\beta(z_t|z_{t-1}, a_{t-1}))\right] \tag{1.6}$$

$$\underset{\theta}{min}\ \mathbb{E}_{(z,a)\sim\rho^E(z,a)}\left[-log(D_\theta(z,a))\right] + \mathbb{E}_{(z,a)\sim\rho^{\pi^L}_{\hat{\mathcal{T}}}}\left[-\log(1 - D_\theta(z,a))\right] \tag{1.7}$$

$$\underset{\pi^L_\theta}{max}\ V^K_{\theta,\beta}(z_t) = \underset{\pi^L_\theta}{max}\ \mathbb{E}_{\pi^L_\theta,\hat{\mathcal{T}}_\beta}\left[\sum_{\tau=t}^{t+K-1} \gamma^{\tau-t}\log(D_\theta(z^{\pi^L_\theta}_\tau, a^{\pi^L_\theta}_\tau)) + \gamma^K V_\beta(z^{\pi^L_\theta}_{t+K})\right] \tag{1.8}$$

With this learning setting the proposed system outperforms previous works such as [58, 52] both in terms of data-efficiency and overall performance, on a set of control tasks (Figure 1.11). Generally speaking, the Generative Adversarial Imitation Learning has shown very promising performance in simulated control tasks and simulated robot manipulation tasks, even in complex high-dimensional state-space. However, it is not so clear, how these methods could perform in real-world robotic manipulation tasks, in terms of data-efficiency, generalization capability, and safety during real-world interactions.

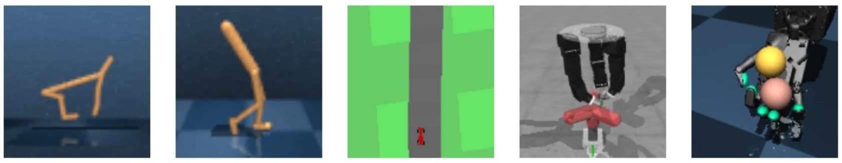


Figure 1.11: Control tasks solved in [60]. From left to right: cheetah run, walker walk, car racing, claw rotate, baoding balls

Learning from Observation (LfO) [MAX 5]
The goal of *Learning from Observation* is to learn a policy by leveraging **state-only-demonstration**. This approach has gained attention in recent years because it theoretically allows a robotic system to be programmed as naturally as possible. In fact, in the most promising setting, a robotic system should be able to reproduce a task by observing a human or another robot performing it, without having access to the action performed, as is the case in all the methods described so far. In designing an LfO system there are at least three aspects to consider: **(1)** In the case the demonstrator has a different embodiment with respect to the imitator, how to solve the embodiment mismatch? **(2)** In the case the demonstrator viewpoint differs from the imitator one, how to solve the correspondence problem between the two different viewpoint? **(3)** Once the problems related to the perception subsystem are solved, how is the policy π^L obtained?.

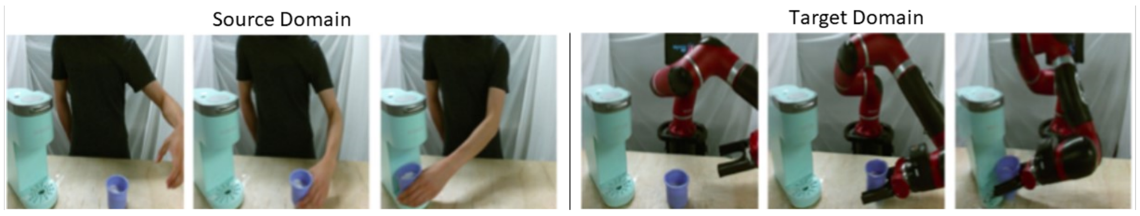


Figure 1.12: Representation of embodiment mismatch problem. (Left) The source domain represented by a video of human performing a task. (Right) The target domain, represented by the robot that executes the observed task.

First of all, the points **(1)** and **(2)** are going to be answered. Regarding, the embodiment mismatch, this may happen, for example, when the video shows a human performing a task, but the goal is to train a robot (Figure 1.12).To solve this problem, in [62, 63, 64], some sort of image-to-image translation is performed, while in [65] the *Temporal-Cycle Consistency* (TCC) [66] was leveraged. In particular, in [62, 64], the Cycle-GAN [67] was used to translate image from the source domain (human image) into the corresponding target domain (robot image). The need for an unsupervised image-to-image translation architecture such as the Cycle-GAN arises from the fact that there are not paired images. The network learns two mappings, the first from the source to the target domain $G : X \to Y$, the second from the target to the source domain $F : Y \to X$. The dataset for the source domain was composed of human demonstrations as well as a small amount of "random" data, in which the human moves around the scene but does not specifically attempt the task, while for the target domain, it consists of robot images executing randomly sampled actions in a few different settings. In [63] a similar approach was proposed, but the MUNIT [68] architecture was adopted for the image-to-image translation. With respect to the mentioned works, in [65] a completely different approach has been proposed. Indeed, starting from a set of video demonstrations characterized by both different length and demonstrator embodiment, the TCC approach was used to learn an encoder able to map demonstration frame into the corresponding embodiment-independent embedding. The idea behind TCC was to learn a correspondence between frames of different videos representing the same overall action (Figure 1.13).

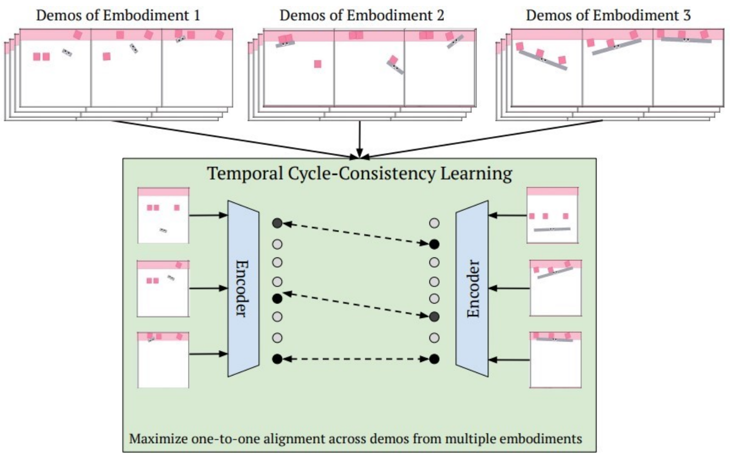
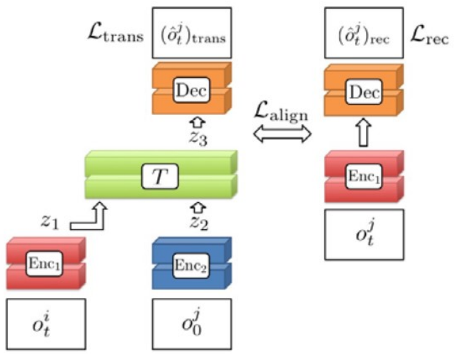

Demos of Embodiment 1
Demos of Embodiment 2
Demos of Embodiment 3
Temporal Cycle-Consistency Learning
Encoder
Encoder
Maximize one-to-one alignment across demos from multiple embodiments

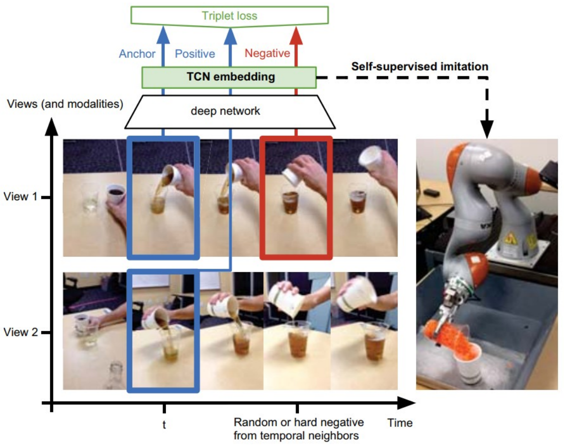
Figure 1.13: Temporal-Cycle Consistency representation, used to learn an embodiment-agnostic encoder in [65]

The correspondence problem between different viewpoints was faced by [69, 57]. In [69] a CNN was trained by means of a *Triplet-Loss* [70]. The idea was to train a network able to predict an embedding independent from the view-point, but which contains only task relevant features. To reach this objective the network has to produce an embedding $f(x)$, such that $||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2$ $\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T}$, where $\mathcal{T}$ is the set of all possible triplets in the dataset. This means that embedding produced by samples coming from different viewpoints, but that share the same time-stamp should have a similar embedding, while embedding produced by samples coming from the same viewpoint, but in different time-stamp should have a different embedding. In [57], a different approach was used, indeed, a *context translation problem* was solved by means of an Encoder-Decoder architecture (Figure 1.14a). The proposed architecture was trained on pairs of demonstrations, $\mathcal{D}_i = [o_0^i, o_1^i, \ldots, o_T^i]$ and $\mathcal{D}_j = [o_0^j, o_1^j, \ldots, o_T^j]$ composed of visual observations. Samples in D_i comes from the source context ω_i, while samples in $\mathcal{D}_j$ comes from the target context ω_j. The model must output the observations in D_j conditioned on $\mathcal{D}_i$, and the first observation o_0^j for the target domain. As it will be explained next, the output of both the Time-Contrastive and the Context-Translation network can be used to obtain an engineered reward function.

Regarding the way how the policy can be obtained, two approaches can be iden-


$\mathcal{L}_{\text{trans}}$
$(\hat{o}_t^j)_{\text{trans}}$
$(\hat{o}_t^j)_{\text{rec}}$
$\mathcal{L}_{\text{rec}}$
Dec
Dec
$\mathcal{L}_{\text{align}}$
z_3
T
Enc$_1$
z_1
z_2
Enc$_1$
Enc$_2$
o_t^i
o_0^j
o_t^j

(a) *Context-Translation network, proposed in [57]*



(b) *Time-Contrastive network, proposed in [69].*

Figure 1.14: Examples of how the mismatch between demonstrator viewpoint and learner viewpoint can be handled.

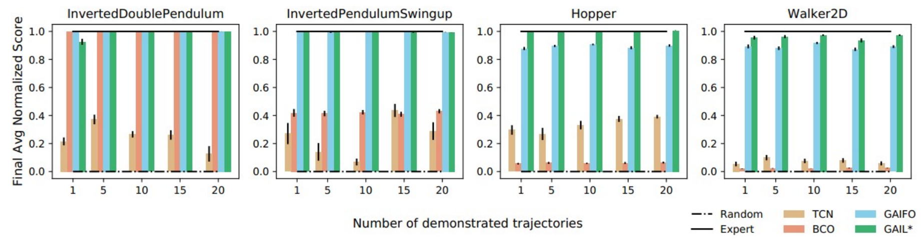
tified, that are Model-Based and Model-Free.
Among **Model-Free** methods, a classification must be made between methods based on *Reward Engineering* or *Adversarial Learning*.
Methods based on Reward Engineering are [69, 63, 65], in which a hand-designed reward function is used to train a RL agent. In particular, the reward functions obtained in the cited works leverage some sort of *Feature Tracking*. In [69] the reward function was $R(o_t^l) = -||Enc_1(o_t^l) - \frac{1}{n}\sum_i^n F(o_t^i, o_0^l)||_2^2 - w_{rec}(||o_t^l - \frac{1}{n}\sum_i^n M(o_t^i, o_0^l)||_2^2)$. The first term is the classic Feature Tracking reward function, where the goal is to minimize the Euclidian Distance between the encoding of the current learner observation o_t^l and the encoding of the demonstration in the learner context, the second term has the aim to penalize the policy for experiencing observations that differ from the translated observation. In [69] the reward function was $R(\mathbf{v}_t, \mathbf{w}_t) = -\alpha||\ \mathbf{w}_t - \mathbf{v}_t\ ||_2^2 - \beta\sqrt{\gamma + ||\ \mathbf{w}_t - \mathbf{v}_t\ ||_2^2}$, where $\mathbf{v}_t$ is the TCN embedding of the video demonstration at timestep t, while $\mathbf{w}_t$ is the TCN embedding produced by the robot observation (Figure 1.14b). In [63], first a keypoint-representation is obtained from two consecutive observations, z_t, z_{t+1}, and from each frame of the translated demonstration video, z^E. The reward is then computed as $R(z_t, z_{t+1}, z^E) = -\lambda_1 \min_p ||z_t - z_p^E|| - \lambda_2 \min_p ||(z_{t+1} - z_t) - (z_{p+1}^E - z_p^E)||$. In [65], the reward function was defined as $R(s_t) = -\frac{1}{k}\ ||\phi(s_t) - g||_2^2$, where g is the mean embedding of the last frame of all the demonstration videos in the dataset. Experimental results, on simulation data proved that the proposed method can be used to learn tasks from cross-embodiment demonstrations, outperforming baseline [69] in terms of both sample efficiency and performance.
Regarding the methods based on Adversarial Learning, there are [71, 72]. Obviously, these methods are strictly related to Generative Adversarial Imitation Learning setting, however, with respect to the methods presented in GAIL paragraph (pag. 16) these methods do not assume access to the demonstrator action. Indeed the pre-

liminary work [71] was an extension of GAIL [51], where the goal was to prove that the Adversarial Learning Setting can be effectively used even without the action information. To prove this hypothesis, the authors performed a series of experiments in simulation (Figure **??**), where the same RL policy was trained in two contexts, the first, where the Discriminator had access to the (state,action) pair, the second where the Discriminator had access to the state only. The results obtained did not show a substantial difference between the two settings.

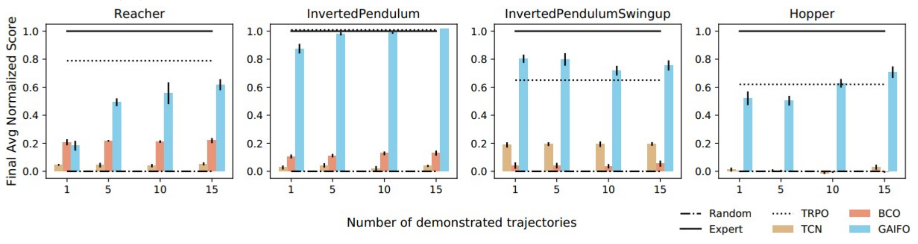
Algorithm 7 GAIfO algorithm [72]

```
Require: Initial policy $\pi^L_\phi$, Initial Discriminator $D_\theta$
Require: State-only expert demonstration trajectories $\tau^E = \{(s, s')\}$
  while Policy Improves do
      Execute $\pi^L_\phi$ and collect state transitions $\tau^L = \{(s, s')\}$
      Update $D_\theta$, with $\mathcal{L}_{D_\theta} = -\ (\ \mathbb{E}_{\tau^L}[\log(D_\theta(s, s'))] + \mathbb{E}_{\tau^E}[\log(1 - D_\theta(s, s'))]\ )$
      Update $\pi^L_\phi$, with reward $r_{\pi^L_\phi} = -\ (\ \mathbb{E}_{\tau^L}[\log(D_\theta(s, s'))]\ )$
  end while
```

The next remarkable work was proposed by the authors of [72], that formalized the *GAIfO* algorithm, (Algorithm 7), that is an extension to state-only demonstration of the GAIL [51]. The proposed algorithm was used to train a network to solve tasks in simulation environments [56], with both low-dimensional state representation, and visual-state representation. Results with respect to the number of demonstrated trajectories are reported in Figure 1.15. As it can be noted from the results, GAIfO outperforms previous observation based methods [69, 73], in a setting with a low number of expert trajectories. The main drawback of GAIfO is the high number of environmental interactions needed to learn a policy, since the model-free TRPO [55] algorithm was used to train the policy. This problem was solved by DEALIO [74], which replaced the model-free algorithm with PILQR [75] the model-based RL algorithm reported next.
Among the **Model-Based** methods a further classification has to be made between methods that obtain the *Inverse Dynamic Model* and methods that obtain the *Forward Dynamic Model*. The former, given in input a transition (s_t, s_{t+1}), obtain a function M, able to map state transition to action i.e. $a_t = M(s_t, s_{t+1})$. While the latter, given in input a state-action pair (s_t, a_t), aim to learn a function F able to generate the next state s_{t+1}, i.e. $s_{t+1} = F(s_t, a_t)$.
Methods that obtain the Inverse Dynamic Model are [76, 73, 77, 78]. In [76] the goal was to obtain a system capable of tying the knot to a rope. To achieve it, a self-supervised learning approach was used to train a CNN, that, taken in input a pair of images (I_t, I_{t+1}) representing two successive rope states, is able to obtain the action to perform in order to reach the state I_{t+1} from I_t. To train the CNN, a dataset composed of 30K tuples (I_t, a_t, I_{t+1}) was collected by means of an exploratory policy. Authors in [73] proposed a general approach, depicted in Figure 1.16, and composed by the two main parts, the learned Inverse Dynamic Model M_θ, and the learned policy π_ϕ. In its general form, the learning procedure is an iterative procedure, where the model M^i_θ is updated by maximizing the probability $p_\theta(a_t|s_t, s_{t+1})$,



(a) Experimental results in low-dimensional state space



(b) Experimental results in high-dimensional state space

Figure 1.15: Experimental results reported in [72].

where the tuples (s_t, a_t, s_{t+1}) are collected by running the current policy. Once the dynamic model is updated then it is used to infer the action $\tilde{a}_t$ given the demonstrations. At the end, since the policy has access to both state and action information, classic BC learning can be run, optimizing the policy parameters through maximum-likelihood estimation $\phi^* = \underset{\phi}{argmax} \prod_{i=0}^{N} \pi_\phi^L(\tilde{a}_i|s_i)$. In [77], the same approach as in [73], but the agent's policy was trained according to a linear combination of Behavioral Cloning and Advantage Actor Critic (A2C) objective function [79], i.e. $\mathcal{L}_\theta^{hyb} = \mathbb{E}_{s,a}[A(s)\log(a|s;\theta) + \alpha\, \mathcal{H}(\pi^L(.|s))] + \mathbb{E}_{(\hat{s}_t,\hat{s}_{t+1})\sim D}[\log(\pi^L(M(\hat{s}_t, \hat{s}_{t+1})|\theta)]$. The assumption that the access to reward signal has been made, which can reduce its applicability in real-robot manipulation tasks. In [78], the work in [80] was extended to state-only demonstration, and the *State-Only Imitation Learning* (SOIL) algorithm was proposed. The context is the one of complex dexterous manipulation, i,e, a simulated humanoid hand must be able to perform tasks such as object reallocation, tool use, in-hand manipulation, and door opening. A MLP network was trained to represent the Inverse Dynamic Model, by minimizing the L2-loss, given in input the action performed during the policy rollout. Then, the policy was updated according to *Demo Augmented Policy Gradient* (DAPG) [80], adapted for state-only demonstration, i.e. $g_{SOIL} = g + \lambda_0\ \lambda_1^k\ \sum_{(s,\tilde{a}_t \in D'} \nabla_\theta \log \pi_\theta^L(\tilde{a}_t, s)$, where g is the Natural Policy Gradient term. The idea is to leverage the demonstrations at the beginning of the training, then exploit the RL algorithm to improve the behavior itself. Indeed, experiments performed in simulation, proved that, with respect to pure RL, the proposed method converges faster, and producing more human-like behaviors.
Methods that obtain the Forward Dynamic Model are [62, 74]. In [62], once the human video demonstration has been translated into the corresponding robot video, the policy was learned according to the model-based RL algorithm SOLARIS [81], which allows to obtain a controller optimized according to LQR procedure, starting

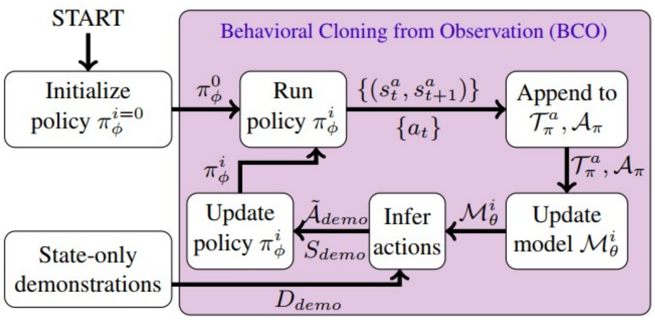
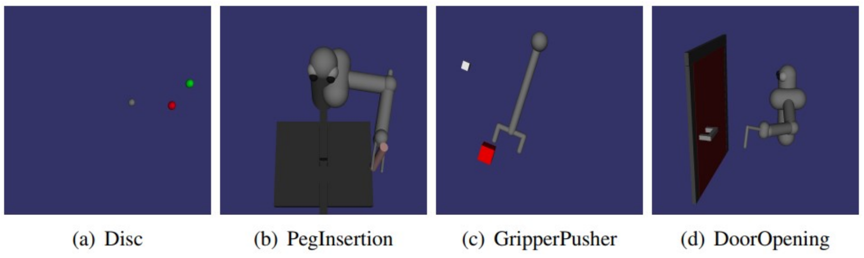

START
Behavioral Cloning from Observation (BCO)
Initialize policy $\pi_\phi^{i=0}$
π_ϕ^0
Run policy π_ϕ^i
$\{(s_t^a, s_{t+1}^a)\}$
$\{a_t\}$
Append to $\mathcal{T}_\pi^a, \mathcal{A}_\pi$
$\mathcal{T}_\pi^a, \mathcal{A}_\pi$
π_ϕ^i
Update policy π_ϕ^i
$\tilde{A}_{demo}$
S_{demo}
Infer actions
$\mathcal{M}_\theta^i$
Update model $\mathcal{M}_\theta^i$
State-only demonstrations
D_{demo}

Figure 1.16: Representation of the learning procedure proposed by [73]

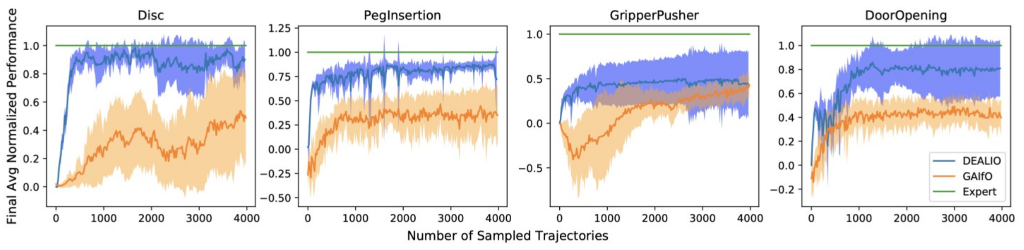
from high-dimensional observations such as images. The idea was to optimize the policy in a low-dimensional high-regularized *latent space*, generated according to Variational Inference [82]. Starting from a sequence of observation-action a Global Dynamic Model over latent trajectory is obtained. Then, given the Latent Dynamic Model, a Linear-Gaussian Controller is obtained through LQR-FLM [49]. Real world robotic experiments, shown that with **2 hours** of robot interaction it was possible to outperform previous works such as [69, 73] and classic BC algorithm, on tasks such as "coffee making" (Figure 1.12) and cup-retrieving (i.e. the robot has to take a cup from a closed drawer). In [74] the sample-inefficiency problem of GAIfO [72] was addressed. The idea was to exploit the adversarial learning setting with state-only demonstration, which has shown promising results (Figure 1.15), and combining it with a more data-efficient RL algorithm, such as PILQR [75]. The core of PILQR is the LQR optimization procedure (pag. 15). In order to use this framework, the linear-gaussian dynamic model was fitted starting from the current policy rollouts, then, to obtain a quadratic cost function as needed by LQR, the dynamic model was used to express the modified discriminator output (Formula 1.9) as function of the pair (s_t, a_t).

$$D_\theta(s_t, s_{t+1}) = \frac{1}{2}\begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T C^{ss}(s_t, s_{t+1})\begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix} + \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T c^{ss}(s_t, s_{t+1}) \tag{1.9}$$

Experiments performed in simulation with low-dimensional state space have shown promising results (Figure 1.17b), in terms of sample-efficiency with respect to the GAIfO baseline. However, improvements would be made in order to: **(1)** reduce the variance, in order to make the learning process more reliable, **(2)** increase the overall performance, **(3)** adapt the algorithm to work with real-world robot manipulation tasks.



(a) Control Tasks solved in [74]



(b) Performance of DEALIO [74] compared against GAIfO [72], with respect to the number of trajectories sampled during the learning process.

Figure 1.17: DEAILO: (1.17a) Control Tasks, (1.17b) Performance Level

1.2.4 Summary

Chapter 2

Research Plan

Based on the State of the Art, presented in Chapter 1.2, the following research-gaps can be identified ...., and the following questions can be made...

Chapter 3

Other Activities

3.1 Attended Courses

3.1.1 Compulsary courses

The mandatory courses supported were:

- Scientific Writing and Publishing. The course is focused on presenting the metrics used to assess the goodness of author, paper, and scientific journal. The course ended with the review and comparison of two scientific articles, brought as examples of a well-written and a poorly written article, according to the guidelines presented during the course.
- Advanced Machine Learning. The course aims to present advanced aspects, both theoretical and practical, related to modern Machine Learning techniques. It ended with a project in which a Deep Network was designed, trained and validated to solve the facial attribute multi-class classification problem.
- Optimization Techniques for Engineers. The course aims to present the theoretical aspects needed to model, solve, and evaluate real world problems by leveraging optimization techniques. It ended with a project in which the presented optimization techniques have been used to formulate and solve a trajectory planning problem.
- Computational Paradigms for problem solving ...
- English Language C1.

3.1.2 Additional courses

I decided to add the following twp exams to my path:

- Mobile Robots for Critical Mission. The course, related to the Master Degree in Computer Engineering, aims to provide the architectural, methodological, and design elements for the realization of intelligent robots capable of moving autonomously in indoor environment. The course ended with the design, implementation, and validation of a software that allows a mobile robot platform to move autonomously, given the desired start and end points.

- Ottimizzazione. The course aims to present the theoretical aspects related to both continuous and integer linear programming problems, combined with the respective solving algorithms.

Bibliography

[1] Softbank Robotics, "Pepper robot."

[2] Franka Emika, "Franka emika panda robot."

[3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, pp. 651–673, PMLR, 2018.

[4] T. Anne, J. Wilkinson, and Z. Li, "Meta-learning for fast adaptive locomotion with uncertainties in environments and robot dynamics," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4568–4575, IEEE, 2021.

[5] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*, pp. 991–1002, PMLR, 2022.

[6] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine learning*, vol. 84, no. 1, pp. 137–169, 2011.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[10] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[11] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[14] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[15] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[16] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 6325–6331, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[17] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4693–4700, IEEE, 2018.

[18] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, IEEE, 2018.

[19] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Autonomous Robots*, vol. 41, no. 3, pp. 593–612, 2017.

[20] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 362–369, 2019.

[21] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 1395–1476, 2021.

[22] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *Conference on Robot Learning*, pp. 879–893, PMLR, 2018.

[23] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1048–1055, IEEE, 2019.

[24] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning*, pp. 1678–1690, PMLR, 2022.

[25] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4613–4619, IEEE, 2021.

[26] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, IEEE, 2018.

[27] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and opportunities," *arXiv preprint arXiv:2106.12177*, 2021.

[28] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, vol. 15, 2002.

[29] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[30] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[31] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, JMLR Workshop and Conference Proceedings, 2010.

[32] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, JMLR Workshop and Conference Proceedings, 2011.

[33] M. Laskey, C. Chuck, J. Lee, J. Mahler, S. Krishnan, K. Jamieson, A. Dragan, and K. Goldberg, "Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 358–365, IEEE, 2017.

[34] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083, IEEE, 2019.

[35] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv:2012.06733*, 2020.

[36] E. Chisari, T. Welschehold, J. Boedecker, W. Burgard, and A. Valada, "Correct me if i am wrong: Interactive learning for robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3695–3702, 2022.

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

[38] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on robot learning*, pp. 357–368, PMLR, 2017.

[39] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.

[40] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv preprint arXiv:1810.11043*, 2018.

[41] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[42] Z. Mandi, F. Liu, K. Lee, and P. Abbeel, "Towards more generalizable one-shot visual imitation learning," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2434–2444, IEEE, 2022.

[43] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.

[44] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736, 2006.

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

[46] N. D. Ratliff, D. Silver, and J. A. Bagnell, “Learning to search: Functional gradient techniques for imitation learning,” *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.

[47] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.

[48] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse optimal control via policy optimization,” in *International conference on machine learning*, pp. 49–58, PMLR, 2016.

[49] S. Levine and P. Abbeel, “Learning neural network policies with guided policy search under unknown dynamics,” *Advances in neural information processing systems*, vol. 27, 2014.

[50] N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier, “Model-based inverse reinforcement learning from visual demonstrations,” in *Conference on Robot Learning*, pp. 1930–1942, PMLR, 2021.

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

[52] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, “Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning,” in *International Conference on Learning Representations*, 2018.

[53] J. Fu, K. Luo, and S. Levine, “Learning robust rewards with adverserial inverse reinforcement learning,” in *International Conference on Learning Representations*, 2018.

[54] S. K. S. Ghasemipour, R. Zemel, and S. Gu, “A divergence minimization perspective on imitation learning methods,” in *Conference on Robot Learning*, pp. 1259–1277, PMLR, 2020.

[55] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[56] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[57] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, “Imitation from observation: Learning to imitate behaviors from raw video via context translation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125, IEEE, 2018.

[58] S. Reddy, A. D. Dragan, and S. Levine, “Sqil: Imitation learning via reinforcement learning with sparse rewards,” in *International Conference on Learning Representations*, 2019.

[59] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarejo, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang, "Task-relevant adversarial imitation learning," in *Conference on Robot Learning*, pp. 247–263, PMLR, 2021.

[60] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn, "Visual adversarial imitation learning using variational models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3016–3028, 2021.

[61] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.

[62] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "Avid: Learning multi-stage tasks via pixel-level translation of human videos," *arXiv preprint arXiv:1912.04443*, 2019.

[63] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg, "Learning by watching: Physical imitation of manipulation skills from human videos," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7827–7834, IEEE, 2021.

[64] J. Li, T. Lu, X. Cao, Y. Cai, and S. Wang, "Meta-imitation learning by watching video demonstrations," in *International Conference on Learning Representations*, 2021.

[65] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "Xirl: Cross-embodiment inverse reinforcement learning," in *Conference on Robot Learning*, pp. 537–546, PMLR, 2022.

[66] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1801–1810, 2019.

[67] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[68] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 172–189, 2018.

[69] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[70] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[71] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, "Learning human behaviors from motion capture by adversarial imitation," *arXiv preprint arXiv:1707.02201*, 2017.

[72] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[73] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

[74] F. Torabi, G. Warnell, and P. Stone, "Dealio: Data-efficient adversarial learning for imitation from observation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2391–2397, IEEE, 2021.

[75] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *International conference on machine learning*, pp. 703–711, PMLR, 2017.

[76] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2146–2153, IEEE, 2017.

[77] X. Guo, S. Chang, M. Yu, G. Tesauro, and M. Campbell, "Hybrid reinforcement learning with expert state sequences," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3739–3746, 2019.

[78] I. Radosavovic, X. Wang, L. Pinto, and J. Malik, "State-only imitation learning for dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7865–7871, IEEE, 2021.

[79] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[80] 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," in *Proceedings of Robotics: Science and Systems*, (Pittsburgh, Pennsylvania), June 2018.

[81] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International Conference on Machine Learning*, pp. 7444–7453, PMLR, 2019.

[82] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.